Міністерство освіти та науки України

Національний університет водного господарства та природокористування

Кафедра автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій

04-03-262

Методичні вказівки

до виконання лабораторних робіт з дисципліни «**Програмування. Частина 1. Програмування мовою С++**» для здобувачів вищої освіти першого (бакалаврського) рівня за освітньо-професійною програмою «Автоматизація та комп'ютерно-інтегровані технології» спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології», 141 «Електроенергетика, електротехніка та електромеханіка» денної та заочної форм навчання

Рекомендовано науково –методичною радою з якості ННІ АКОТ
Протокол № 8 від 29.04.2020 р.

Рівне – 2020

Методичні вказівки до виконання лабораторних робіт з навчальної дисципліни «Програмування. Частина 1. Програмування мовою С++» для здобувачів вищої освіти першого (бакалаврського) рівня за освітньо-професійною програмою «Автоматизація та комп'ютерно-інтегровані технології» спеціальностей 151 «Автоматизація та комп'ютерно-інтегровані технології», 141 «Електроенергетика, електротехніка та електромеханіка» денної та заочної форм навчання [Електронне видання] / Сафоник А. П., Присяжнюк О. В. – Рівне : НУВГП, 2020. – 145 с.

Укладачі: Сафоник А. П., доктор технічних наук, професор, професор кафедри автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій (АЕКІТ) НУВГП; Присяжнюк О. В., кандидат технічних наук, старший викладач кафедри АЕКІТ НУВГП.

Відповідальний за випуск: Древецький В. В., д.т.н., професор, завідувач кафедри автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій.

Керівник групи забезпечення спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»: Древецький В. В.
Керівник групи забезпечення спеціальності 141 «Електроенергетика, електротехніка та електромеханіка»: Василець С. В., д.т.н., професор кафедри автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій

Зміст

Лабораторна робота №1
Програмування на основі лінійних алгоритмів

1.1. Мета роботи

Отримати знання і навички, необхідні для програмування з використанням лінійних алгоритмів та навчитися використовувати їх на практиці в процесі розроблення програм мовою програмування C++.

1.2. Теоретичні відомості

1.2.1. Поняття алгоритму. Типи алгоритмів

Алгоритм — система правил виконання обчислювального процесу, що обов'язково приводить до розв'язання певного класу задач після скінченного числа операцій. При написанні комп'ютерних програм алгоритм описує логічну послідовність операцій. Для візуального зображення алгоритмів часто використовують блок-схеми.

Кожен алгоритм є списком точно визначених інструкцій для розв'язання задачі. Починаючи з початкового стану, інструкції алгоритму описують процес обчислення, які відбуваються через послідовність станів, які, зрештою, завершуються кінцевим станом. Перехід з одного стану до наступного не обов'язково детермінований — деякі алгоритми містять елементи випадковості.

Алгоритм - це опис процесу вирішення того чи іншого завдання. Алгоритмом називається кінцевий набір правил, розташованих у певному логічному порядку, що дозволяє виконавцю вирішувати будь-яку конкретну задачу з деякого класу однотипних задач.

Алгоритми мають ряд важливих властивостей:

Скінченність. Алгоритм має завжди завершуватись після виконання скінченної кількості кроків. Процедуру, яка має решту характеристик алгоритму, без, можливо, скінченності, називають методом обчислень.

Дискретність. Процес, що визначається алгоритмом, можна розчленувати (розділити) на окремі елементарні етапи (кроки), кожен з яких називається кроком алгоритмічного процесу чи алгоритму.

Визначеність. Кожен крок алгоритму має бути точно визначений. Дії, які необхідно здійснити, повинні бути чітко та недвозначно визначені для кожного можливого випадку.

Вхідні дані. Алгоритм має деяку кількість (можливо, нульову) вхідних даних, тобто, величин, заданих до початку його роботи або значення яких визначають під час роботи алгоритму.

Вихідні дані. Алгоритм має одне або декілька вихідних даних, тобто, величин, що мають досить визначений зв'язок із вхідними даними.

Ефективність. Алгоритм вважають ефективним, якщо всі його оператори досить прості для того, аби їх можна було точно виконати за скінченний проміжок часу з допомогою олівця та аркушу паперу.

Виділяють наступні основні способи запису алгоритмів:

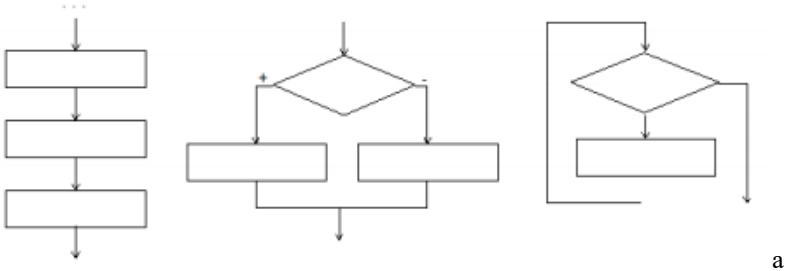
- ❖ вербальний - алгоритм описується на природній мові;
- ❖ символьний - алгоритм описується за допомогою набору символів;
- ❖ графічний - алгоритм описується за допомогою набору графічних зображень.

Таблиця 1.1 Таблиця основних елементів блок-схем

№. п\п	Елемент	Назва	Зміст
1.		Блок розрахунків	Розрахункові дії або послідовність дій
2.		Логічний блок	Вибір напрямку виконання алгоритму в залежності від певної умови
3.		Блок вводу-виводу даних	Загальне позначення вводу (виводу) даних
4.		Початок (кінець)	Початок або кінець алгоритму, вхід або вихід в підпрограму

Базові структури алгоритмів – це визначений набір блоків і стандартних способів їх з'єднання для виконання типових послідовних подій. До основних структур відносяться наступні:

- лінійні;
- розгалужені;
- циклічні.



а б в

Рис.1.1. Лінійна структура (а), розгалуження (б) та циклічна структура (в)

Лінійними називаються алгоритми, в яких дії відбувається послідовно одна за одною.

Розгалуженим називається алгоритм, в якому дія виконується по одній із можливих гілок рішення задачі, в залежності від виконання умов. В розгалужених алгоритмах є умова, в залежності від виконання або невиконання якої виконується та чи інша послідовність команд (дій). В якості умови в розгалуженому алгоритмі може бути використано будь-яке зрозуміле виконавцю твердження, яке може бути істинним або хибним. Таке твердження може бути виражене як словами, так і формулою.

Циклічним називається алгоритм, в якому деяка частина операцій (тіло циклу – послідовність команд) виконується багатократно. Організація циклів які ніколи не приводять до зупинки у виконанні алгоритму, являє собою порушенням вимоги його результативності – отримання результату за кінцеву кількість кроків.

1.2.2. Підходи до розробки алгоритмів

Процес вирішення складної задачі досить часто зводиться до вирішення більш простих задач. Відповідно процес розробки складного алгоритму може розбиватися на етапи складання окремих алгоритмів, які називаються допоміжними. Кожен такий допоміжний алгоритм описує рішення окремої задачі.

Процес побудови алгоритму методом послідовної деталізації полягає в наступному. Спочатку алгоритм формулюється в «крупних» блоках (командах), які можуть бути незрозумілими виконавцю, тобто не входять в систему його команд, і записуються як виклики допоміжних алгоритмів. Після того проходить деталізація і всі допоміжні алгоритми детально розписуються із використанням команд, які зрозумілі виконавцю.

Правила виконання схем алгоритмів регламентує ГОСТ 16.002-80

(єдина система програмної документації).

Мова С, створена Денісом Рітчі на початку 70-х років у Bell Laboratory американської корпорації AT&T, є однією з універсальних мов програмування. Мова С вважається мовою системного програмування. Вірніше сказати, що вона найбільше ефективна при рішенні задач системного програмування, хоча вона, безумовно, зручна і при написанні прикладних програм. Серед переваг мови С можна відзначити можливість перенесення програм, написаних на мові С, на комп'ютери різної архітектури і з однієї операційної системи в іншу, лаконічність запису алгоритмів, логічну стрункість і зручність читання програм, можливість одержати ефективний код програм, порівнянних по швидкості з програмами, написаними на асемблері. Зручність мови С заснована на тому, що вона є одночасно і мовою високого рівня, що має повний набір конструкцій структурного програмування, що підтримує модульність, блокову структуру програм, можливість роздільної компіляції модулів. У той же час мова С має набір низькорівневих засобів, що дозволяють мати зручний доступ до апаратних засобів комп'ютера, що зокрема дозволяють добратися до кожного біта пам'яті. Гнучкість і універсальність мови С забезпечує її широке поширення.

Перший опис мови було дано в книзі Б. Кернігана і Д. Рітчі, що була переведена на російську мову. Довгий час цей опис був стандартом, однак ряд моментів допускали неоднозначне тлумачення, що породило множину трактувань мови С. Для виправлення цієї ситуації при Американському національному інституті стандартів (ANSI) були утворені комітет по стандартизації мови С і в 1983 році були затверджені стандарт мови С, що одержав назву ANSI C.

На початку 80-х років у тієї ж Bell Laboratory Б'єрном Строуструпом (Yjarne Stroustrup) у результаті доповнення і розширення мови С була створена нова по суті мова, що одержала назву "С з класами". У 1983 році ця назва була замінена на С++.

Автор мови створював її з метою поліпшити мову С, підтримати абстракції даних і об'єктно-орієнтоване програмування. Мова С++ є мовою об'єктно-орієнтованого програмування. Концепція об'єктно-орієнтованого програмування виникнула не раптом. Ідея використання програмних об'єктів розвивалася різними дослідниками протягом багатьох років. Одним із представників мов такого типу є Simula 67. Більш докладно ми розповімо про особливості об'єктно-орієнтованого програмування нижче, а зараз коротко зупинимося на його якостях.

Об'єктно-орієнтована мова - мова програмування, на якій програма задається описом поводження сукупності взаємозалежних об'єктів. Об'єкти обмінюються запитами; реагуючи на отриманий запит, об'єкт надсилає

запит іншим об'єктам, одержує відповіді, змінює значення своїх внутрішніх перемінних і видає відповідь на отриманий запит. Механізм запитів в об'єктно-орієнтованих мовах відрізняється від механізму процедур у процедурних мовах тим, що при виконанні запиту об'єктом безпосередньо можуть бути змінені тільки значення перемінних цього об'єкта.

Об'єктно-орієнтоване програмування має справу з об'єктами, містить у собі створення об'єктів, що об'єднують дані і правила опрацювання цих даних. Об'єкти можуть містити у собі приватні, закриті (private) дані і правила їхнього опрацювання, доступні тільки об'єкту і його спадкоємцям, а також загальні (public) дані і правила, що доступні об'єктам і модулям в інших частинах програми. Важливою рисою об'єктно-орієнтованого програмування є спадкування, тобто можливість створювати ієрархічну послідовність об'єктів від більш загальних до більш специфічних. У цій ієрархії кожен об'єкт успадковує характерні риси об'єктів-прабатьків, об'єктів, що передують йому.

Таким чином, мови об'єктно-орієнтованого програмування містять у собі наступні основні риси: наявність об'єктів і інкапсуляцію даних, спадкування, поліморфізм, абстракцію даних. Надалі ми більш докладно зупинимося на цих поняттях і проілюструємо їхнє використання в мові C++.

1.2.3. Створення виконуваного коду програми

Написання програми передбачає виконання визначеного числа дій, що із більшою або меншою деталізацією можна розділити на наступні найважливіші етапи:

- постановка задачі;
- вибір методу вирішення задачі;
- написання вихідного тексту програми на мовах C і C++;
- введення вихідного тексту програми за допомогою текстового редактора; текст може бути розбитий на декілька файлів (модулів); на цьому етапі ми одержуємо файли вихідного тексту з розширенням .c або .cpp;
- компіляція модулів (кожного модуля окремо або всіх модулів разом); на цьому етапі ми одержуємо об'єктний файл, тобто файл із розширенням .obj;
- налагодження синтаксису програми;
- об'єднання відкомпільованих модулів у програму (це часто називають компонуванням або лінкуванням програми); на цьому етапі до програми приєднуються необхідні стандартні бібліотеки і ми одержуємо виконуваний файл із розширенням .obj);

- запуск програми на виконання;
- налагодження програми (тестування програми і алгоритму);
- остаточне оформлення програми.

При виконанні кожного із зазначених етапів програмування виникає необхідність повернення на попередні етапи, іноді аж до зміни постановки задачі. Сучасні системи програмування дозволяють зручно переходити від одного етапу написання програми до іншого. Це здійснюється наявністю так званого інтегрованого середовища програмування, що містить у собі текстовий редактор, компілятор, компоновщик, вбудований відладник і, в залежності від системи або її версії, дає програмісту додаткові зручності для написання і налагодження програм.

Деякі реалізації мови С++, наприклад, Microsoft Visual C++, Borland C++, реалізовані у вигляді інтегрованих середовищ розробки, що дозволяють виконувати всі вищеописані етапи створення виконуваного файлу в автоматичному режимі.

Розглянемо декілька вільних інтегрованих середовищ розробки (Integrated development environment, IDE), які можуть використовуватися для розроблення програм мовами С/С++:

– **Dev-C++.** У склад дистрибутива входить компілятор MinGW. Оригінальну версію було розроблено компанією Bloodshed Software. На даний момент продовження розробки виконується компанією Orwell. Завантажити останню версію середовища можна за посиланням http://sourceforge.net/projects/orwelldevcpp/;

– **Eclipse CDT.** Інтегроване середовище на базі платформи Eclipse. Завантажити останню версію середовища можна за посиланням https://eclipse.org/cdt/downloads.php.

– **Codeblocks та компілятор GCC MinGW.**

Процес встановлення та особливості налаштування та використання вказаних IDE наведено в додатку А.

1.2.4. Оголошення змінних

Дані в програмі можна розділити на змінні і константи. Перед використанням змінні і константи повинні бути оголошені за допомогою оператора оголошення.

Змінна – це іменована область пам’яті, в яку записуються значення відповідно до оголошеного типу під час виконання програми. Оголошення змінної відбувається наступним чином:

тип ім’я_змінної;

> *int a;*
> *float g, sum;*

Під типом даних розуміють множину допустимих значень цих даних і множину дозволених операцій над ними. Водночас тип даних визначає і розмір пам'яті, що займають змінні і константи даного типу.

Для опису основних типів мови С++ використовують службові слова:

- int (цілий);
- char (символьний);
- bool (логічний);
- float (дійсний);
- double (дійсний з подвійною точністю);
- void (порожній, не має значення).

Типи int, char, bool називають цілими, а типи float та double — дійсними з плаваючою крапкою. Код, що формує компілятор для обробки цілих величин, відрізняється від коду для величин з плаваючою крапкою.

Для уточнення внутрішнього подання та діапазону значень стандартних типів мова С++ використовує чотири специфікатори типу:

- short (короткий);
- long (довгий);
- signed (знаковий);
- unsigned (беззнаковий).

Типи даних, що використовуються у С++ наведені у таблиці 1.2.

Таблиця 1.2 – Типи даних у мові С++

Тип	**Розмір пам'яті, байт**	**Діапазон значень**
[signed] **char**	1	-128…127
unsigned **char**	1	0…255
[signed] **short** [int]	2	-32768…32767
unsigned **short** [int]	2	0…65535
[signed] **int**	4	-2147483648…2147483647
unsigned **int**	4	0… 4294967295
[signed] **long** [int]	4	-2147483648…2147483647
unsigned **long** [int]	4	0…4294967295
float	4	3.4e-38…3.4e38
double	8	1.7e-308…1.7e308
long **double**	10	3.4e-4932... 3.4e4932

Порядок обчислення виразу визначається розташуванням знаків операцій, круглих дужок і пріоритетами виконання операцій. Вирази із найвищим пріоритетом обчислюються першими.

Символ «=» означає бінарну операцію простого присвоювання, у результаті виконання якої значення правого операнду присвоюється лівому операнду:

ім'я_змінної = вираз;

Приклад оголошення змінних та константи:

```
int a=1, b;
const float g = 8.1;
```

1.2.5. Математичні функції мови С++

Для використання математичних функцій у програмах мовами С та С++ необхідно підключити заголовний файл <math.h>. Основні функції даної бібліотеки:

– *cos*(*x*) – косинус;
– *acos*(*x*) – арккосинус;
– *exp*(*x*) – експонента;
– *log*(*x*) – натуральний логарифм;
– *log10*(*x*) – десятковий логарифм;
– *round*(*x*) – повертає значення, округлене до цілого (значення, що повертається, є значенням з плаваючою комою);
– *floor*(*x*) – округлення до найближчого меншого цілого числа;
– *ceil*(*x*) – округлення до найближчого більшого цілого числа;
– *pow*(*x, y*) – піднесення *x* у степінь *y*;
– *sin*(*x*) – синус;
– *asin*(*x*) – арксинус;
– *tan*(*x*) – тангенс;
– *atan*(*x*) – арктангенс;
– *sqrt*(*x*) – квадратний корінь;
– *fabs*(*x*) – абсолютна величина для чисел з плаваючою крапкою;
– *random*(*x*) – виводить випадкове число від 0 до значення аргумента.

1.2.6. Приклади програм мовою С++

До складу кожної програми мовою С++ повинна входити головна

функція main(). Дана функція є початковою точкою входу в програму. Основну структуру програми мовою С++ наведено на рисунку:


```
#include <stdio.h>
/*Приклад 1*/
main()
{
int year;
year=2019;
printf("Зараз %d рік \n",year);
}
```

Перший рядок:

#include <stdio.h>

Він повідомляє компілятору про необхідність підключити файл stdio.h. Цей файл містить інформацію, необхідну для правильного виконання функцій бібліотеки стандартного введення/виводу мови С. Мова С передбачає використання деякого числа файлів такого типу, що називаються заголовними файлами (header files). У файлі **stdio.h** знаходиться інформація про стандартну функцію виводу **printf()**, що ми використовували.

Для виводу на екран символу служить функція **putchar().**

Другий рядок:

/ *Приклад 1* /

є коментарем.

При уважному розгляді програми зауважимо, що між 5-м і 6-м рядками знаходиться порожній рядок. Порожні рядки у мові С не роблять ніякого впливу і можуть бути вставлені для зручності читання програми.

Рядок

void main ( )

визначає ім'я функції. Будь-яка програма на мові С складається з однієї

або декількох функцій. Виконання програми починається з виклику функції main(). Тому кожна С-програма повинна містити функцію main().

Наступний рядок,

{

містить відкриваючу фігурну дужку (brace), що позначає початок тіла функції main(). Фігурні дужки в мові С завжди використовуються парами (відкриваюча і закриваюча). Дужку, що закриває, ми ще зустрінемо в нашій програмі.

Рядок

int year;

оголошує (declare) змінну, названу year, і повідомляє компілятору, що ця змінна ціла. У мові С усі змінні повинні бути оголошені перед тим, як вони будуть використані. Процес оголошення змінних містить у собі визначення імені (ідентифікатора) змінних (year) і вказівка типу перемінних (**int**).

Рядок

year=2017;

є оператором присвоєння. У цьому рядку змінній з ім'ям year присвоюється значення 2017. Зауважимо, що в мові С використовується просто знак рівності в операторі присвоєння. Всі оператори в мові С завершуються символом "крапка з комою".

Рядок

printf (“Зараз %d рік\n", year);

є викликом стандартної функції **printf()**, що виводить на екран деяку інформацію. Цей рядок складається з двох частин: імені функції printf і двох її аргументів "Зараз %d рік\n" і year, розділених комою. У мові С немає вбудованих функцій введення/виводу. Але бібліотеки мови С і Borland C++ містять багато корисних і зручних функцій введення/виводу. Функція **printf**, що ми використовували, є універсальною функцією форматного виводу.

Для виклику функції потрібно написати ім'я функції і у дужках зазначити необхідні фактичні аргументи. Перший аргумент функції **printf()** -це рядок у лапках "Зараз %d рік\n", що іноді називають керуючим рядком (control string). Цей рядок може містити будь-які символи або специфікації формату, що починаються із символу **'%'**. Звичайні символи просто відображаються на екрані у тому порядку, у якому вони йдуть.

Специфікація формату, що починається із символу **'%'**, вказує формат, у якому буде виводитися значення перемінної year, що є другим аргументом функції **printf().** Специфікація **%d** вказує, що буде виводиться ціле число в десятковому записі. Комбінація символів **'\n'** повідомляє функції **printf()** про необхідність переходу на новий рядок. Цей символ

називається символом нового рядка (newline).

Останній рядок програми:

}

містить закриваючу фігурну дужку. Вона позначає кінець функції **main().**

Якщо при наборі програми ви не допустили помилок, то ви одержите результат, про який написано вище. Якщо, крім того, ви відключили видачу попереджень, то ніяких повідомлень вам не буде видано.

Розглянемо другий приклад, у якому буде реалізовуватися введення даних із клавіатури. Для цього буде використовуватися бібліотечна функція **scanf(),** що дозволяє користувачу вводити інформацію з клавіатури під час виконання програми.

```
#include <stdio.h>
/*Приклад 2. Обчислення довжини кола.*/
void main()
{
int radius;
float len;
printf("Радіус= \n");
scanf("%d", &radius);
len=3.1415*2*radius;
printf("Радіус- %d \n довжина- %f \n", radius, len);
}
```

У цій програмі, у порівнянні з попередньою, використано декілька нових інструкцій.

По-перше, оголошені дві змінні двох різних типів: radius - ціле число (**int**); len – число з плаваючою комою (**float**), що містить дробову частину.

По-друге, використовується функція **scanf()** для введення з клавіатури значення радіуса кола. Перший аргумент функції **scanf() "%d"** вказує, що буде вводитися ціле десяткове число. Другий аргумент - ім'я перемінної, котрому буде привласнене введене значення. Символ **&** (амперсанд, ampersand) перед ім'ям перемінної radius необхідний для правильної роботи функції **scanf()**. Більш докладно необхідність використання символу **&** перед ім'ям змінної буде обговорюватися надалі. Ввід символьних даних із клавіатури здійснюється за допомогою наступних функцій:

getchar() – із відображенням введеного символу на екрані;

getch() – без відображення введеного символу на екрані.

Операції вводу - виводу можна також реалізувати із використанням

бібліотеки потокового вводу - виводу iostream (функції вводу – виводу з використанням класів C++):

ввід даних із клавіатури: **cin>><ідентифікатор змінної>** ;

вивід даних на екран: **cout<< <вираз>.**

Тут <вираз> - ідентифікатор змінної, рядок символів або арифметичний вираз.

Для включення у потік символу нового рядка (еквівалентний \n) служить endl. Приклад програми, що ілюструє використання операцій потокового вводу - виводу :

```
#include <stdio.h>
#include<iostream>
#include <windows.h>
using namespace std;
void main()
{
char ch='1';
int k,m; //опис змінних
float a; //опис змінних
SetConsoleOutputCP(1251);
SetConsoleCP(1251);
cout << ch; //вивід символа на екран
//ввід підказки для вводу даних та перехід на новий рядок
cout <<endl<< "введіть дані k,m,a"<<endl;
cin >> k >> m >> a ; //ввід даних
//вивід даних та перехід на новий рядок
cout << "k=" << k << "\tm=" << m << "\ta=" << a
<<endl;
}
```

Примітка: директива **using namespace std** вказує, що ми будемо працювати із іменами зі стандартної бібліотеки. При відсутності такої директиви замість cin необхідно писати std::cin, а замість cout - std::cout.

1.3. Програма роботи

1.3.1. Ознайомитися з основними теоретичними відомостями за темою роботи, використовуючи дані методичні вказівки, лекції, а також рекомендовану літературу.

1.3.2. Встановити GCC MinGW та Codeblocks або Dev-C++ або Eclipse CDT

1.3.3. Написати та скомпілювати програму згідно свого варіанту

1.3.4. Налаштувати компілятор згідно поставлених задач.
1.3.5. Оформити звіт по зробленій роботі

1.4. Обладнання та програмне забезпечення

1.4.1. Персональний комп'ютер.

1.4.2 Програмне забезпечення: Dev-C++ або Eclipse CDT або GCC MinGW та CodeBlocks

1.5. Порядок виконання роботи і опрацювання результатів

1.5.1. Запустити файл інсталяції вибраної IDE.
1.5.2. Слідкуючи за пунктами установки, встановити середовище на ПК.
1.5.3. Налаштувати компілятор згідно поставлених задач.
1.5.4. Згідно свого варіанту (завдання 1 та 2) написати дві програми у вибраному середовищі та запустити їх на виконання.
1.5.5.Оформити звіт по зробленій роботі.

Вимоги до звіту

Звіт повинен включати в себе:

- Титульний лист із зазначенням номеру варіанту
- Мету роботи
- Результати виконання індивідуальних завдань: блок-схеми алгоритмів, тексти програми, результати виконання програми, скопійовані з монітора
- Відповіді на контрольні запитання

Приклад виконання завдання 1. Розробити схему алгоритму і створити програмний проект для обчислення

$$y = \frac{0.2x^2 - x}{(\sqrt{3}+x)(1+2x)} + \frac{2(x-1)^3}{\sin^2 x + 1}$$

де x − довільна змінна, яку слід ввести.

Реалізація:

```
# include <iostream>
#include <stdlib.h>
#include <math.h>
using namespace std;
int main () {
setlocale(0,".1251");
double y, x;
cout<<"Введіть x= ";
cin>>x;
y=(0.2*x*x-x)/((sqrt(3)+x)*(1+2*x))+2*pow(x-1,3)/(pow(sin(x),2)+1);
cout<<"y= "<< y;
return 0;
}
```

1.6 Контрольні запитання

1.6.1. Яка основна структура програми мовою С++?
1.6.2. Для чого необхідна функція main?
1.6.3. Як створити виконуваний код програми?
1.6.4. Що таке змінна?
1.6.5. Яким чином відбувається оголошення змінної?
1.6.6. Що таке константа?
1.6.7.Які типи даних вам відомі?
1.6.8. Який розмір пам'яті відповідає відомим вам типам даних?
1.6.9. Які існують модифікатори типів?
1.6.10. Що таке IDE? Які існують сучасні IDE?
1.6.11. Як запустити програму на виконання?
1.6.12. Як записати у програмі коментар?
1.6.13. Чи впливають пробіли і порожні рядки на виконання програми?
1.6.14. Яку функцію обов'язково повинна містити програма?
1.6.15. Для чого призначені функції printf() та scanf() ?

Завдання 1

Варіант	**Завдання**	**Варіант**	**Завдання**
1	$z=\left(\frac{e^{-x}-12.34}{\lg x-\cos x^3}\right)^{-0.5}$	**16**	$y=\frac{e^{-3x}+tg(4x-1)}{\lvert\cos x\rvert+\sqrt{\cos 2x}}$
2	$y=\frac{\sqrt{x-1}-tg(x+1)}{\arccos x+\ln x}+2,75$	**17**	$y=\frac{e^{-3x}+\ln^3(x-1)}{\ln\lvert x+1\rvert+tg(x^2-1)}$

3	$g=\dfrac{\sin x^2-\cos^4(x-1)^2}{arctg(x+2,6)+\sqrt[3]{\ln x}}$	**18**	$y=\sin\left(\dfrac{x+2.3\cdot\lg(x+1)}{\sqrt{2\ln x}+\cos x}\right)$
4	$p=\dfrac{e^{-3x}+tg(3x-3)}{\lvert\sin x\rvert+\sqrt[4]{\cos x+\cos 2x}}$	**19**	$u=\ln\lvert 1-x\rvert+\dfrac{tg\,x-\sin^2 x}{1-\sqrt{\ln x}}$
5	$y=\dfrac{e^{-x}-4x-\ln^3 x}{\lg\lvert x+1\rvert+ctg(x^2-1)}$	**20**	$y=\dfrac{e^{-x}+tg(x-1)}{\lvert\ln x\rvert+\sqrt{sin\,x+\cos 2x}}$
6	$y=\arcsin\left(\dfrac{x\cdot\ln x}{1+\cos x}+1\right)$	**21**	$c=tg\lvert x\rvert-\dfrac{\sqrt{\ln 2+\ln x}}{\sqrt[3]{tg\,x}+\sqrt[4]{\cos x^{-1}}}$
7	$y=x+\dfrac{\sqrt{\arcsin x+arctg\pi x}}{\lg(13.4x+\pi)}$	**22**	$y=\dfrac{\sqrt{x+1}-\sin(x-\pi)}{\cos(x-3.1)+\ln^2 x}+x\cdot\lg x$
8	$y=\operatorname{arc}tg\,x+\dfrac{e^{0,6x-1}-\sqrt{(x+6,1)^3}}{\ln x+tg^2 x}$	**23**	$y=tg\left(\dfrac{x+3\cdot\lg(x+1)}{\sqrt{\ln 4x}+\cos x}\right)$
9	$u=0,3\cdot\lg e^{-x}+\dfrac{arctg\,x-\sin^2 x}{4\cdot\sqrt{\ln\lvert x-1\rvert}}$	**24**	$y=\dfrac{\sin^3 x-\cos(x-1)^2}{arctg(x+1)+\sqrt[3]{\lg x}}$
10	$y=e^2\cdot\lg x^4\cdot\dfrac{(x-0,5)^2-\cos x}{\sqrt{\lvert x+1\rvert+\lvert x\rvert}}$	**25**	$c=lg\lvert x\rvert-\dfrac{\sqrt{\ln(x-1)+\ln x}}{\sqrt[3]{tg\,x}+\sqrt{\cos(x-\pi)}}$
11	$y=\dfrac{e^{-x}-4\cdot\lg x}{\ln x-\cos\lvert x+1\rvert}$	**26**	$y=\dfrac{\sqrt{\sin(x+1)+arctg\pi x}}{\lg(x+2\pi)}-1$
12	$c=\sin^2 x-\dfrac{\sqrt{\lg\lvert 2-x\rvert+\lg x}}{\sqrt[3]{tg\,x}+\sqrt{\cos^3 x}}$	**27**	$y=\arcsin^2(x-1)+\left(\dfrac{x\cdot\ln x-2\sqrt{x}}{1+\cos x}\right)$
13	$y=\arccos\left(\dfrac{x-\lg x}{1+\cos 3x}+1\right)$	**28**	$y=\dfrac{e^{-x}-x\cdot\sin x-\ln^2 x}{\lg\lvert\cos x\rvert+ctg(x^2-1)}$
14	$c=arctg\,x-\dfrac{\sqrt{\ln 4+\ln x}}{\sqrt[3]{\lg 2.4}+\sqrt[5]{\cos x^{-1}}}$	**29**	$y=\arcsin x+\dfrac{e^{x-1}-\sqrt{(x+1)^5}}{\lg^3 x+tg\,x}$
15	$y=\dfrac{\ln e^{-x}+\cos(x-1)}{\ln^3 x+\sqrt{sin\,3x+\cos 2x}}$	**30**	$y=3\cdot\lg x^4\cdot\dfrac{(x-1)^2-tg\,x}{\cos x+\sqrt{\lvert x+1\rvert+\lvert x\rvert}}$

Завдання 2

Варіант	Завдання
1	$y=\frac{x^2-z^2}{\lg\lvert x-7\rvert}$, $x=\frac{\sin^2 a^3-\arcsin b}{\ln\lvert a+b\rvert-1}$, $z=\sqrt{\left\lvert\frac{a+b}{ab}\right\rvert+\pi}$, $a=3{,}5$, $b=-2.16$;
2	$z=\ln\left\lvert\frac{x\sqrt{x}+\cos^3 y^2}{1{,}604-arctg\, y}\right\rvert$, $x=\ln b+\frac{(a-b)^2}{b}$, $y=\sqrt[3]{\cos a^2+ab}$, $a=-0{,}2$, $b=7$;
3	$y=-\sqrt{\lvert\lg x-\ln z\rvert+1}$, $x=\frac{e^{-2,5a}+\sin a^3}{2\lg\lvert ba\rvert}$, $z=\frac{arctg^3(b-a)+\sqrt[3]{b}}{1+\log_b a}$, $a=0{,}6$, $b=3$;
4	$z=c\cdot e^{-2,5x+y^2}-\sqrt[3]{cx}$, $x=\frac{\lg\lvert c+\alpha\rvert}{arcctg\frac{\pi}{\alpha}}+0{,}1$, $y=\frac{\sin^2\frac{\alpha^3}{2}-ctg\frac{c}{4}}{\ln\lvert\alpha\rvert+\ln c^2}$, $c=4{,}5$, $\alpha=2$;
5	$z=\frac{\lvert x-1\rvert+e^{-y}}{12{,}34-\lg\sqrt{x}}$, $y=2a\sqrt[3]{a+b}$, $x=arcctg\frac{e^a+e^b}{\sqrt{a+e}}$, $a=1{,}75$, $b=0{,}4$;
6	$p=\frac{(-1)^x\cdot e^{-xy}+17{,}4}{\sqrt[3]{\sin^2 xy}}$, $x=\left(a^2+b^2\right)^{-4,1}$, $y=arctg^3\frac{1}{b}$, $a=-2{,}004$, $b=0{,}87$;
7	$r=ctg\frac{x+y}{(x-y)^2}+1{,}3$, $x=\sin^4 e^{-b}+\lvert ab\rvert$, $y=\ln\lvert a-b\rvert+\lg\frac{\pi}{a}$, $a=1{,}8$, $b=-0{,}62$;
8	$\varphi=\arccos\frac{x^2}{0{,}13}+\ln\lvert y^{-1}\rvert$, $x=\sqrt{(k+6{,}1)^3}$, $y=\ln k^4+\lg m^{-6}$, $k=14$, $m=0{,}42$;
9	$\alpha=\frac{e^{-3,5\lvert x\rvert+\sqrt{\pi}}}{arctg^3(y-1)}$, $x=a+\cos\frac{\pi}{b}$, $y=\ln\left\lvert\frac{\pi}{16}-b\right\rvert$, $a=\frac{1}{2}$, $b=1{,}4\cdot10^3$;
10	$t=\ln\lvert m-y\rvert+\cos^3 my$, $m=\sqrt{\lvert x+a\rvert}+1.3\cdot\lg\frac{\pi}{3}$, $y=a\cdot\sqrt[3]{\sin^4 x^3}$, $x=3$, $a=-1{,}7$;
11	$\varepsilon=e^2\cdot\lg x^4-\sqrt{\lvert y+1\rvert}$, $x=21{,}4(a-1)^2+\cos\frac{\pi}{b}$, $y=\ln\left\lvert\frac{\pi}{a}-b\right\rvert+tg^2 b$, $a=1$, $b=-4$;
12	$\gamma=tg\frac{x+1}{y-2}+\lg\lvert k+x\rvert$, $x=\sqrt{\lvert m+n\rvert^3}$, $y=\sqrt[3]{\lvert km-3\rvert}+\frac{\pi}{6}$, $m=3$, $n=-2{,}2$, $k=0{,}8$;
13	$j=\log_\pi x^{-m}+\left\lvert\frac{\pi}{5}-y\right\rvert$, $x=arcctg\frac{5{,}4}{m}+mn$, $y=\sqrt{\lvert m-3\rvert}+\ln n^2$, $m=-2$, $n=3{,}87$;
14	$f=\frac{x^e-e^{-x}+0{,}12}{\sqrt{\lvert\sin(y-1)\rvert}}$, $x=e^{-\pi}+\pi^{-e}$, $y=\lg a^3-arctga$, $a=6{,}45$;
15	$a=\sqrt{\lvert\pi-y\rvert}+\sin x+1{,}6$, $y=tg^4(\beta-1)^2-0{,}3$, $x=tg\frac{\alpha}{e}+(-2)^{\frac{\alpha}{2}}$, $\alpha=4{,}4$, $\beta=1{,}87$;

16	$n=arctg\left(\sin^2 x+tg^3 y\right)$, $x=\ln\lvert\alpha+2\rvert-\lg\lvert\beta-3\rvert$, $y=\sin^2\left(\alpha-\beta\right)^3$, $\alpha=5$, $\beta=-0{,}1$;
17	$b=\left(\beta+z\right)^{-e}+\sqrt[3]{z+0{,}1}$, $\beta=e^{k-1}+\lg\lvert k+x\rvert$, $z=\ln^3\lvert 2^x-1\rvert-12{,}47$, $x=0{,}03$, $k=4$;
18	$y=\omega x^{-3{,}1}+e^{\omega z}$, $x=tg\dfrac{z}{\omega}+ctg\sqrt{z}$, $z=\sqrt[3]{\ln\omega+\ln\omega^2}$, $\omega=2{,}77$;
19	$t=\dfrac{x^2-y^3}{e^{-(x+y)}}$, $x=\sqrt{e^y+y}$, $y=\sqrt[3]{\lg e}+\sqrt[5]{\lvert\cos e\rvert}$; $x=\cos\dfrac{\pi-z}{3}$, $z=\sqrt{\lvert 1+\sin^2 y\rvert}$, $y=-1$;
20	$m=\lg^2\lvert y-5{,}5\rvert+\sin^2\dfrac{y}{4}$, $y=\ln\lvert\pi-x\rvert+\lg\left\lvert\dfrac{\pi}{x}\right\rvert$, $x=\sqrt{\lvert\sin e^2+3{,}41\rvert}$;
21	$g=e^{-3{,}5z}+\ln z^4$, $z=\sqrt[5]{\left(x+6\right)^3}$, $x=21{,}4\left(\alpha-0{,}5\right)^2-\cos\dfrac{\pi}{\alpha}$, $\alpha=6{,}42$;
22	$t=\dfrac{\ln\lvert m-y\rvert+\cos^3 my}{\sqrt{\lvert m+y\rvert^3}+17{,}14}$, $y=\left(2m\right)^{-e}+arctg\sqrt{e}$, $m=2{,}7\cdot 10^{-3}$;
23	$\varphi=\cos\dfrac{x^2}{\pi}+\ln\lvert y^{-1}\rvert$, $x=e^{-\pi}+\pi^{-e}$, $y=\lg a^3-arctg\,a$, $a=2{,}3$;
24	$j=\lg\ x^{-m}+\left\lvert\dfrac{\pi}{4}-y\right\rvert$, $x=arcctg\dfrac{n}{m}$, $y=\sqrt{\lvert m-3\rvert}+\ln n^2$, $m=-2$, $n=2{,}4$;
25	$i=\left(-1\right)^{\lg m}\dfrac{\sqrt{m^3+2{,}5x}}{e^{-m}}$, $x=\cos^2\dfrac{\pi}{y}-29{,}45$, $y=\left(3m\right)^e$, $m=13{,}44$;
26	$a=\gamma\cdot\sqrt[3]{y+0{,}01}+\sin^2\pi x$, $y=tg^4\left(x-1\right)$, $x=\lg\lvert\gamma+6{,}6\rvert+0{,}77$, $\gamma=-3{,}41$;
27	$d=\sqrt{\sin^3\left(x-1\right)+\cos\gamma}$, $x=\log_\gamma\left\lvert\dfrac{\pi}{e}+1\right\rvert+tg\gamma$, $\gamma=23{,}41$;
28	$y=\sin 2x+tg3y$, $x=\ln\lvert\alpha-2\rvert-\lg\lvert y+2\rvert$, $y=e^{-\alpha}+\dfrac{\pi}{8}$, $\alpha=4{,}45$;
29	$h=\dfrac{\pi}{8}\sin^2\left(\dfrac{x-y}{8\pi}\right)$, $x=e^{-\pi}+y^{-e}+0{,}15$, $y=\arccos\left(\pi e\right)^{-1}$;
30	$z=\lg\lvert x+1\rvert-\ln^3\lvert 2^x-1\rvert$, $x=e^{ky-5{,}1}+\cos^2 ky$, $y=\sqrt{\lvert x-e\rvert}$.

Лабораторна робота №2
Розробка програм розгалуженої структури

2.1. Мета роботи

Навчитися складати програми розгалуженої структури. Вивчити типи циклічних алгоритмів та циклічні оператори мови С++.

2.2. Теоретичні відомості

2.2.1. Арифметичні операції

До арифметичних операцій мови С відносяться:

- вирахування й унарний мінус;
- + додавання;
- множення;
- / ділення;
- % % ділення по модулю;
- ++ збільшення на одиницю (increment);
- -- зменшення на одиницю (decrement).

Операції додавання, віднімання, множення і ділення діють так само, як і в більшості інших алгоритмічних мов. Вони можуть застосовуватися до всіх вбудованих типів даних. Операції виконуються зліва направо, тобто спочатку обчислюється значення лівого операнда, потім значення, яке стоїть справа від знака операції. Якщо операнди мають один тип, то результат арифметичної операції має той же тип. Тому, коли операція ділення / застосовується до цілих змінних або символьних змінних, залишок відкидається. Так, 11/3 буде дорівнює 3, а вираз 1/2 буде рівний нулю.

Операція ділення по модулю % дає залишок від цілочисельного ділення. Операція % може застосовуватися тільки до цілочисельних змінних. У наступному прикладі обчислюється ціла частина і залишок від ділення двох цілих чисел.

```
#include <stdio.h>
main() {
int x, y;
printf("Введіть ділен і дільник:");
scanf(“%d%d", &x, &y);
printf(“\nЦіла частина %d\n", x/y);
printf("Залишок від ділення %d\n", x%y);
}
```

Мова С дає користувачу ще дві дуже корисні операції специфічні саме для мови С. Це унарні операції ++ і --. Операція ++ додає одиницю до операнду, операція -- віднімає одиницю з операнда. Обидві операції можуть стояти перед операндом або після операнда (префіксна і постфіксна форми). Три написані нижче оператора дають той самий результат, але мають різницю при використанні у виразах:

x = x + 1 ; ++x ; x++.

Проста програма дозволить зрозуміти цю відмінність.

```
#include <stdio.h>
main()
{
int x=5;
int y=60;
x++;
++y;
printf("x=%d y=%d\n", x, y);
printf("x=%d y=%d\n", x++, ++y);
}
```

Результатом роботи цієї програми буде наступне:

x=6,y=61;

x=6, y=62.

Зверніть увагу на те, що надруковане значення x не змінилося при другому звертанні до функції printf(), а значення у збільшилося на одиницю. Насправді значення змінної x також збільшилося на одиницю, але вже після виходу з функції printf()/. Розходження у використанні префіксної ++x і постфіксної x++ форм полягає в наступному:

x++ - значення змінної x спочатку використовується у виразі і лише потім змінна збільшується на одиницю;

++x - змінна x спочатку збільшується на одиницю, а потім її значення використовується у виразі.

Пріоритет арифметичних операцій наступний:

1. ++, --
2. - (унарний мінус)
3. *, /, %
4. +, -

Операції, однакові по старшинству, виконуються в порядку зліва праворуч. Звичайно ж, для того щоб змінити порядок операцій, можуть використовуватися круглі дужки.

2.2.2. Операції відношення і логічні операції

Операції відношення використовуються для порівняння. Повний список операцій відношення в мові С наступний:

< менше,
<= менше або дорівнює,
> більше,
>= більше або дорівнює,
== дорівнює,
!= не дорівнює.

У мові С є також три логічні операції:

&& і (AND),
|| або (OR),
! не (NOT).

Операції відношення використовуються в умовних виразах, або, коротше, умовах. Приклади умовних виразів:

a<0, 101>=105, 'a'=='A', 'a'!='A'

Кожний умовний вираз перевіряється: істинний він або ні. Точніше варто сказати, що кожний умовний вираз отримує значення "істинно" ("true") або "невірно" ("false"). У мові С немає логічного (булевого, boolean) типу. Тому результатом логічного виразу є цілочисельне арифметичне значення. У мові С "істинно" - це ненульове значення, "невірно" - це нуль. У більшості випадків у якості ненульового значення "true" використовується одиниця. Так, із приведених вище прикладів умовних виразів 2-е і 3-є отримали значення "нуль", а 1-е і 4-е - ненульові значення. Розглянемо приклад.

```
#include <stdio.h>
main()
{
int tr, fal;
tr = (101<=l05); /*вираз "істинний" */
fal = (101>l05); /*вираз "невірний" */
printf("true - %d false - %d\n", tr, fal);
return 0;
}
```

Логічні операції в мові С відповідають класичним логічним операціям AND(&&), OR (||) і NOT (!), а їхній результат - відповідним таблицям, що прийнято називати таблицями істинності:

X	Y	X AND Y	X OR Y	NOT X	X XOR Y
1	1	1	1	0	0
1	0	0	1	0	1
0	1	0	1	1	1
0	0	0	0	1	0

Операція XOR називається операцією "що виключає або". У мові С немає знака логічної операції XOR, хоча вона може бути реалізованою за допомогою операцій AND, OR і NOT. Однак надалі ми будемо розглядати побітові операції, серед яких операція "що виключає або" уже є.

Операції відношення і логічні операції мають пріоритет нижче чим арифметичні операції. Це значить, що вираз 12>10+1 розглядається як вираз 12>(10+1).

Пріорітет логічних операцій і операцій відношення наступний:

1. !
2. >, <, >=, <=
3. ==, !=
4. &&
5. ||

У логічних виразах, як і у всіх інших, можна використовувати круглі дужки, що мають найвищий пріоритет. Крім того, круглі дужки дозволяють зробити логічні вирази більш зрозумілими і зручними для читання. Умовні і логічні вирази використовуються в керуючих операторах мови С, таких, як if for і інших.

Особливість логічних операцій && і || перебуває в тому, що якщо при обчисленні результату операції (вир1) && (вир2) - значення лівого операнда (вир1) буде нульовим, то значення другого операнда на результат операції не зробить ніякого впливу. У цьому випадку другий операнд не обчислюється. А отже, надії на те, що при обчисленні другого операнда може відбутися збільшення якоїсь перемінної завдяки операції ++, не виправдаються. Те ж саме стосується операції ||. У цьому випадку другий операнд не обчислюється, якщо значення лівого операнда не нульове.

2.2.3. Операція присвоювання

Операція присвоювання в мові С позначається просто знаком =. На відміну від інших мов у мові С оператор присвоювання може використовуватися у виразах, що також містять у собі оператори порівняння або логічні оператори. У фрагменті

if ((f=x-y)>0) printf ("Число x, більше чим y)

спочатку обчислюється розмір x-y, що привласнюється перемінній f, потім

дорівнюється її значення з нулем. Ще одною особливістю використання оператора присвоювання в мові С є можливість записати оператор:

*a=b=c=x*y*

Таке багатократне присвоювання в мові С - звичайна справа, і виконується справа на ліво. Спочатку обчислюється значення x*y, потім це значення привласнюється c, потім b, і лише потім a. У лівій частині оператора присвоювання повинен стояти вираз, якому можна привласнити значення. Такий вираз в мові С, наприклад просто змінна, називається розміром lvalue. Вираз 2=2 помилковий, тому що константі не можна привласнити ніяке значення: константа не є розміром lvalue.

У мові С є додаткові операції присвоювання +=, -=, /=, *= і %=.

Замість виразу n = n + 5 можна використовувати вираз n += 5. Тут += адитивна операція присвоювання, у результаті виконання якої розмір, що стоїть справа, додається до значення перемінної, стоячої зліва.

Аналогічно

m-=20 те ж саме, що і m=m-20;

m*=20 те ж саме, що і m=m*20;

m/=l0 те ж саме, що і m=m/10;

m%=10 те ж саме, що і m=m%10.

Ці операції мають той же пріоритет, що й операція присвоювання =, тобто нижче, чим пріоритет арифметичних операцій. Є ще декілька додаткових операцій присвоювання, про які ми згадаємо нижче. Операція x+=5 виконується швидше, чим операція x=x+5.

Порозрядні операції (побітні операції)

Порозрядні операції можна проводити з будь-якими цілочисельними перемінними і константами. Не можна використовувати ці операції з перемінними типу float, double і long double. Результатом побітної операції буде цілочисельне значення.

Порозрядними операціями є:

& AND,

| OR,

^ XOR,

~ NOT,

<< зсув вліво,

>> зсув вправо.

Операції AND, OR, NOT і XOR нам уже відомі, вони цілком аналогічні відповідним логічним операціям. Тільки в цьому випадку ми порівнюємо не значення виразів, а значення кожної відповідної пари розрядів, (бітів).

Порозрядні операції дозволяють, зокрема, забезпечити доступ до кожного біту інформації. Часто порозрядні операції знаходять застосування в драйверах пристроїв, програмах, зв'язаних із принтером,

модемом і іншими пристроями.

При виконанні порозрядної операції над двома змінними, наприклад типу char, операція робиться над кожною парою відповідних розрядів. Відмінність порозрядних операцій від логічних і операцій відношення перебуває в тому, що логічні операції й операції відношення завжди в результаті дають 0 або 1. Для порозрядних операцій це не так.

Приведемо декілька прикладів порозрядних операцій. Якщо треба встановити значення старшого розряду змінної типу char рівним нулю, то зручно застосувати операцію & (AND):

```
ch=ch & 127;
Нехай ch='A'
'A' 11000001
01111111
---------------------------
'A'&127 01000001
```

Якщо ж ми хочемо встановити старший розряд рівним одиниці, то зручна операція OR:

```
ch = ch | 128;
'A' 11000001
10000000
---------------------------
‘A' | 128 11000001
```

Порозрядні операції зручні також для організації збереження в стиснутому виді інформації про стан on/off (включений/виключений). В однім байті можна берегти 8 таких прапорів.

Якщо перемінна ch є сховищем таких прапорів, то перевірити, чи знаходиться прапор, що міститься в третьому біті, у стані on, можна в такий спосіб:

```
if(ch & 4) printf("3 біт містить 1, стан on");
```

Ця перевірка грунтується на двійковому представленні числа 4=00000100.

Операції зрушення << і >> застосовні тільки до цілочисельних змінних. У результаті застосування цієї операції зрушуються усі біти лівого операнда на число позицій, обумовленої виразом справа відповідно вправо або вліво. Відсутні значення бітів доповнюються нулями. Форма цієї операції наступна:

value >> число позицій,

value << число позицій.

Приклад: двійкове представлення числа x=9: 00001001, тоді x=9<<3 01001000,

x=9>>3 00000001,

x=9>>5 00000000.

При застосуванні операції зрушення може відбуватися втрата старших або молодших розрядів. Застосування операцій << і >> по черзі до однієї і тієї ж перемінної може змінити значення цієї перемінної через втрату розрядів.

Нехай unsigned char x=255. Двійкове представлення змінної x має вид 11111111.

Значенням виразу x=x<<3 у двійковому виді буде 11111000.

Значенням виразу x=x>>3 у двійковому виді буде 00011111.

2.2.4. Операції () і []

У мові С круглі і квадратні дужки також розглядаються як операції. Причому ці операції мають найвищий пріоритет.

Порозрядні операції породжують ще декілька складних операцій присвоювання:

|=, &=, ^=, <<=, >>=.

2.2.5. Операція умова ? :

Операція умова - єдина операція мови С, що має три операнда. Ця операція має вид:

(вир1)? (вир2):(вир3)

Обчислюється вираз (вир1). Якщо цей вираз має ненульове значення, то обчислюється вираз (вир2). Результатом операції буде значення виразу (вир2).

Якщо значення виразу (вир1) дорівнює нулю, то обчислюється вираз (вир3) і його значення буде результатом операції. У будь-якому випадку обчислюється тільки одне з виражень: (вир2) і (вир). Наприклад, операцію умову зручно застосовувати для знаходження найбільшого з двох чисел x і y:

max = (x>y) ? x : y ;

або для знаходження абсолютного розміру числа x:

abs = (x>0) ? x : -x ;

Якщо другий і третій операнди є розмірами типу lvalue, тобто можуть стояти в лівій частині операції присвоювання, то і результат операції умова є розміром типу lvalue. За допомогою цієї операції можна в один рядок вирішити задачу: найбільше з чисел x або y замінити значенням 1:

(x>y)? x :y=1;

2.2.6. Операція кома

Операція кома має самий низький пріоритет із всіх операцій мов C и C++. Операція кома виконується зліва направо, і її значенням є значення правого операнда. У вираженні (вир1), (вир2) спочатку обчислиться значення (вир1), потім - значення (вир2). Це значення і буде результатом операції.

2.2.7. Операція sizeof

Операція sizeof має дві форми: sizeof (тип) або sizeof (вираз).

Результатом цієї операції є цілочисельне значення розміру типу або вираз в байтах. При використанні другої форми значення виразу не обчислюється, а лише визначається його тип. Приклади використання: sizeof (short int) або sizeof (x). У якості типу в операції sizeof не може використовуватися тип void.

Операції . і -> будуть визначені нижче.

Далі ми побачимо, що деякі знаки операцій мають декілька змістових значень. Так, знак операції & має два значення: бінарна операція побітне AND і унарна операція узяття адреси.

2.2.8. Керуючі оператори

Ми вже знаємо найпростішу форму двох таких операторів: if і for. Розглянемо ці й інші оператори докладніше. Керуючі оператори можна розділити на три категорії:

1. Умовні оператори if, if-else і switch.
2. Оператори циклу for, while і do-while.
3. Оператор безумовного переходу goto.

2.2.9. Умовний оператор if

Ми вже знаємо найпростішу форму оператора if. Повна форма оператора наступна:

if (умова) оператор;
else оператор;

Якщо значення умови "істинно", то виконується оператор (їм може бути складовий оператор - блок), що слідує за умовою. Якщо ж умова приймає значення "хибно" то виконується оператор, що слідує за ключовим словом else. У записі оператора if друга частина (тобто Оператор else) може бути відсутнім. Тоді, якщо умова приймає значення "хибно" виконується відразу наступний оператор програми. Насправді в

якості умови може стояти довільний вираз. В операторі if лише перевіряється, чи є значення цього виразу ненульовим (вірним) або нульовим (помилковим). За допомогою оператора if можна, наприклад, обчислити значення функції sgn(x) - знак x. Функція sgn(x) приймає значення 1, якщо x > 0, значення -1, якщо x < 0, значення 0, якщо x = 0.

```
#include <stdio.h>
int main( ) {
int sgn;
float x; printf("Введіть число:");  scanf("%f",%x);
if(x>0) { sgn=l;printf("число %f позитивне sgn");}
if(x==0) { sgn=0;pirintf("число %f дорівнює нулю sgn");}
if(x<0) {sgn= -l;printf("число %f негативне sgn");
return 0;}
```

Часто зустрічається необхідність використовувати конструкцію if-else-if:

```
if (умова) оператор;
else if (умова) оператор;
else if (умова) оператор;
   ……..
else оператор;
```

У цій формі умови операторів if перевіряються зверху вниз. Як тільки якась з умов приймає значення "істинно", виконається оператор, що слідує за цією умовою, а вся інша частина конструкції буде проігнорована. Оператори if попереднього приклада можуть бути записані в іншому виді:

```
#include <stdio.h>
int main( ) {
int sgn;
float x;
printf(" Введіть число:");  scanf("%f", &x);
if (x>0) { sgn=1;printf("Число %f позитивне \n", x); }
else if(x<0)  (sgn= -l;printf("число %f негативне \n", x); }
else (sgn=0;printf("число %f дорівнює нулю \n", x); }
return 0;}
```

У якості умови оператора if може використовуватися, як ми вже сказали, деякий вираз. Так, для того щоб перевірити, дорівнює число x нулю або не дорівнює, можна написати

```
if(x=0) printf("Число дорівнює нулю");
```

else printf("Число не дорівнює нулю");

Той же результат можна одержати наступним оператором:

if(!x) print("Число дорівнює нулю");

else print("Число не дорівнює нулю");

Вкладеним оператором if називається наступна конструкція:

if(x)

if (y) оператор1;

else оператор2;

У такій формі незрозуміло, до якого з операторів if відноситься else. В мові С оператор else асоціюється з найближчим if у відповідному блоці. В останньому прикладі else відноситься до if(y). Для того щоб віднести else до оператора if(x), потрібно відповідним чином розставити операторні дужки:

if(x)

{

if (y) оператор1;

}

else оператор2;

Тепер if(y) відноситься до іншого блоку.

2.2.10. Оператор SWITCH

Мова С має вбудований оператор множинного вибору, названий switch. switch (вираз)

{

case constant 1: послідовність операторів

break;

case constant2: послідовність операторів

break;

……..

case constantN: послідовність операторів

break;

default послідовність операторів

}

Спочатку обчислюється вираз в дужках за ключовим словом потім проглядається список міток (case constant 1 і т.д.) доти поки не знаходиться мітка, відповідна значенню обчисленого виразу. Далі відбувається виконання відповідної послідовності операторів, що випливають за двокрапкою. Якщо ж значення виразу не відповідає жодній з міток оператора switch, то виконується послідовність операторів, що слідують за

ключовим словом default.

Допускається конструкція оператора switch, коли слово default і відповідна послідовність операторів може відсутньому.

Ще один не відомий нам раніше оператор - break. Коли після послідовності операторів зустрічається ключове слово break, то виконання оператора break приводить до виходу з оператора switch і переходу до наступного оператора програми. Наявність оператора break в операторі switch необов'язково. Що буде, якщо операторів break не буде? Відповідь на це питання дадуть результати роботи наступних двох варіантів програми:

```
#include <stdio. h>
int main( ) {
char ch;
printf("Введіть заголовну літеру алфавіту:");
ch=getchar( );
if(ch>='A' && ch<='Я')
switch(ch)
{
case 'А': printf("Алексєєв \n"); break;
case 'Б': printf("Булгаков \n"); break;
case 'В': printf("Волошин \n"); break;
case 'Г': print("Гоголь\n"); break;
default: printf("Достоєвський, Зощенко й інші \n"); break;}
else printf("Треба було ввести заголовну літеру \n");
return 0;}
```

Припустимо, ви запустили першу програму і ввели літеру Б. Результатом роботи програми буде наступний рядок:

Булгаков

Виконався тільки один оператор, що відповідає мітці 'Б'. У випадку запуску іншої програми введення літери Б результат роботи програми буде наступний:

Булгаков

Волошин

Гоголь

Достоєвський, Зощенко й ін.

Ми бачимо, що виконалися всі оператори, починаючи з мітки 'Б', включаючи той, що випливає за словом default.

Оператор break закінчує послідовність операторів, що відносяться до кожної мітки. Далі виконується наступний оператор програми. Якщо ж

оператор break відсутній, то виконання продовжується до першого оператора break або до кінця оператора switch.

2.2.11. Цикли

Цикли необхідні, коли нам треба повторити деякі дії декілька разів, як правило, поки виконується деяка умова. У мові С відомі три види оператора циклу: for, while і do-while.

Цикл for

Основна форма циклу for має наступний вид:

for (ініціалізація ; перевірка умови ; зміна) оператор;

насправді в загальному виді

for (вираз1 ; вираз2 ; вираз 3) оператор;

У найпростішій формі ініціалізація використовується для присвоєння початкового значення параметру циклу. Перевірка умови - звичайно умовний вираз, використовується для зміни параметра циклу щораз при повторенні циклу. Ці три поділи заголовка циклу повинні бути розділені крапкою з комою. Виконання циклу відбувається доти, поки умовне вираз істинно. Як тільки умова стає помилковою, починає виконуватися наступний за циклом for оператор. Найпростіший приклад оператора циклу for:

for(i=0; i<10; i++) printf("%d\n", i);

У результаті виконання цього оператора будуть надруковані в стовпчик цифри від 0 до 9. Для друку цих цифр у зворотному порядку можна використовувати наступний оператор:

for(i=9; i>0; i++) printf("%d\n", i);

Цикл for схожий на аналогічні цикли в інших мовах програмування, і в той же час цей оператор у мові С набагато більш гнучкий, потужний і застосовується у багатьох ситуаціях.

У якості параметра циклу необов'язково використовувати цілочисельний лічильник. Приведемо фрагмент програми, що виводить на екран літери алфавіту:

unsigned char ch;

for (ch='A'; ch<="Я"; ch++) printfl("%c", ch);

Наступний фрагмент програми

for(ch='0'; ch! ='N';) scanf("%c", &ch);

буде виконуватися доти, поки з клавіатури не буде введений символ 'N'. Зауважимо, що місце, де повиний бути приріст, порожньо. Випадково або намірено може утворитися цикл, із котрого немає виходу, так називаний

безкінечний цикл.

Приведемо три приклада таких циклів.

for (;;)printf{“ Безкінечний цикл \n");

for(i=l;l;i++) printf("Безкінечний цикл\n");

for (i=10:i>6;i++) printf(“Безкінечний цикл\n");

Проте для таких циклів також може бути організований вихід. Для цього використовується оператор break, що ми зустрічали вище. Якщо оператор break зустрічається в складовому операторі циклу, то відбувається негайне припинення виконання циклу і починається виконання наступного оператора програми.

```
#include <stdio. h>
main( ) {
unsigned char ch;
for(;;)  {
ch=getchar( ); /*Прочитати символ */
if (ch=='Q') break; /'Перевірка символу */
printf("%c", ch); /*Друк символу */
      ...
return 0;}
```

У цій програмі будуть друкуватися введені символи доти, поки не буде введений символ 'Q'. Можливо, і це синтаксично правильно, наявність порожнього оператора (відсутність оператора) у циклі for.

Наприклад

for (i=0;i<10000;i++);

Цикли while і do-while

Наступний оператор циклу в мові С - це цикл while. Основна форма має наступний вид: while (умова) оператор; де оператор може бути простим, складовим або порожнім оператором. "Умова", як і у всіх інших операторах, є просто виразом. Цикл виконується доти, поки умова приймає значення "істинно". Коли ж умова прийме значення "хибно", програма передасть керування наступному оператору програми. Так само як і в циклі for у циклі while спочатку перевіряється умова, а потім виконується оператор. Це так названий цикл із передумовою.

На відміну від попередніх циклів у циклі do-while умова перевіряється наприкінці оператора циклу. Основна форма оператора do-while наступна:

do{ послідовність операторів }

while (умова);

Фігурні дужки необов'язкові, якщо всередині них знаходиться один оператор. Проте вони частіше усього ставляться для кращого читання програми, а також щоб не сплутати (програмісту, а не компілятору) з оператором while. Оператор do-while називається оператором циклу з післяумовою. Яка б умова не стояла наприкінці оператора, набір операторів у фігурних дужках один (перший) раз виконується обов'язково. У циклах for і while оператор може не виконатися жодного разу.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
/* Гра " Вгадай число " Програма вибирає випадкове число від 1 до
100, ви повинні вгадати його */
main( ) { int s, x, n=0;
randomize( );
s=random(100)+l;
do
{ printf("Введіть число від 1 до 100: ");
scanf("%d", &x);
n++;
if(s<x) print("Загадане число менше\n");
if(s>x) printf("Загадане число більше\n");
} while (s-x);
printf("Ви вгадали число ! \n");
printf("Затратили на вгадування %d спроб\n", n);
return 0;}
```

Вкладені цикли

Коли один цикл знаходиться усередині іншого, то говорять, що це вкладені цикли. Часто зустрічаються вкладені цикли, наприклад, при заповненні таблиць. Як приклад розглянемо програму друку таблиці множення цілих чисел.

```
#include <sidio. h>
int  main( )
{ int i, j;
for(i=0;i<10;i++) {
for(j=1;j<5;j++) printf("%d * %d = %2d ", i, j, i*j);
printf(\n");
...
```

Оператор break

Оператор break має два застосування. Перше - закінчення case в операторі switch. Друге - негайне закінчення циклу, не зв'язане з перевіркою звичайної умови закінчення циклу. Коли оператор break зустрічається усередині оператора циклу, то відбувається негайний вихід із циклу і перехід до виконання оператора, наступного за оператором циклу.

```
#include <stdio. h>
main( ) {
 int i;
 for(i=0;i<1000;i++)
  {
   printf("%d-%d \n", i, i*i*i);
   if (i*i*i>= 10000) break;
  }
}...
```

Оператор continue

Ще один корисний оператор - оператор continue. Якщо оператор continue зустрівся в операторі циклу, то він передає керування на початок наступної ітерації циклу. У циклах while і do-while - на перевірку умови, у циклі for - на збільшення. Цей оператор необхідний, якщо ви хочете закінчити поточну ітерацію циклу і не виконувати залишені оператори, а відразу перейти до наступної ітерації циклу. Наприклад, його можна використовувати в програмі, що друкує натуральні числа кратні семи.

```
#include <stdio. h>
main( )
{ int i;
  for(i=0;i<1000;i++)
  {
    if(i%7) continue;
    printf("%8d", i);
   }
}
```

2.2.12. Оператор goto

Мова С має всі можливості для написання добре структурованих

програм. Апологети структурного програмування вважають поганим тоном використання оператора goto, без якого було важко обійтися в таких мовах, як FORTRAN або BASIC. Проте оператор goto у мові С є й іноді може бути корисний, хоча без нього можна обійтися в будь-якій ситуації. Для використання оператора goto треба ввести поняття мітки (label). Мітка - це ідентифікатор, за яким випливає двокрапка. Мітка повинна знаходиться в тієї ж функції, що й оператор goto. Одне з корисних застосувань оператора goto - це вихід із вкладених циклів:

```
for( ) {
    ...
  goto exit;
  ...
 while ( ) {
...
goto exit; ... }
 }
exit: printf ("Швидкий вихід із вкладених циклів");
```

2.3. Програма роботи

2.3.1. Запустити середовище Dev-C++.

2.3.2. Скласти алгоритм програми для знаходження значення функції, яка обчислюється в залежності від значення аргументу, із завдання 1 згідно свого варіанту. Розробити блок-схему алгоритму.

2.3.3. Написати програму для обчислення значення функції завдання 1.

2.3.4. Скласти алгоритм програми для обчислення значення функції (завдання 2) на вказаному проміжку із заданим кроком, згідно свого варіанту. Намалювати блок-схеми алгоритмів.

2.3.5. Написати програми для обчислення значень функцій з завдання 2.

2.3.6. Скласти алгоритм програми для обчислення значення функції (завдання 3) на певному інтервалі із заданим кроком зміни аргументу згідно свого варіанту. Намалювати блок-схему алгоритму.

2.3.7 Скласти програму для обчислення значення функції з завдання 3.

Вимоги до програм:

Завдання 1: дані потрібно вводити з клавіатури, кінцеві значення та проміжні змінні виводити на екран у зручній формі.

Завдання 2:

- вхідні дані (початкове та кінцеве значення аргументу, крок зміни аргументу) ввести оператором введення;
- результат обчислення вивести у вигляді таблиці (колонка 1 - № точки, 2 – значення аргументу, 3 – значення функції);

- написати 2 варіанта програми: 1 – використовуючи оператор безумовного переходу goto, 2 – використовуючи оператор циклу.

Завдання 3:

- межі області визначення функції, крок зміни аргументу, початкове значення аргументу ввести оператором введення;
- результат обчислення вивести у вигляді таблиці (колонка 1 - № точки, 2 – значення аргументу, 3 – значення функції).

2.4. Обладнання та програмне забезпечення

2.4.1. Персональний комп'ютер.
2.4.2. Програмне забезпечення: IDE Dev-C++.

2.5. Контрольні запитання

2.5.1. Які ви знаєте арифметичні операції?
2.5.2. Назвіть операції відношення.
2.5.3. Які логічні операції застосовуються у мові С?
2.5.4. Як виглядає операція присвоєння у мові С?
2.5.5. Назвіть порозрядні операції, які можна застосовувати у мові С.
2.5.6. Для чого застосовується і яка форма запису операції умова ? : ?
2.5.7. Для чого призначена операція sizeof ?
2.5.8. Що таке мітка?
2.5.9. Оператор IF: форма запису, призначення.
2.5.10. Оператор switch: форма запису, призначення.
2.5.11. Які типи циклів Ви можете назвати?
2.5.12. Назвіть відомі Вам оператори циклу мови С.
2.5.13. Запишіть загальний вигляд циклічного оператора for.
2.5.14. Запишіть загальний синтаксис циклічного оператора while.
2.5.15. Відтворіть загальний запис циклічного оператора do-while.

Завдання 1

Варіант	**Завдання**	**Варіант**	**Завдання**
1	$y=\begin{cases}-\cos^2(x-\pi), -\pi<x<\frac{\pi}{4}\\ \sqrt{\lvert x+1\rvert}, \frac{\pi}{4}\le x\le 1\\ \frac{1}{x-1}, 1<x\end{cases}$	**16**	$y=\begin{cases}x\sqrt{x-5,4}, 0\le x<2\\ arctgx^2, 2\le x<8\\ \lg\lvert x-7,8\rvert, 8\le x\end{cases}$

2	$y=\begin{cases}-\dfrac{1,4+x}{\ln x}, 1<x<3,2\\ x^2-0,75, 0<x\le 1\\ \cos^3 x^2-\sin^3 x^2, x\le 0\end{cases}$	**17**	$y=\begin{cases}-arctg\dfrac{x+\pi}{x^2}, 0<x\le 1\\ \ln\left\|x^3\right\|, 1<x<10\\ e^{-x}, x\ge 10\end{cases}$
3	$y=\begin{cases}e^{-2,5x^3}+1, x<0, x\ne -1\\ \sqrt{\left\|\lg x-\ln x\right\|}, 1<x\le 5,5\\ \dfrac{x-1}{x-\sin^2 x}, x=-1, x>5,5\\ 2x, 0\le x\le 1\end{cases}$	**18**	$y=\begin{cases}\sin e^x-2, x\le 4\\ \dfrac{x^2-1,2}{x+4}, 4<x<10\\ x, x\ge 10\end{cases}$
4	$y=\begin{cases}2x\sqrt[3]{x^2+z^2}, 1<x<20,4\\ arctg(x-z), 0<x\le 1\\ e^{x+z}, x\le 0\end{cases}$	**19**	$y=\begin{cases}-\sqrt[3]{cx}, c>9, x<-1\\ ctg\dfrac{c}{x}, c<0, -1<x\le 1\\ \ln\left\|c^2-x^2\right\|, c>0, x<c\end{cases}$
5	$y=\begin{cases}\lg\left\|\dfrac{\pi}{16}-x\right\|, 0<x<\dfrac{1}{4}\\ (x^2-2,0,4)^{-3,14}, \dfrac{1}{4}<x<1\\ \arccos\dfrac{x}{4}, x\ge 1\end{cases}$	**20**	$y=\begin{cases}\dfrac{x+a}{e^{xa}}, \left\|xa\right\|<1, x<0\\ -\ln^2 x, 2<x, a\le 0\\ \lg\sqrt{a}, 0<a, 0\le x\le 2\end{cases}$
6	$y=\begin{cases}e^{-\left\|x\right\|}, 1\le x\\ \lg\sqrt{1-x^2}, \left\|x\right\|<1\\ arctgx, x\le -1\end{cases}$	**21**	$y=\begin{cases}x^e-e^{-x}, \left\|x\right\|<2\\ \lg x^2, x\le -2\\ \sin^2 x, x\ge 2\end{cases}$
7	$y=\begin{cases}2^{x-1}+2,71, \pi\le x<8,5\\ \sqrt{\left\|\pi-x\right\|}, 8,5<x<\pi\\ x-2,7,\ 8,5\le\left\|x\right\|\end{cases}$	**22**	$y=\begin{cases}0, x\le -10\\ ctg\dfrac{x-1}{e}, -10<x\le 0\\ \ln x, 10\le x\\ \sqrt{x^3}, 0<x<10\end{cases}$
8	$y=\begin{cases}\sqrt[3]{\lg x+\ln x^2}, x>1\\ e^{-x}+1, x\le 1\end{cases}$	**23**	$y=\begin{cases}e^{-x}+\left\|x^2-1\right\|, x>1\\ \lg\sqrt{\left\|1-x\right\|}, -\pi<x\le 1\end{cases}$

9	$y=\begin{cases}\arccos\dfrac{\pi-x}{2}, x<-1\\ e^{-x^2}, -1<x<1\\ \pi\ln^2 x, x>1\\ 10^{-3}, x=1\end{cases}$	**24**	$y=\begin{cases}sin(x+1), x\le -1\\ ctg\dfrac{x-1}{e}, -1<x\le 0\\ \ln x, 1\le x\\ \sqrt{x^3}, 0<x<1\end{cases}$
10	$y=\begin{cases}x\sqrt[3]{x^2+3}, 1<x<10\\ arctg(x-9), 0<x\le 1\\ e^{x+1}, x\le 0\end{cases}$	**25**	$y=\begin{cases}(-1)^{3x-1}x^2, x\ge 5\\ \ln x^{-1}, 0<x<1\\ \cos\lvert x-1\rvert, 1<x<5\end{cases}$
11	$y=\begin{cases}\arcsin\left(-x^2+1\right), x<0\\ \lg^2(2x)+4, 4, 0\le x\le 3\\ -e^{\frac{1}{x}}, x>3\end{cases}$	**26**	$g=\begin{cases}\dfrac{\pi}{8}\sin^2\left(\dfrac{x-y}{3}\right), x=1\\ y^{-e}, 1<x<5\\ e^{-x}, 5\le x<11\\ 0,15x, x>11\end{cases}$
12	$y=\begin{cases}\sqrt{\sin^3(x-1)}, -6\le x\le 2\\ e^{-x}, x\ge 6\\ 4,4\cdot\lg^3\lvert x\rvert, 6>x>2\\ \ln x^2, x<-6\end{cases}$	**27**	$y=\begin{cases}arctg(\pi x), 0<x<\pi\\ \ln(x-3,18), 2\pi\le x\\ \dfrac{1}{\sqrt{x-\pi}}, \pi<x<2\pi\\ \pi, x\le 0\end{cases}$
13	$y=\begin{cases}e^{-\pi}+x^{-e}, 1<x\le e\\ \ln^2(x-e), e<x<10^3\\ \dfrac{x^2}{e}, 0<x\le 1\\ 2^x ctg^3 x^2, x<0\end{cases}$	**28**	$y=\begin{cases}e^{-x}, 1<x<2\\ x^e+1, 2\le x\le 5\\ 1, x<1\end{cases}$
14	$y=\begin{cases}\sqrt{x}, x>0\\ 2-x^2, x\le 0\end{cases}$	**29**	$y=\begin{cases}\sin^2 x, x\le -1\\ \sqrt{-x}, -1<x<0\\ x-\lg x, x>1\end{cases}$
15	$y=\begin{cases}\dfrac{\sin x}{x}, x>0\\ 2x^2+\ln\lvert x\rvert, x<0\\ 0, x=0\end{cases}$	**30**	$y=\begin{cases}124-e^x, x<1\\ tg(x-1), 1\le x\le 10\\ 1, x>10\end{cases}$

Завдання 2

Варіант	Завдання
1	$y=\frac{\cos^2 x}{x^2+1}$, $3{,}8\le x\le 7{,}6$, $\Delta x=0{,}6$;
2	$y=\frac{tg\,0{,}5x}{x^3+7{,}5}$, $0{,}1\le x\le 1{,}2$, $\Delta x=0{,}1$;
3	$y=\frac{e^{2x}-8}{x+3}$, $-1\le x\le 2{,}3$, $\Delta x=0{,}7$;
4	$y=\frac{x+\cos 2x}{3x}$, $2{,}3\le x\le 5{,}4$, $\Delta x=0{,}8$;
5	$y=\frac{x+\cos 2x}{x+2}$, $0{,}2\le x\le 10$, $\Delta x=0{,}8$;
6	$y=\frac{\cos^3 t^2}{1{,}5t+2}$, $2{,}3\le t\le 7{,}2$, $\Delta t=0{,}8$;
7	$z=\frac{x^3+2x}{3\cos\sqrt{x}+1}$, $0\le x\le 2$, $\Delta x=0{,}4$;
8	$z=\frac{t+\sin 2t}{t^2-3}$, $2{,}4\le t\le 6{,}9$, $\Delta t=0{,}4$;
9	$y=\frac{x^3-2}{3\ln x}$, $4{,}5\le x\le 16{,}4$, $\Delta x=2{,}2$;
10	$z=\frac{2{,}3t+8}{2\cos t+1}$, $0\le t\le 6{,}5$, $\Delta t=1{,}1$;
11	$y=\frac{\arccos x}{2x+1}$, $0{,}1\le x\le 0{,}9$, $\Delta x=0{,}1$;
12	$y=\frac{5tg(x+7)}{(x+3)^2}$, $1{,}2\le x\le 6{,}3$, $\Delta x=0{,}2$;
13	$y=\frac{1{,}5t-\ln 2t}{3t+1}$, $2{,}5\le t\le 9$, $\Delta t=0{,}8$;
14	$y=\frac{2{,}5x^3}{e^{2x}+2}$, $0\le x\le 0{,}5$, $\Delta x=0{,}1$;
15	$y=\frac{3x-2}{2arctg\lvert x\rvert+1}$, $3{,}2\le x\le 5{,}2$, $\Delta x=0{,}4$;
16	$y=\frac{5\lg x}{x^2-1}$, $1{,}2\le x\le 3{,}8$, $\Delta x=0{,}4$;
17	$z=\frac{6x+4}{\sin 3x-x}$, $2{,}3\le x\le 7{,}8$, $\Delta x=0{,}9$;

18	$z=\frac{2\sin^2(x+2)}{x^2+1}$, $7,2\le x\le 12$, $\Delta x=0,5$;
19	$y=\frac{(3x+2)^2}{\sin x+3}$, $4,8\le x\le 7,9$, $\Delta x=0,4$;
20	$y=\frac{2\sin^3 x}{3x+1}$, $-1\le x\le 1$, $\Delta x=0,25$;
21	$y=\frac{tg2t-3t}{t+3}$, $0,2\le t\le 0,8$, $\Delta t=0,1$;
22	$y=\frac{3x+1}{arctgx}$, $0,1\le x\le 1,5$, $\Delta x=0,2$;
23	$y=\frac{2t+8}{\cos 3t+1}$, $2\le t\le 6,5$, $\Delta t=0,8$;
24	$y=\frac{\arccos x}{3x+1}$, $0,1\le x\le 0,9$, $\Delta x=0,1$;
25	$y=\frac{(x+2)^2}{\sqrt{x^2+1}}$, $2,3\le x\le 8,3$, $\Delta x=0,6$;
26	$y=\frac{t-\ln 2t}{3t+1}$, $2,1\le t\le 8,5$, $\Delta t=0,7$;
27	$y=\frac{x^2+2x}{\cos 5x+2}$, $-2\le x\le 4,5$, $\Delta x=0,5$;
28	$y=\frac{\ln\lvert x+1\rvert+5}{2x+3}$, $0,2\le x\le 0,9$, $\Delta x=0,15$;
29	$y=\frac{x+\cos 2x}{3x}$, $2,7\le x\le 8$, $\Delta x=0,7$;
30	$z=\frac{\arcsin 2x+\lvert x\rvert}{x^2+1}$, $0\le x\le 0,4$, $\Delta x=0,2$.

Завдання 3

Варіант	Завдання
1	$w=\begin{cases} y+\sin y, & -6,5<y<0,5 \\ \ln\left(y+\sqrt[3]{y}\right), & 0,5\le y\le 8;\Delta y=0,5; \end{cases}$
2	$x=\begin{cases} 1,26^{v}+v, & 0\le v<0,1 \\ arcctg\,(v+0,4), & 0,1\le v<4;\Delta v=0,1 \end{cases}$
3	$f=\begin{cases} y+0,1\cos y, & -2<y\le 0,5 \\ \lg\left(y+\sqrt{y+0,6}\right), & 0,5<y\le 3;\Delta y=0,5 \end{cases}$
4	$y=\begin{cases} \sin x+e^{x}, & -2\le x\le 0 \\ arcctg\,(x-0,3), & 0<x\le 3;\Delta x=0,5 \end{cases}$
5	$w=\begin{cases} z-\sin z, & -2\le z\le 0,5 \\ arcctg\,(x-0,3), & 0,5<z\le 3;\Delta z=0,5 \end{cases}$
6	$v=\begin{cases} t+\cos t, & 0\le t\le 0,5 \\ arctg\,(t+\ln t), & 0,5<t\le 2;\Delta t=0,3 \end{cases}$
7	$y=\begin{cases} arcctgx+e^{x}, & 0\le x\le 0,5 \\ \ln(x+\sin x), & 0,5<x\le 8;\Delta x=0,5 \end{cases}$
8	$w=\begin{cases} 0,3^{v}-v^{2}+\cos v, & -3<v \\ ctg\,(0,34v-0,2), & 1<v\le 7;\Delta v=1 \end{cases}$
9	$z=\begin{cases} x^{3}+\sin x, & 0\le x\le 0,3 \\ arctg\,(x+\ln x), & 0,3<x\le 2;\Delta x=0,3 \end{cases}$
10	$w=\begin{cases} 0,6v-0,3^{v}, & -2<v\le 0,3 \\ \ln\left(v+\sqrt{v+\cos v}\right), & 0,3<v\le 5;\Delta v=0,5 \end{cases}$
11	$u=\begin{cases} x-0,8\sin x, & 0\le x\le 2,2 \\ arctg\,(\ln x+0,3), & 2,2\le x\le 3;\Delta x=0,4 \end{cases}$
12	$v=\begin{cases} \cos z-z, & 0\le z<0,5 \\ \ln\left(z+\sqrt{z}\right), & 0,5<z\le 7;\Delta z=4;z\ne 4 \end{cases}$
13	$u=\begin{cases} 1,3t-\sin t, & -4\le t\le -2 \\ \lg\left(t+\sqrt{t}\right), & -2<t\le 4;\Delta t=0,5 \end{cases}$
14	$u=\begin{cases} 0,2t+arctgt, & -2\le t\le 0 \\ \arcsin(0,25t), & 0<t\le 5;\Delta t=0,8 \end{cases}$

15	$y=\begin{cases} arcctgz+z, \\ \lg z+\sqrt{z}, \end{cases}$	$-2\le z<0$ $0<z\le 5;\Delta z=0,5$
16	$r=\begin{cases} z+\cos z, \\ arctg\,(z+\ln z), \end{cases}$	$-1<z\le 0$ $0<z\le 1;\Delta z=0,4$
17	$w=\begin{cases} v^2+\sqrt[3]{v}, \\ \ln(v+\sin v), \end{cases}$	$0\le v\le 0,5$ $0,5<v\le 8;\Delta v=0,5$
18	$y=\begin{cases} x-e^x, \\ arcctg\left(x+\sqrt{x}-1.4\right), \end{cases}$	$-2\le x<2$ $2\le x\le 5;\Delta=0,5$
19	$t=\begin{cases} 1.3y+\sin y, \\ arctg\left(y+\sqrt{y}\right), \end{cases}$	$0\le y\le 0,3$ $0,3\le y\le 2;\Delta y=0,3$
20	$x=\begin{cases} w+\cos w, \\ arctgw-\lg\left(w+\sqrt{w}\right), \end{cases}$	$0\le w<0,5$ $0.5\le w\le 2;\Delta w=0,2$
21	$t=\begin{cases} x^2-e^x, \\ \ln(arctgx+x), \end{cases}$	$0\le x<0,4$ $0,4<x\le 2;\Delta x=0,2$
22	$f=\begin{cases} t+e^x, \\ \sqrt{t}-\ln(t+arctg\,t), \end{cases}$	$-0,5\le t\le 0,5$ $0,5<t\le 4,5;\Delta t=0,5$
23	$u=\begin{cases} arctg\,v-e^v, \\ \lg(v+\cos v), \end{cases}$	$0\le v\le 1$ $1\le v\le 3;\Delta v=0,5$
24	$x=\begin{cases} \sin z-z^2, \\ arctg\left(\ln z+\sqrt{z}\right), \end{cases}$	$0\le z\le 0,2$ $0,2\le z\le 3;\Delta z=0,2$
25	$f=\begin{cases} v^2-\sqrt[3]{v}, \\ arcctg\,(v+\ln v), \end{cases}$	$-2<v<0$ $0\le v<3;\Delta v=0,5$
26	$z=\begin{cases} x-\ln x, \\ x^2-2, \end{cases}$	$1\le x<2$ $2\le x\le 5;\Delta x=0,2$
27	$y=\begin{cases} t^2-\sin t, \\ \sin t-t^2, \end{cases}$	$-\pi\le t<0$ $0\le t\le \pi;\Delta t=0,1$
28	$f=\begin{cases} x\left\|x^2-3\right\|, \\ \sqrt{x^4}, \end{cases}$	$0\le x<\sqrt{3}$ $\sqrt{3}\le x\le 10;\Delta x=0,5$
29	$y=\begin{cases} x^3\cos x, \\ x^2\sin x^2, \end{cases}$	$-\pi\le x\le 0$ $0<x\le \pi;\Delta x=0,1$
30	$z=\begin{cases} t-5t^2, \\ \sqrt{t}+t^2, \end{cases}$	$5\le t<7$ $7\le t\le 10;\Delta t=1$

Лабораторна робота №3
Розробка програм з одновимірними масивами.

3.1. Мета роботи

Вивчити організацію однотипних даних у вигляді масивів на мові С++, їх оголошення, способи доступу до елементів та програмування алгоритмів обробки.

3.2.Теоретичні відомості

3.2.1. Вказівники

Покажчики (вказівники) – змінні, які зберігають адреси іншої змінної. Найчастіше ці адреси позначають місцезнаходження в пам'яті інших змінних. Наприклад, якщо змінна x містить адресу змінної *y*, то про змінну x говорять, що вона "вказує" на *y*.

Змінні-покажчики (або змінні типу покажчика) мають бути відповідно оголошені. Формат оголошення змінної-покажчика є таким:

*тип *ім'я_змінної;*

У цьому записі елемент тип означає базовий тип покажчика, причому він повинен бути допустимим С++-типом. Елемент ім'я_змінної є іменем змінної-покажчика.

Для розуміння сказаного розглянемо такий приклад. Щоб оголосити змінну *p* покажчиком на int-значення, використаємо таку настанову:

*int *p;*

Для оголошення покажчика на float-значення використаємо таку настанову:

*float *p;*

У загальному випадку використання символу "зірочка" (*) перед іменем змінної в настанові оголошення перетворює цю змінну на покажчик.

Тип даних, на які посилатиметься покажчик, визначається його базовим типом. Розглянемо ще один приклад:

*int *ip;* // Покажчик на цілочисельне значення

*double *dp;* // Покажчик на значення типу double

Як зазначено в коментарях до цієї програми, змінна *ip* – покажчик на int-значення, оскільки його базовим типом є тип int, а змінна *dp* – покажчик на double-значення, оскільки його базовим типом є тип double, Отже, в попередніх прикладах змінну *ip* можна використовувати для вказівки на int-значення, а змінну *dp* – на double-значення. Проте

запам'ятайте: не існує реального засобу, який би зміг перешкодити покажчику посилатися на "казна-що". Ось через це покажчики потенційно небезпечні.

3.2.2. Оператори роботи з покажчиками

З покажчиками використовуються два оператори: "*" і "&". Оператор "&" – унарний. Він повертає адресу пам'яті, за якою розташований його операнд1. Наприклад, у процесі виконання такого фрагмента коду програми

ptr = &balance;

у змінну ptr поміщається адреса змінної balance. Ця адреса відповідає області у внутрішній пам'яті комп'ютера, яка належить змінній *balance*. Виконання цієї настанови ніяк не впливає на значення змінної *balance*. Призначення оператора "&" можна "перекласти" українською мовою як "адреса змінної", перед якою він знаходиться. Отже, наведену вище настанову присвоєння можна виразити так: "змінна *ptr* отримує адресу змінної *balance*". Щоб краще зрозуміти суть цього присвоєння, припустимо, що змінна *balance* розташована у області пам'яті з адресою 100. Отже, після виконання цієї настанови змінна *ptr* набуде значення 100.

Другий оператор роботи з покажчиками (*) слугує доповненням до першого (&). Це також унарний оператор, але він звертається до значення змінної, розташованої за адресою, заданою його операндом. Іншими словами, він посилається на значення змінної, яка адресується заданим покажчиком. Якщо (продовжуючи роботу з попередньою настановою присвоєння) змінна *ptr* містить адресу змінної *balance*, то у процесі виконання настанови

*value = *ptr;*

змінній *value* буде присвоєне значення змінної *balance*, на яку вказує змінна *ptr*. Наприклад, якщо змінна *balance* містить значення 3200, то після виконання останньої настанови змінна value міститиме значення 3200, оскільки це якраз те значення, яке зберігається за адресою 100. Призначення оператора "*" можна виразити словосполученням "за адресою". У цьому випадку попередню настанову можна прочитати так: "змінна *value* набуває значення (розташоване) за адресою *ptr*". Дію наведених вище двох настанов схематично показано на рис. 3.1.

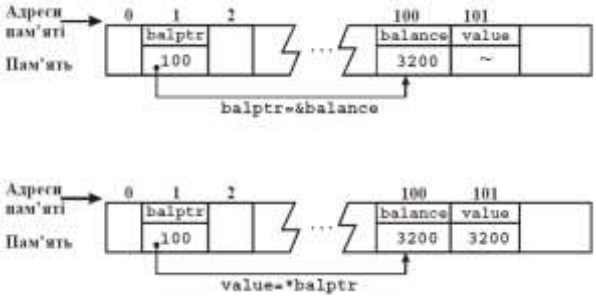


Рис. 3.1. Дія операторів "" і "&"*

Послідовність операцій, відображених на рис. 3.1, безпосередньо реалізується у наведеному нижче коді програми.

```
#include <iostream> // Потокове введення-виведення
#include <conio> // Консольний режим роботи

using namespace std; // Використання стандартного простору імен

int main() {
int balance;
int *ptr;
int value;
balance = 3200;
ptr = &balance;
value = *ptr;
cout << "Баланс дорівнює: " << value << endl;
getch(); return 0;
}
```

Внаслідок виконання ця програма відображає на екрані такі результати: Баланс дорівнює: 3200

Знак множення (*) і оператор із значенням "за адресою" позначаються однаковими символами "зірочка". Ці операції ніяк не пов'язані одна з іншою.

Оператори "*" і "&" мають вищий пріоритет, ніж будь-який з арифметичних операторів, за винятком унарного мінуса, пріоритет якого такий самий, як у операторів, що вживаються для роботи з вказівниками.

Під час присвоєння значення області пам'яті, яка адресується

покажчиком, його (покажчик) можна використовувати з лівого боку від оператора присвоєння. Наприклад, у процесі виконання такої настанови (якщо p – покажчик на цілочисельний тип)

**p = 101;*

число 101 присвоюється області пам'яті, в *p*, яка адресується покажчиком. Таким чином, цю настанову можна прочитати так: "за адресою *p* поміщаємо значення 101". Щоб інкрементувати або декрементувати значення, розташоване у області пам'яті, яка адресується покажчиком, можна використовувати настанову, подібну до такої:

*(*p)++;*

Круглі дужки тут є обов'язковими, оскільки оператор "*" має нижчий пріоритет, ніж оператор "++".

3.2.3. Масиви

Масивом називають кінцеву сукупність елементів одного типу.

Оголошення *k*-мірного масиву здійснюється згідно наступного формату:

<тип> <ім'я масиву> [*i*1][*i*2]*…*[*ik*]

де ім'я масиву є правильним ідентифікатором, *i*1, *i*2, ..., *in* – максимальна кількість елементів масиву по кожному виміру.

Кожен індекс масиву повинен записуватися в окремих квадратних дужках [ ]. Кількість індексів масиву необмежена. За замовчуванням загальний розмір масиву не повинен перевищувати одного сегменту (64К). Для оголошення масиву, більшого за 64К, між типом та ім'ям масиву вказується модифікатор huge.

Індекс масиву по кожному виміру змінюється від 0 до *m*-1, де *m*=1,…,*k*.

В якості типу масиву можуть вказуватися:

1) один із основних типів (int, char, float, double або синоніми);
2) тип іншого масиву;
3) вказівник, у тому числі вказівник на функцію;
4) структура (structure);
5) об'єднання (union).

Наприклад:

int mas[10]; /*масив із 10 цілих */
char line[80]; /*масив із 80 символів*/
float a[5][4]; /*двомірний масив із 5x4 елементів типу float*/
double b[7][7][7]; /* масив із 7x7x7 елементів типу double*/
*int *x[10]* /*масив із 10 вказівників на тип int*/

При оголошенні масиву допускається його ініціалізація. Початкові значення елементів повинні задаватися через кому у фігурних дужках { }. Значення елементів по кожному окремому виміру може охоплюватися фігурними дужками, між якими ставиться кома, наприклад:

int year[12]={31,28,31,30,31,30,31,31,30,31,30,31};

float mas[2][3]={{0.0,0.1,0.2},{1.0,1.1,1.2}};

Остання ініціалізація еквівалентна наступній:

float mas[2][3]={0.0,0.1,0.2,1.0,1.1,1.2}.

Кількість елементів не може бути більшою від вказаної розмірності. Якщо кількість елементів менша від вказаної розмірності, то присвоєння відбувається тільки для початкових елементів, а інші елементи приймають нульові значення, якщо масив оголошений як зовнішній або статичний, і приймають невизначені значення в інших випадках:

static int vec[10]={1,8,3}; /*Перші три елементи приймуть задані значення, а наступні сім – нульові значення*/

auto float matr[3][4]={{0.0}, {1.0,1.1},{2.0,2.1,2.2}};

/*елементи з індексами [0][0],[1][0],[1][1],[2][0],[2][1],[2][2] приймуть значення, а всі інші – невизначені значення*/.

При явній ініціалізації масиву, його розмірності можна не вказувати. Тоді розмірність масиву визначається кількістю заданих елементів, наприклад:

int i_array []={4,-2,7,10,-3,5,3}

/*оголошений та ініціалізований масив із семи цілих чисел*/

При явній ініціалізації двовимірних масивів перший індекс може бути опущений, а другий вказується обов'язково:

float f_mas[][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};

В даному прикладі оголошений масив, що складається з 3-х рядків і 4-стовпчиків. Наведене оголошення еквівалентне наступному:

float f_mas[][4]={1,2,3,4,5,6,7,8,9,10,11,12};

Згідно ANSI–стандарту ініціалізувати масив можна будь-де, у тому числі і всередині функції для старих компіляторів. Щоб ініціалізувати масив в середині функції, його потрібно оголосити як static:

static float vec[]={5.2,6.3,8.4};

У пам'яті елементи масиву розміщуються у неперервній області, так що правий індекс змінюється першим.

Приклад розміщення у пам’яті масиву int matrix[2][3]

[0][0]	[0][1]	[0][2]	[1][0]	[1][1]	[1][2]
0	2	4	6	8	12 байт

Посилання на елемент масиву відбувається вказуванням імені масиву і його індексів. Індекси повинні бути константами, змінними, або виразами цілого типу. Кожен індекс береться в окремі квадратні дужки, наприклад:

i_array[3]; /*звертання до четвертого елемента масиву i_array*/

matrix[1][2]; /*звертання до елементу масиву matrix, який знаходиться на перетині другого рядка і третього стовпчика*/

Інший спосіб доступу до елементів масиву полягає у використанні вказівників.

Ім’я одного масиву є вказівником на його нульовий елемент, наприклад:

int vec{5};

vec-→*vec*[0] *vec*[1] *vec*[2] *vec*[3] *vec*[4]

vec=&vec[0];

vec+i=&vec[*i*];

**vec=vec*[0];

*(*vec+i*)=*vec*[*i*];

Відповідно до оголошення двовимірного масиву int matrix[2][3]; позначення matrix[i], де i-0,1, визначає адресу розміщення в пам’яті і-го рядка, а ім’я масиву matrix визначає вказівник на початок масиву int matrix[2][3]. Ім’я масиву matrix є сумісне з типом вказівника на масив з трьох цілих чисел: int (**vec*)[3];

вказівними на рядки	Елементи масиву		
matrix[0]	[0][0]	[0][1]	[0][2]
matrix[1]	[1][0]	[1][1]	[1][2]

Значення вказівників matrix та *matrix є однакові, але не сумісні по типу. Правила сумісності та еквівалентності приведені нижче:

matrix=&matrix[0]=matrix[][m];

matrix+i=&matrix[i];

**matrix=matrix[0]=&matrix[0][0];*

**(matrix+i)=matrix[i]=&matrix[i][0];*

**(matrix+i)+j=matrix[i]+j=&matrix[i][j];*

***matrix=*matrix[0]=matrix[0][0];*

***(matrix+i)=*matrix[i]=matrix[i][0];*

**(*(matrix+i)+j)=*(matrix[i]+j)=matrix[i][j];*

Аналогічно для оголошення тривимірного масиву int fx [5][4][8];

позначення fx[i] визначає адресу розміщення в пам’яті і-го двомірного

масиву розмірністю 4x8 елементів, а позначення fx[i][j] визначає адресу 8-елементного j-го рядка і-го двовимірного під масиву.

Враховуючи, що перші p індексів масиву (p<k) визначають адресу відповідного підмасиву, для звертання до елементів масиву чи їх вказівників допускається форма, аналогічна адресуванню комірок пам'яті через вказівники:

Наприклад:

*(ip)[ім'я масиву [i1][i2]/ *...*/'[i(p-1)]].*

де *ip* - змінна або вираз цілого тішу, але не константа.

Наприклад:

(i)[vec] /* еквівалентно звертанню vec[i] */

(j)[matrix[i]] /* еквівалентно звертанню matrix[i][j]*/.

При роботі з масивами потрібно пам'ятати, що ім'я масиву є константою, тому не дозволяються операції модифікації значення, яке визначається цією константою, наприклад:

```
int a[5],y,n,
int *z;
/ * наступні операції не дозволяються *
a=y;  a++;  a+=n;  z=&a;
```

В мові С не можна присвоїти цілком один масив іншому такого ж типу та розмірності. Копіювання масивів здійснюється поелементно за допомогою операторів циклу, масивів вказівників

В мові С допускається використання масивів вказівників на всі типи даних, наприклад:

```
int *r[3];
```

де r[3] - масив, що містить адреси трьох елементів типу int.

Для доступу до значення, яке знаходиться за адресою r[i], необхідно використати операцію розіменування індексованого вказівника *r[i]. Робота із масивом вказівників демонструється наступним прикладом:

```
int i;
float x1=1, x2=2, x3=3, sum=0, *r[3];
r[0]=&x1;
r[1]=&x2;
r[2]=&x3;
for(i=0;i<3;i++) sum = sum+*r[i];
printf("Сума = %f\n",sum);
```

Для вводу елементів масиву використовується бібліотечна функція scanf, яка має наступний формат:

int scanf("форматний рядок", список вказівників на елементи масиву);

Для виводу елементів масиву використовується бібліотечна функція printf, яка має такий формат:

int printf("форматний рядок", список елементів масиву);

Форматний рядок для вводу-виводу повинен містити специфікації форматів списку даних. Специфікація формату повинна починатися із символу процента (%)

Код	Формат
%c	Символ
%d	Десяткове ціле із знаком
%i	Десяткове ціле із знаком
%e	Експоненціальне представлення (рядкова буква e)
%E	Експоненціальне представлення (прописна буква E)
%f	Значення з плинною крапкою
%g	Використовує коротший з двох форматів: %e або %f (якщо %e, використовує рядкову букву e)
%G	Використовує коротший з двох форматів: %E АБО%F (якщо %e використовує прописну букву e)
%o	Вісімкове ціле без знаку
%s	Рядок символів
%u	Десяткове ціле без знаку
%x	Шістнадцяткове ціле без знаку (рядкові букви)
%P	Покажчик Відповідний аргумент повинен бути покажчиком на ціле. Даний специфікатор зберігає у цьому цілому число символів, виведених у вихідний потік до поточного моменту (до виявлення специфікатора %n)
%%	Виводить символ %

Для вводу довгого цілого (long) або дійсного з подвійною точністю (double) перед символом формату вказується символ l. Для вводу типу long double перед символом формату вказується символ L.

У форматний рядок можуть входити інші символи таблиці ASCII.

Якщо такі символи включені у форматний рядок функції scanf, то вони повинні зустрітися у вхідному потоці. Якщо ж додаткові символи включені у форматний рядок функції printf, то вони будуть поміщені у вихідний потік. Таким чином можна виводити допоміжні повідомлення. Приклад вводу-виводу елементів одновимірного масиву

```
#include <stdio.h>
int main( ){
  float vec[10];
// Ввід
  for(int i=0;i<10;i++)
  scanf("%f",&vec[i]);
// Вивід
  for(int i=0;i<10;i++)
  printf("%g ", vec[i]);
 printf("\n");
return 0;
}
```

Приклад вводу-виводу елементів двомірного масиву:

```
#include <stdio.h>
int main(){
  int i,j,mas[3][4];
     // Ввід
 for(i=0;i<3;i++)
     for(j=0;j<4;j++)
          scanf("%d",&mas[i][j]);
   // Вивід по рядках
 for(i=0;i<3;i++){
      for(j=0;j<4;j++)
         printf(" %d ",mas[i][j]);
 printf("\n");
}
return 0;
```

3.3. Програма роботи

3.3.1. Запустити середовище IDE.

3.3.2. Скласти алгоритм програми для визначення деяких параметрів одновимірних масивів (завдання 1) згідно свого варіанту. Намалювати блок-схему алгоритму.

3.3.3. Скласти програму для визначення деяких параметрів одновимірних масивів (завдання 1) згідно свого варіанту.

3.3.4.Скласти алгоритм програми для визначення числових характеристик матриці (завдання 2) згідно свого варіанту.

3.3.5. Скласти програму для визначення числових характеристик

матриці (завдання 2) згідно свого варіанту.

Вимоги до програм:

вхідні дані (початкове та кінцеве значення індексної змінної, початкове значення аргументу, крок зміни аргументу) ввести оператором введення.

3.4. Обладнання та програмне забезпечення

3.4.1. Персональний комп'ютер.

3.4.2. Програмне забезпечення: IDE Dev-C++

3.5 Контрольні запитання

3.5.1. Як оголошується масив?

3.5.2. Які можливі типи масивів?

3.5.3. Який вигляд буде мати масив, якщо при його ініціалізації вказати кількість елементів, меншу за розмірність масиву?

3.5.4. Які є способи доступу до елемента масиву?

3.5.5. Як скопіювати один масив в інший?

3.5.6. Як описати масив вказівників?

3.5.7. Для чого здійснюється розіменування?

3.5.8. Як здійснюється ввід-вивід масивів?

3.5.9. Назвіть відомі вам специфікатори формату.

Завдання 1

Варіанти:

1) Знайти і надрукувати суму додатних елементів масиву

$$B(6)=(5.0;-2.3;-6.9;-1.1;2.0;6.6).$$

2) Підрахувати і надрукувати кількість додатних елементів, які стоять на парних місцях

$$C(8)=(-6.3;-1.0;10.3;-8.8;6.3;-1.1;0.0;0.1).$$

3) Вивести на друк середнє арифметичне від'ємних елементів масиву:

$$A(6)=(6.3;-2.1;4.2;5.3;-7.2;-4.5).$$

4) Знайти мінімальний елемент масиву

$$B(7)=(6.3;-1.6;1.1;0.1;-2.0;2.3;6.3).$$

5) Надрукувати суму від'ємних елементів, які стоять на парних місцях в масиві

$$X(17)=(-2.3;4.0;-8.9;6.3;4.9;-7.8;-6.5;5.1;3.8;-4.3;-5.1;7.2).$$

6) Надрукувати середнє арифметичне невід'ємних елементів масиву, які стоять на непарних місцях

$$B(10)=(6.3;0.0;-8.3;7.2;6.1;-4.2;5.7;6.4;5.6;-4.8).$$

7) Знайти і надрукувати кількість додатнім елементів масиву

$$C(9)=(1.6;2.1;-3.1;0.0;1.1;-2.2;3.7;8.9;9.2).$$

8) Обчислити добуток додатнім елементів масиву

$$D(5)=(1.1;-6.2;0.0;2.3;5.1).$$

9) Знайти суму елементів масиву

$$X(6)=(3.5;-6.3;2.1;0.1;5.1;-2.1),$$

значення яких менше 0,21.

10) Обчислити добуток модулів значень елементів масиву

$$Y(7)=(-2.2;0.2;3.1;2.1;-3.1;6.1;0.5).$$

11) Визначити кількість елементів масиву

$$B(5)=(2.2;3.1;-3.6;0.1;2.1),$$

значення яких менше 0,99.

12) Визначити кількість від'ємних елементів масиву

$$D(5)=(1.2;25.3;-2.3;-3.1;0.0).$$

13) Обчислити добуток елементів масиву

$$(B)=(2,3;4,3;-15,2;1,1;-1,2;-3,3),$$

значення яких більше 2.0.

14) Надрукувати порядкові номери від'ємних елементів масиву

$$—(8)=(-7,9;1,0;1,1;-2,2;5,0;-1,1;2,0).$$

15) Підрахувати кількість елементів масиву

$$-(6)=(2,1;3,6;-6,3;4,1;2,2;-2,3),$$

значень яких більше 2,3.

16) Обчислити добуток елементів масиву

$$A(5)=(3,1;-7,8;6,2;-3,3;1,1),$$

які більші -5,4.

17) Обчислити суму значень від'ємних елементів масиву

$$X(8)=(-1,2;6,3;0,2;-0,7;1,1;2,3;-3,6;2,2).$$

18) Визначити номери додатнім елементів масиву

$$C(7)=(1,1;2,3;-6,4;0,0;2,1;2,3;1,2).$$

19) Обчислити добуток елементів масиву

$$A(5)=(1,3;6,3;2,4;-3,6;-2,5).$$

20) Визначити мінімальний елемент масиву

$$X(6)=(2,1;-3,6;-2,0;0,0;-6,3;1,0)$$

та номер цього елемента.

21) Надрукувати номер першого від'ємного елемента масиву
$$A(8)=(3,2;-6,3;2,0;-3,3;-6,6;-2,2;0;2,1).$$

22) Визначити номер максимального елемента масиву
$$C(6)=(2,3;7,9;12,3;-6,8;-22,3;0,0).$$

23) Вивести на друк номери від'ємних елементів масиву
$$D(7)=(2,2;-3,3;2,1;-3,0;-7,1;-5,1;0,0).$$

24) Знайти мінімальний елемент масиву
$$B(6)=(21,3;30,5;-6,8;0.3;-1,2;5,3).$$

25) Визначити максимальний по модулю елемент масиву
$$C(8)=(-3,6-5,3;2,1;0,1;-0,7;5,3;6,6;-2,2).$$

26) Визначити кількість невід'ємних елементів масиву
$$D(7)=(-2,3;2,3;0,0;3,2;6,0;-6,0;3,2).$$

27) Підрахувати кількість елементів в масиві
$$D(8)=(6,3;26;-3,6;2,1;0,0;6,6;-7,2;1,1).$$

які не перевищують по модулю число 5.

28) Обчислити середнє арифметичне елементів масиву
$$A(5)=(3,2;6,3;-3,3;2,3;5,5).$$

29) Знайти кількість додатнім елементів масиву
$$B(6)=(6,2;-3,2;0,0;3,3;2,2;-3,6).$$

30) Визначити, який номер має найменший елемент масиву
$$C(7)=(3,3;0,0;-3,3;-6,17;6,6;2,1).$$

Завдання 2

Варіанти:

1. Із матриці A(n,n) (n<=6) отримати нову матрицю B(n,n) шляхом ділення всіх елементів матриці A на її максимальний по модулю елемент.

2. В матриці A(6,8) необхідно поміняти рядок, який містить мінімальний елемент, на рядок що містить максимальний елемент. Вважати, що ці елементи єдині.

3. Із матриці A(m,n) (m<=5, n<6) отримати числа a1,....,am, де a1-значення першого по порядку додатного елемента i-го рядка.

4. Транспонувати матрицю A(m,n) (m<=4, n<6) і роздрукувати отриману матрицю.

5. Координати m векторів задані матрицею A(m,n) (m<=6, n<7) .

Необхідно обчислити довжини цих векторів, роздрукувати значення і серед цих знайти і вказати номер вектора мінімальної довжини.

6. Провести таке перетворення матриці A(m,n) (n<=4) при якому всі додатні елементи замінюються на суму відповідних індексів, а від'ємні - на добуток індексів.

7. Знайти мінімальний елемент матриці A(m,n) (m<=5) і вивести його на екран його. Елементи матриці, що лежать нижче головної діагоналі, замінити мінімальним елементом.

8. В матриці A(6,6) знищити 4-й рядок і роздрукувати отриману матрицю.

9. Провести таке перетворення матриці A(m,n) (m<=5, n<7), при якому останній стовпчик займе місце першого, а всі інші змістяться на один стовпчик вправо.

10. Координати n векторів задані матрицею A(m,n) (m<=5, n<=6). Обчислити довжини цих векторів, роздрукувати і нанести їх значення одновимірний масив. Серед елементів масиву знайти максимальний елемент та його номер.

11. В матриці A(3,7) знищити 5-й стовпчик і роздрукувати отриману матрицю.

12. Матриця A(m,n) (m<=4, n<=3) містить додатні та від'ємні елементи. Сформулювати з елементів даної матриці два масиви: B - що містить додатні елементи, та C - від'ємні. Підрахувати кількість елементів в даних масивах.

13. Цілочисельна матриця A(n,n) (n<=5). Знайти найменше із значень елементів стовпчика, який має максимальну суму модулів елементів. Вказати номер стовпчика.

14. Провести таке перетворення матриці A(m,n) (m<=7, n<=3), при якому останній рядок поміняється місцями з першим, передостанній з другим і т.д. Роздрукувати перетворену матрицю.

15. В матриці B(m,n) (m<=4, n<=6), всі елементи якої різні. В кожному рядку вибирається елемент з найменшим значенням, потім серед цих чисел вибирається найбільше. Вивести на екран знайдений елемент.

16. Піднести до квадрату всі непарні елементи матриці A(m,n) (m<=5, n<=4) і сформулювати із цих квадратів одновимірний масив.

17. У матриці A(m,n) (m<=5, n<=5) замінити елемент головної діагоналі елементами побічної.

18. У матриці A(m,n) (m<=5, n<=4) знайти суму елементів, які обрамляють дану матрицю і поміняти місцями мінімальний елемент лівої сторони на максимальний елемент правої.

19. Зробити таке перетворення матриці A(m,n) (m<=5, n<=5). Елементи, які стоять над побічною діагоналлю замінити систематичними їм відносно

побічної діагоналі;

20. У матриці A(m,n) (m<=5, n<=5) замінити елементи, які стоять над побічною діагоналлю, їм симетрично відносно цієї діагоналі.

21. У матриці A(m,n) (m<=5, n<=5) поміняти елементи діагоналі, яка прилягає до головної діагоналі згори, на елемент діагоналі, яка прилягає до головної діагоналі знизу.

22. У матриці A(m,n) (m<=5, n<=4) впорядкувати двох останніх рядків по спаданню.

23. Впорядкувати елементи матриці A(m,n) (m<=5, n<=4), які стоять вище головної діагоналі по зростанню і записати їх нижче головної діагоналі по рядках.

24. Впорядкувати елементи матриці A(m,n) (m<=5, n<=4) по спаданню і розмістити їх по рядках.

25. Провести таке перетворення матриці A(m,n) (n<=5) при якому всі додатні елементи замінюються на суму відповідних індексів, а від'ємні - на різницю індексів.

26. Зробити таке перетворення матриці A(m,n) (m<=5, n<=5): замінити елементи, які стоять над побічною діагоналлю, їм симетричними відносно головної діагоналі.

27. В матриці B(m,n) (m<=5, n<=5), всі елементи якої різні. В кожному рядку вибирається елемент з найбільшим значенням, потім серед цих чисел вибирається найменше. Роздрукувати знайдений елемент.

28. Провести таке перетворення матриці A(m,n) (m=5, n=6), при якому останній стовпчик займе місце першого, а всі інші змістяться на один стовпчик вправо

29. В матриці A(6,6) знищити 3-й рядок і роздрукувати отриману матрицю.

30. Провести таке перетворення матриці A(m,n) (m=8, n=3), при якому останній рядок поміняється місцями з першим, передостанній з другим і т.д. Роздрукувати перетворену матрицю.

Лабораторна робота №4
Розробка програм з рядковими змінними та функціями користувача

4.1. Мета роботи

Навчитися працювати з текстовими даними, Отримати знання і навички, необхідні для програмування на основі створення і використання користувацьких функцій, та навчитися використовувати їх на практиці в процесі розроблення програм мовою програмування С++

4.2. Теоретичні відомості

4.2.1. Символи

Змінні символьного типу оголошуються за допомогою ключового слова char і займають у пам'яті 1 байт. Тип char є цілочисельним типом і може задаватися зі знаком або без знаку. Спосіб інтерпретації змінних типу char може задаватися неявно або явно. Неявна форма типу char визначається опцією компілятора. В інтегрованому середовищі ця опція задається за допомогою меню Options/Compile/Code Generation. Явна форма визначається за допомогою модифікаторів типу signed або unsigned.

Приклади оголошень:

char c;

unsigned char t;

signed char v;

Значення змінної типу char визначає код одного із 256 символів кодової таблиці.

Якщо тип char розглядається як signed; то старший біт коду визначає знак. В цьому випадку діапазон значень типу char становить від - 128 до 123. Для типу unsigned char діапазон значень коду становить від 0 до 255. Ініціалізація змінних типу char може здійснюватися неявно або явно.

Неявно статичні та глобальні змінні типу char ініціалізуються значенням '\0'. Локальні змінні, які не є статичними, приймають невизначене значення.

Явна ініціалізація змінних типу char може здійснюватися при їх оголошенні або використанням операції присвоєння чи функцій вводу. Змінній типу char можна присвоїти числове або символьне значення. Символьна константа задається в апострофах явно або своїм вісімковим чи шістнадцятковим кодом, перед яким повинен йти символ \, наприклад:

char c1='A';

char c1='x41';

char c3,c4=0x41;

В усіх випадках змінні приймуть значення 0x41 (або десяткове 65), яке, в залежності від контексту використання, можна інтерпретувати як число, або символ з відповідним кодом.

4.2.2. Масиви символів

Масив символів - це послідовність елементів типу char, розміщених у неперервній області пам'яті. Приклад оголошення масиву символів, наприклад:

char buffer[10];

У масив символів можуть входити латинські букви, символи кирилиці, знаки пунктуації і керуючі символи. Керуючі символи задаються своїм мнемонічним позначенням або значенням ASCII-коду, перед яким повинен записуватися символ \.

Ініціалізація масиву символів може здійснюватися одним з наступних способів:

1. по змовчуванню глобальні та статичні масиви ініціалізуються символом '0';
2. явна посимвольна ініціалізація під час оголошення;
3. посимвольна ініціалізація за допомогою операції присвоєння або посимвольного вводу.

Якщо масив символів ініціалізується під час його оголошення, то кількість елементів може вказуватися явно, наприклад:

- char bufrer[10]={'T','u','r','b','o',' ','C'};

або неявно:

- char buffer[]={'T','u','r','b','o',' ','C'};

В першому прикладі задається масив з 10 елементів. Першим 7 з них будуть присвоєні значення, а наступні 3 приймуть значення '0', якщо масив є глобальним або статичним, і невизначені значення - в інших випадках. Кількість елементів списку ініціалізації не повинна перевищувати заданого значення.

В другому прикладі кількість елементів масиву визначається неявно по їх списку, В результаті буде створений масив з 7 символів, які приймуть значення із заданого списку ініціалізації.

Доступ до елементів масиву може бути здійснений за допомогою індекса або вказівника. Індекс масиву - це вираз цілого типу, який записується в квадратних дужках після імені масиву. Індекс першого елемента масиву дорівнює 0. Наприклад, для звертання до символу 'r' раніше визначеного масиву, необхідно записати buffer[2].

При доступі до символів за допомогою індекса необхідно пам'ятати, що компілятор не здійснює контроль його допустимого значення. Некоректне

використання індексу може призвести до звертання за межі масиву без видачі зауваження або повідомлення про помилку,

Враховуючи, що ім'я масиву є вказівник-константа на перший по порядку байт цього масиву (елемент buffer[0]), для доступу до - і-го елемента масиву символів можна використати операцію *, наприклад; *(buffer+i)

Посимвольна ініціалізація за допомогою операції присвоєння здійснюється шляхом звертання до окремих елементів масиву одним із приведених нижче способів:

buffer[1]='U';

**(buffer+2)='R';*

Поелементний ввід масиву символів організується за допомогою оператора циклу, наприклад:

```
for(i=0;i<10;i++)
getchar(buffer[i]);
```

Для вводу символів можна використати спеціально призначені бібліотечні функції getchar, getch, getche; або функцію форматованого вводу scar.

Масиви символів інакше називаються буферами символів. Для роботи з буферами символів призначені спеціальні функції, прототипи яких приведені у файлі mem.h.

4.2.3. Рядки символів

Рядок символів (string) на мові С - це масив символів, що закінчується символом '\0' (NULL), Оголошення рядків здійснюється так само, як масивів символів, наприклад::

```
char line[20];  /* оголошений рядок із 20 символів */
```

В дійсності в такий масив можна записати 19 символів, а 20-й символ є признаком кінця рядка '\0', під який обов'язково необхідно зарезервувати пам'ять. Кожен символ рядка займає в пам'яті 1 байт. Максимальна довжина рядка залежить від вибраної моделі пам'яті.

Початкова ініціалізація рядків символів може здійснюватися двома способами. При першому способі символи задаються як елементи масиву, наприклад::

```
charstr[6]= {'w','h','i','l','e','\0'};
```

При такому способі оголошення рядка символ '\0' повинен вказуватися явно в кінці масиву елементів. Якщо кількість перерахованих у фігурних дужках символів є меншою від вказаної розмірності, то рядок доповнюється справа до визначеної довжини символами '\0'.

Якщо символ '\0' в кінці списку ініціалізації не вказаний, то в пам'яті

буде сформований не рядок, а масив символів.

При другому способі ініціалізації рядок не розділяється на окремі символи, а задається в лапках, наприклад:

char str[] = "while";

При такому способі ініціалізації компілятор сам додасть символ '\0' в кінець рядка.

Рядок символів може бути пустим. Такий рядок складається тільки з одного символа '\0' і ініціалізується наступним чином:

char str[]="";

Звертання до символа масиву здійснюється по його порядковому номеру, наприклад:

char c, str[]="while";

c=str[2]; //c='i'

Ім'я рядка символів є вказівником на нульовий елемент (str ==&str[0]), тобто приймає константне значення, присвоєне йому на етапі компіляції. Тому не дозволяється модифікація цього значення, наприклад операція str++ є недопустимою.

Для оголошення рядка символів можна використовувати вказівник, наприклад:

*char *str1 = "while";*

*або char *str1;*

str1="while";

Змінна str1 містить адресу рядка "while". На відміну від попереднього опису рядка як масиву символів, допускається її модифікація, наприклад операція str1++ допустима і нове значення змінної str1 вказує на наступний елемент.

Вираз *(srt1+i) забезпечує звертання до і-го елементу рядка символів, наприклад:

c=(str1+2) /*c='i' */*

4.2.4. Масиви символьних рядків

Рядки символів можна об'єднувати в масиви довільної розмірності. На практиці найбільш поширена робота з двомірними масивами символів (одномірними масивами рядків символів).

В залежності від опису рядки символів зберігаються в неперервній області пам'яті як двомірні символьні масиви або в різних областях пам'яті у вигляді окремих рядків. Перший спосіб опису:

charmasstr[4][10]={"green", "red", "white", "blue"};

визначає прямокутний масив символів. Всі чотири рядки матимуть однакову довжину. Короткі рядки доповнюються справа символами '\0' до

вказаної довжини (до 10 символів).

Опускаючи другий індекс, будемо мати адреси кожного з рядків, наприклад, запис masstr[2] визначає вказівник на рядок "white", а запис *(masstr[2]+j) визначає j-й символ цього рядка (j = 0,...,9).

Для доступу до символів можна використовувати також спосіб, характерний для числових масивів, наприклад, запис masstr[0][1] визначає символ 'r' рядка "green".

Другий спосіб опису:

char *prtstr[4]={"green", "red", "white", "blue"};

визначає масив із 4-х вказівників на рядки різної довжини. Значення рядків, можливо, будуть зберігатися не підряд, а в різних областях пам'яті.

Для доступу до j-го символу і-го рядка можна скористатися наступним виразом; *(ptrstr[i]+j).

4.2.5. Організація стандартного виводу та вводу символьних даних. Форматний вивід даних

Здійснюється за допомогою функції printf, яка має змінну кількість аргументів. Прототип функції приведений вище у таблиці.

Звертання до функції:

printf ("форматний рядок", арг1,арг2,...);

Форматний (керуючий) рядок використовується для того, щоб задати кількість та типи аргументів і може включати в себе:

1. звичайні символи, які виводяться на екран дисплея;
2. специфікації перетворення даних, кожна з яких викликає на екран значення наступного аргумента із списку,
3. керуючі символьні константи:
   - \a - викликає звуковий сигнал;
   - \n - перехід на новий рядок;
   - \b - повернення на позицію вліво;
   - \r - перехід на початок біжучого рядка;
   - \f - перехід на нову сторінку;
   - \t - горизонтальна табуляція;
   - \v - вертикальна табуляція;
   - \ddd - вісімковий код символу;
   - \’ - апостроф;
   - \xddd - шістнадцятковий код символу;
   - \" - подвійні лапки;
   - \0 - нульовий символ (пусто);
   - \\ - зворотня коса риска;

Специфікація перетворення має наступний формат:

%[вирівнювання][ширина][точність]символ перетворення

Квадратні дужки не є символами специфікації, а лише вказують, що дане поле може бути опущене. Специфікація перетворення починається знаком % і закінчується символом перетворення (форматом), між якими можуть бути:

1. Знак "-" (мінус), який показує, що перетворений параметр повинен бути вирівняний вліво у своєму полі (по змовчуванню вирівнюється вправо);
2. Рядок цифр - мінімальна ширина поля. Якщо значення змінної перевищує ширину поля, то виводиться стільки символів, скільки потрібно;
3. Капка;
4. Рядок цифр - максимальне число символів, які необхідно вивести для типу char. Аргументами функції printf можуть бути змінні, константи, вирази, виклики функцій. Значення аргументів повинні відповідати заданій специфікації.

Вивід символів

Специфікація перетворення:

%[-][ширина]c

Вивід символів рядка здійснюється доти, поки не зустрінеться символ '\0', або до вказаної точності. Якщо довжина рядка більша від заданої точності, то залишок рядка відкидається.

Здійснюється за допомогою функції scanf, яка може мати змінну кількість аргументів. Функція scanf призначена в основному для зчитування сукупності даних різних типів. Формат функції:

scanf("форматний рядок", арг1, арг2, ...);

Характерною особливістю даної функції є те, що її аргументи повинні бути вказівниками на значення. Для кожного аргумента у форматному рядку задається своя специфікація перетворення.

Ввід символа

Специфікація перетворення; %[*][ширина]c

Ширина визначає число символів, які повинні бути прочитані із вхідного потоку і присвоєні масиву символів. Якщо ширина опущена, то вводиться один символ. По даній специфікації можна вводити пусті символи.

Приклад:

```
char a[5], b; /* Вхідний потік: 1234567890 */
scanf("%5c", a);
a[4]='\0';     /* Результат a = 1234\0 */
```

scanf("%c", &b); / Результат b = 5 */*

Ввід рядка символів

Специфікація перетворення: %[*][ширина]s

Ширина задає максимальну довжину ввідного рядка. Рядки у вхідному потоці повинні розділятися пустими символами. Ведучі пусті символи ігноруються. Зчитування відбувається до першого пустого символа (пропуска, табуляції переходу на новий рядок), або до закінчення вказаної ширини. В пам'яті у кінець рядка додається символ '\0' для заповнення оголошеної довжини.

Приклад:

char a[5], b[6]; / Вхідний потік; 1234567890 */*
scanf("%3s", a); / Результат a = 123\0\0 */*
scanf("%5s", b); / Результат b = 45678\0 */*

Функції безформатного виводу

Вивід символів

Функція putchar використовується у програмі для відображення символу на екрані відеотермінала.

Звертання до функції: putchar(символ);

Після виводу символа курсор залишається в рядку виводу. Якщо вивідний рядок заповнений, то при виводі наступного символу відбувається перехід на початок нового рядка. Приклади виводу символів:

char ch='b';
putchar('a'); / a */*
putchar('\n'); / перехід на новий рядок */*
putchar('\007'); / звуковий сигнал */*
*putchar(ch); /*b */*
putchar(getchar()); / вивід введеного символу */*

Вивід рядка символів

Для виводу рядка символів використовується функція puts. Звертання до функції: puts(вказівник на рядок);

Функція puts зупиняє вивід символів, якщо зустріне символ '\0'. Вивід цього символу не здійснюється. Рядок символів, що виводиться функцією puts завжди починається з нового рядка на екрані. Приклади використання функції puts:

char str1 [] = "abcdefgh";
*char *str2 = "1234567890";*
puts("Вивід повідомлення");
*puts(str1); /*abcdefgh*/*
puts(str2); / 1234567890 */*

*puts(&str1[4]); /*efgh*/*
puts(str2+6); / 7890 */*

4.2.6. Функції безформатного вводу

Ввід символів

Для вводу символів використовуються функції getchar, getch та getche. Прототип функції getchar знаходиться у файлі stdio.h, а прототипи функцій getch та getche - у файлі conio.h .

Функція getchar призначена для зчитування символа з клавіатури, відображення його на екрані та передачі у програму. Функція реалізує буферизований ввід - передача символу у програму відбувається після його набору на клавіатурі та натискання клавіши <Enter>.

Звертання до функції getchar; змінна = getchar();

Змінна, якій присвоюється значення функції, повинна мати тип char. Наприклад:

char ch;
ch=getchar();
while((ch=getchar())!='') {/* тіло циклу */}*

Функція getch реалізує небуферизований ввід - передача символу у програму відбувається зразу після його набору на клавіатурі без натискання клавіші <Enter>. Характерною особливістю даної функції є також те, що введений символ не відображається на екрані.

Звертання до функції getch:

змінна = getch();

Функція getche теж реалізує небуферизований ввід символу, але з відображенням його на екрані. Звертання до функції getche:

змінна = getche();

Якщо у вхідному потоці функції getchar(), getch() або getche() зустрінуть код кінця файлу, то вони повертають признак EOF. Оголошення змінної EOF приведено у стандарному файлі stdio.h. Ознака кінця файлу імітується одночасним натисканням двох клавіш "Ctrl+Z" (при введенні символа з клавіатури). Це можна використати як умову виходу з циклу, якщо введений символ використовується в тілі циклу, наприклад:

char ch;
while((ch=getchar())!=EOF)
{ }

Ввід рядків символів

Для вводу рядка символів використовується функція gets. Дана функція читає символи з вхідного потоку доти, поки не зустрінеться символ нового

рядка '\n', який формується при натисканні клавіші <Enter>. Символ '\n' не включається в кінець рядка, а замість нього автоматично формується символ '\0', на що необхідно зарезервувати додатковий байт пам'яті.

Перед використанням функції gets необхідно виділити пам'ять для рядка символів. Звертання до функції gets:

gets(вказівник на рядок);

Наприклад:

char name[81];

gets(name);

Наступний приклад демонструє неправильне використання функції gets:

*static char *name;*

gets(name);

Хоча аргумент функції вказаний правильно, як вказівник на символьний тип, але не зарезервована пам'ять під рядок символів. Розміщення рядка в пам'яті по даній адресі може привести до затирання іншої інформації і, в результаті, до непередбачених наслідків.

На відміну від функції scanf, функція gets дає можливість вводити пусті символи, наприклад, пропуски. Функція gets повертає вказівник на введений рядок символів. При неправильному зчитуванні даних або при зустрічі у вхідному потоці коду кінця файлу функція gets повертає вказівник NULL, наприклад:

while(gets(name)!==NULL) {}

4.2.8. Обробка рядків символів

При обробці рядків символів найчастіше необхідно виконувати операції визначення довжини рядка, копіювання, конкатенації та порівняння рядків. Для роботи з рядками призначені функції, прототипи яких приведені у файлі string.h.

Довжина рядка визначається за допомогою функції strlen(), яка має наступний прототип:

*unsigned strlen(char *str);*

Функція повертає кількість символів рядка до нуль-символа завершення рядка '\0'. Приклад:

char str[20]="Рядок символів";

int k;

k=strlen(str);

Змінна k прийме значення, рівне кількості символів рядка str, тобто k=14.

При виконанні операції копіювання необхідно правильно визначити рядок-приймач символьних даних. Цей рядок можна визначити як буфер

символів у статичній або динамічній пам'яті.

У статичній пам'яті рядок оголошується як масив символів, наприклад; char buf[20]. Динамічна пам'ять під рядок виділяється за допомогою функцій malloc() або calloc() :

*char *buf;*

*buf=(char *)malloc(20);*

В обох випадках необхідно передбачити виділення пам'яті під символ кінця рядка '\0'. Копіювання рядків здійснюється за допомогою бібліотечної функції strcpy(), прототип якої має вигляд:

*char *strcpy(char *str1, char *str2);*

Ця функція копіює рядок str2 у рядок str1. Обидва рядки задаються своїми вказівниками. Функція повертає вказівник на рядок-приймач str1. Довжина рядка str1 повинна бути достатньою для зберігання рядка str2, включаючи нуль-символ завершення рядка. Наприклад;

char str1 [15];

char str2[]='GCC MinGW";

strcpy (str1, str2);

Об'єднання двох рядків здійснює функція strcat():

*char *strcat(char *str1, char *str2);*

Рядок str2 буде дописаний в кінець рядка str1. Довжина рядка str1 повинна бути достатньою для розміщення результату об'єднання, включаючи символ завершення рядка '\0'. Приклад;

char str1 [20]="Турбо ";

*char *str2="Ci";*

strcat(str1,str2);

Результатом роботи функції strcat() буде рядок str1, який прийме значення символьної константи "Турбо СІ"

Для порівняння рядків використовується функція strcmp(), яка має наступний прототип:

*int strcmp(char *str1, char *str2);*

Ця функція порівнює рядки str1 і str2 і повертає ціле число менше 0, якщо str1<str2; рівне 0, якщо str1=str2; більше 0, якщо str1>str2. Порівняння рядків здійснюється посимвольно зліва направо до першого неспівпадання кодів символів. При порівнянні необхідно пам'ятати, що в таблиці ASCII коди малих літер є більші від кодів великих літер. Приклад:

char str1[]="Turbo C" char str2[]="TURBO C++"

if(strcmp(str1,str2)) puts("Перший рядок більший від другого"), Інші функції обробки рядків символів приведені у таблиці 1.

4.2.9. Функції мові С

Функції - це будівельні блоки мови С, самостійні одиниці програми, спроектовані для рішення конкретних задач, що звичайно повторюються декілька разів.

Основна форма опису (definition) функції має вигляд:

тип <ім'я функції>(список параметрів)
{
тіло функції
}

Тип функції визначає тип значення, що повертає функція за допомогою оператора return. Якщо тип не зазначений, то по умовчанню передбачається, що функція повертає ціле значення (типу int). Список параметрів складається з переліку типів і імен параметрів, розділених комами. Функція може не мати параметрів, але круглі дужки необхідні в будь-якому випадку.

У списку параметрів для кожного параметра повинний бути зазначений тип. Приклад правильного списку параметрів:

t(int x, int y, float z)

Приклад неправильного списку параметрів:

f(int x, y, float z)

Приведемо приклад функції, що реалізує зведення числа a в натуральний степінь b:

```
float step(float a, int b)
 {
  float i;
  if(a<0) return (-1); /* основа негативна */ a=1;
  for(i=b;i;i--) a*=a;
  return a;
 }
```

Ця функція повертає значення -1, якщо основа негативна, і an - якщо основа невід'ємна.

Оператор *return* має два використання.

По-перше, цей оператор викликає негайний вихід із поточної функції і повернення в програму, що викликає.

По-друге, цей оператор може використовуватися для повернення значення функції.

Відразу слід зазначити, що в тілі функції може бути декілька операторів return, але може і не бути жодного. У цьому випадку повернення в програму , що викликає , відбувається після виконання останнього оператора тіла функції.

Інший приклад - функція для знаходження найбільшого з двох чисел:

```
max(int a, int b)
 {
  int m;
  if(a>b) m=a;
  else m=b;
  return m;
  }
```

Можливо також написати цю функцію без використання додаткової змінної:

```
max(int a, int b)
 {
  if(a>b) return a;
  else return b;
 }
Можна ще коротше:
max(int a, int b)
 {
  if(a>b) return a;
  return b;
 }
А можна і так:
max(int a, int b)
 {
  return (a>b)? a: b;
 }
```

Якщо функція повинна повертати значення, але не робить цього, компілятор видає попередження. Всі функції, що повертають значення, можуть використовуватися у виразах мови С, але вони не можуть використовуватися в лівій частині оператора присвоювання, за винятком тих випадків, що коли повертається значення покажчика.

Використання функцій, що повертають покажчики, має деякі особливості. Покажчики не є ні типом ціле (int), ні типом беззнакове ціле (unsigned int). Їхніми значеннями є адреси пам'яті даних визначеного типу. Відповідно повинна бути описана і функція. Розглянемо приклад опису функції, що повертає покажчик на тип char. Ця функція знаходить у рядку перший пробіл і повертає його адресу.

```
char* find(char* string)
 {
  int i=0;
  while (siring[i] != ' ')&&(string[i] != '\0') i++;
  if(s[i]) return &s[i]; /* повертає адреса першого пробілу */
  else return NULL; /* повернення нульового покажчика */
 }
```

Коли функція не повертає ніякого значення, вона повинна бути описана як функція типу void (порожня).

Ви не зобов'язані описувати функцію типу void, тоді вона по умовчанню буде мати тип int і не повертати ніякого значення. Це викликає попереджуюче повідомлення компілятора, але не буде перешкодою для компіляції. Однак оголошення типу що повертається значення функції є гарним правилом.

4.2.10. Прототипи функцій

Особливістю стандарту ANSI мови С є те, що для створення правильного машинного коду функції йому необхідно повідомити до першого виклику тип що повертається результату, а також кількість і типи аргументів. Для цієї цілі в С використовується поняття прототипу функції. Прототип функції задається в такий спосіб: тип <ім'я функції>(список параметрів);

Використання прототипу функції є оголошенням функції (declaration). Частіше усього прототип функції цілком збігається з заголовком в описі функції, хоча це і не завжди так. При оголошенні функції компілятору важливо знати ім'я функції, кількість і тип параметрів і тип значення що повертається. При цьому імена формальних параметрів функції не грають ніякої ролі й ігноруються компілятором. Тому прототип функції може виглядати або так:

```
int func(int a, float b, char* c);
```

або так:

```
int func(int, float, char*);
```

Два цих оголошення абсолютно рівноправні.

Розглянемо приклад.

```
# include <stdio. h>
```

```
float sqr(float a);   /* це прототип функції, оголошення функції */

int main()
 {
  float b;
  b=5.2;
  printf("Квадрат числа %f дорівнює %f", b, sqr(b));
return 0;
 }

float sqr(float a)    /* Це опис функції */
 {
  return a*a;
 }
```

Наступні два приклади використання функцій викликають повідомлення про помилку при компіляції. У першому прикладі перешкодою для компіляції буде невідповідність значення що повертається оголошеному типу функції. Мови С і С++ автоматично перетворять дані до іншого типу, але тільки тоді, коли це можливо. Цілий тип не може бути автоматично перетвореним в покажчик на ціле.

```
#include <stdio.h>    /* Приклад неправильний */
int *sqr(int *i) /* Прототип функції */
main( )
{
int i;
sqr(&x);
}
int *sqr(int *i) /* Оголошення функції */
{
return *i=(*i)*(*i);
}
```

```
#include <stdio.h>
/*Приклад неправильний */
int sqr(int *i) /* Прототип функції */
main()
{
int x=10:
sqr(&x, 10); /* Невідповідність кількості аргументів */
```

```
}
int sqr(int *i)
{
*i=(*i)*(*i);
}
```

Звернемо увагу на те, що якщо ми виправимо ці програми, то функція буде повертати квадрат числа і не через значення функції, а через параметр функції.

Якщо функція не має аргументів, то при оголошенні прототипу такої функції слід замість аргументів писати ключове слово void. Це повинно стосуватися і функції main().Її оголошення повинно мати вигляд void main(void) або main(void).

```
#include <stdio.h>
void line_ (void); /* прототип функції /
main (void)
 {
    line_();
 }
void line_(void)
 {
int i:
for(i=0;i<80;i++) printf("-“);
 }
```

Заголовні файли мови С містять прототипи стандартних функцій, що відносяться до цього заголовного файла. Прикладами таких заголовних файлів є файли stdio.h, string.h, conio.h і ін.

4.2.11. Область дії й область видимості

Область дії (scope rules) перемінної - це правила, що встановлюють, які дані доступні з даного місця програми.

У мові С кожна функція - це окремий блок програми. Потрапити в тіло функції не можна інакше, як через виклик даної функції. Зокрема, не можна оператором локального переходу goto перейти в середину іншої функції.

З погляду області дії перемінних розрізняють три типи перемінних: глобальні, локальні і формальні параметри. Правила області дії визначають, де кожна з них може застосовуватися.

Локальні перемінні - це перемінні, оголошені усередині блока, зокрема усередині функції. Мова С підтримує просте правило: перемінна може бути оголошена усередині будь-якого блока програми. Локальна перемінна доступна усередині блока, у якому вона оголошена. Згадаємо що блок відчиняється фігурною дужкою і закривається фігурною дужкою. Область дії локальної перемінної - блок.

Локальна перемінна існує поки виконується блок, у котрому ця перемінна оголошена. При виході з блока ця перемінна (і її значення) втрачається.

```
#include < stdio.h>
void f(void);
main(void)
 {
  int i;
  i=1;
  f( );
  printf("В функції main значення і дорівнює %d\n", i);
 }
void f(void)
 {
  int i;
  i=10;
 printf("В функції f() значення і дорівнює %d\n", i);
 }
```

Приклад показує, що при виклику функції значення перемінної і, оголошеної в main( ), не змінилося.

Формальні параметри - це змінні, оголошені при описі функцій як її аргументи. Функції можуть мати деяку кількість параметрів, що використовуються при виклику функцій для передачі значень у тіло функції. Формальні параметри можуть використовуватися в тілі функції так само, як локальні перемінні, якими вони по суті діла і є. Область дії формальних параметрів - блок, що є тілом функції.

Глобальні змінні - це змінні, оголошені поза функціями. На відміну від локальних змінних глобальні змінні можуть бути використані в будь-якому місці програми, але перед їхнім першим використанням вони повинні бути оголошені. Область дії глобальної перемінної - уся програма.

Використання глобальних перемінних має свої недоліки:

• вони займають пам'ять протягом усього часу роботи програми;

• використання глобальних змінні робить функції менше загальними й

утруднює їхнє використання в інших програмах;

• використання зовнішніх перемінних зумовлює появу помилок через побічні явища. Ці помилки, як правило, важко відшукати.

4.2.12. Приклади роботи з рядками символів

1. Для заданого масиву рядків символів підрахувати кількість голосних букв в кожному рядку.

```
#include <stdio.h>
#include <string.h>
#define KMAX 5
int main()
{ inti,k:
char voc[]="aAeEiIoOuUyY";
char *masstr[]={"typedef", "while", "for", "unsigned", "float"};
char *ptr;
for(i=0;i<KMAX;i++)
{ k=0:
ptr=masstr[i];
while(ptr!=NULL) {ptr=strpbrk(ptr,voc);
if(ptr) {k++; ptr++;}
}
fprintf(stdprn."k=%d\n",k);
}
return 0;
}
```

2. Впорядкувати в лексикографічному порядку елементи масиву рядків символів, введених з клавіатури.

```
#include <stdio.h>
#inctude <io.h>
#include <string.h>
#define n 5
#define m 10
void sswap(char ,*char *);
int main()
{
int i,j;
char masstr[n][m];
```

```
char *ptr;
clrscr();
for(i=0;t<n;i++)
   gets(masstr[i]);
for(i=0;i<n-1;i++)
   for(j=i+1:j<n;j++)
      if(strcmp(masstr[i],masstr[j])>0) sswap(masstr[i],masstr[j]);
for(i=0;i<n:i++)
   puts(masstr[i]);
return 0;
}
void sswap(char *a, char *b)
{
char s[m];
strcpy(s.a);
strcpy(a,b);
strcpy(b,s);
}
```

Таблиця 4.1. Функції для роботи з рядками _#include <string.h>

Функція	**Прототип**	**Дія**
atof	double atof(char *str);	Перетворює рядок str в дійсне число подвійної точності. Перетворення здійснюється до першого недопустимого символа або до символа '\0'. Якщо не може перетворити, то повертає 0
atoi	int atoi(char*str);	Перетворює рядок str в десяткове ціле. Якщо число перевищує діапазон int, то повертає 2 молодші байти. Якщо не може перетворити, то повертає 0
atol	long atol(char*str);	Перетворює рядок str в довге десяткове ціле

ecvt	char *ecvt(double v, int dig, int *dec, int *sign);	Перетворює дійсне v у рядок: dig - кількість цифр числа, що будуть перетворені в рядок, dec - позиція десяткової крапки від початку рядка (якщо dec<=0, то позиція десяткової крапки знаходиться зліва від числа), sign є {0,1} - знак числа. Символ '\0' додається. Повертає вказівник на рядок
fcvt	char *fcvt(double v, int dig, int*dec, int*sign);	Те ж саме що й ecvt, тільки dig - кількість цифр після крапки
gcvt	char *gcvt(double v, int dig, char *buf);	Перетворює дійсне v у рядок. На відміну від ecvt() та fcvt() розміщає рядок у попередньо оголошений буфер buf. dig - це кількість символів рядка. Результуючий рядок містить представ-лення числа зфіксованою або плаваючою крапкою в залежності від того, чи може число розміститися в dig позиціях
itoa	char *itoa(int v, char *str, int baz);	Перетворює ціле v в рядок str у системі числення baz (2<=baz<=36). Повертає вказівник на рядок
ltoa	char *itoa(long v, char *str, int baz);	Перетворює довге ціле v в рядок символів str
strcat	char *strcat(char *sp, char *si);	Приписує рядок si до рядка sp
strch	char *strchr(char *str, char c);	Знаходить в рядку str перше входження символа c.
strcmp	int strcmp(char *str1, char *str2);	Порівнює рядки str1 і str2. Результат: <0, якщо str1<str2; =0, якщо str1=str2; >0, якщо str1>str2
strcmpi	int strcmpi(char *str1. char *str2);	Порівнює рядки str1 і str2 без врахування регістру для буквенних символів. Повертає те ж саме значення що й strcmp.
strcpy	char *strcpy(char *sp,char *si);	Копіює рядок si в рядок sp
strcpy	char *strcpy(char *sp, char *si);	Копіює рядок si в рядок sp
strcspn	int strcspn(char *str1, char *str2);	Визначає довжину першого сегменту рядка str1, що містить символи, які не входять в множину символів рядка str2

strlen	unsigned strlen(char *str);	Обчислює довжину рядка str
striwr	char *strlwr(char *str);	Перетворює букви верхнього регістра в рядку в букви нижнього регістра
strncat	char *strncat(char *sp, char *si, int kol);	Приписує kol символів рядка si до рядка sp
trncmp	int strncmp(char *str1, char *str2, int kol);	Порівнює kol перших символів рядків str1 та str2. Результат аналогічний функції strcmp
strncmpi	int strncmpi(char *str1, char *str2, int kol);	Порівнює kol перших символів рядків str1 та str2 без врахування регістру буквенних символів. Результат аналогічний функції strcmp
strncpy	char *strncpy(char *sp, char *sp. int kol);	Копіює kol символів рядка si в рядок sp
strpbrk	char *strpbrk(char *str1, char *str2);	Знаходить в рядку str1 перше входження довільного символа із множини символів рядка str2
strrchr	char *strrchr(char *str, char c);	Знаходить в рядку str останнє входження символа c
strset	char*strset(char*str, intch):	Записує символ ch у всі позиції рядка str. Повертає вказівник на str
Strnset	char *strset(char *str, int ch, zise t n);	Записує символ ch у перші n позиції рядка str. Повертає вказівник на str. Символ '\0' не затирається, якщо n > strien(str)
Strspn	int strspn(char *str1, char *str2);	Знаходить довжину першого сегменту рядка str1, що містить символи із множини символів, що входять в рядок str2
Strstr	char *strstr(char *str1, char *str2);	Повертає вказівник на елемент рядка str1, який є початком підрядка str2, і NULL, якщо str2 не входить в str1
Strupr	char *strupr(char *str);	Перетворює букви нижнього регістра рядка str у верхній
Ultoa	char *ultoa(unsigned long v, char *ctr, int baz); n+-	Перетворює беззнакове довге ціле v в рядок символів

Таблиця 4.2. Функції перевірки та перетворення символів #include <ctype.h>

Функція	**Прототип**	**Дія**
Isalnum	int isalnum(int c);	Дає відмінне від 0 значення, якщо с - буква (A-Z,a-z) або цифра (0-9), і 0 - в іншому випадку
Isalpha	int isalpha(int c);	Дає відмінне від 0 значення, якщо с - буква (A-Z,a-z), і 0 - в інших випадках
Isascii	int isascii(int c);	Дає відмінне від 0 значення, якщо код символа с від 0 до 127, і 0 - в інших випадках
Iscntrl	int iscntrl(int c);	Дає відмінне від 0 значення, якщо с - управляючий символ (Ox7F або 0x00-0x1 F), і 0 - в інших випадках
Isdigit	int isdigit(int c);	Дає відмінне від 0 значення, якщо с - цифра (0-9), і 0 - в інших випадках
Isgraph	int isgraph(int c);	Дає відмінне від 0 значення, якщо с - символ, що має графічне позначення (0x21-Ox7E), і 0 - в інших випадках
Slower	int islower(int c);	Дає відмінне від 0 значення, якщо с - символ нижнього регістру, і 0 - в інших випадках
ispnnt	int isprint(int c);	Дає відмінне від 0 значення, якщо с - друкований символ (0x20-Ox7E). і 0 - в інших випадках
Ispunct	int ispunct(intc);	Дає відмінне від 0 значення, якщо с - управляючий символ
Toupper	int toupper(int c);	Перетворює букву с до верхнього регістру

Таблиця 4.3. Функція для роботи з буферами (масивами символів) #include<mem.h> або #include<string.h>

Функція	**Прототип**	**Дія**
memcpy	void *memcpy (void *dest, void *src, size_t n);	Копіює блок n байт з src в dest. Буфери не повинні перекриватися. Повертає вказівник dest.

memccpy	void *memccpy (void *dest, void *src, int c, size_t n);	Копіює блок n байт з src в dest. Буфери не повинні перекриватися. Копіювання продовжується до тих пір, поки: 1. не зустрінеться символ c, який теж копіюється в dest. Повертає вказівник на наступний після символа c байт; 2. поки не скопіюється n байт. Повертає вказівник NULL
memmove	void *memmove (void *dest, void *src, size_t n);	Копіює блок n байт з src в dest. Буфери можуть перекриватися. Повертає вказівник dest.
movmem	void *moymem (void *src, void *dest, unsigned n);	Копіює блок n байт з src в dest. Буфери можуть перекриватися. Повертає вказівник dest.
movedata	void movedata (unsigned srcseg, unsigned srcoff, unsigned destseg, unsigned destoff, size_t n);	Копіює n байт з srcseg: srcoff в destseh: destoff
memcmp	int memcmp (void *s1, void *s2, size_t n);	Порівнює n перших байтів двох буферів s1 та s2 в лексикографічному порядку. Повертає значення: <0, якщо s1<s2; ==0, якщо s1==s2; >0, якщо s1>s2
memicmp	int memicmp (void *s1, void *s2, size_t n);	Теж саме що й memcmp, але без урахування регістру буквенних символів.
memchr	void *memchr (void *s, int c, size_t n);	Шукає символ c в перших n байтах буфера s. Повертає вказівник на символ c. Якщо символ не знайдений, то повертає NULL

4.3. Програма роботи

4.3.1. Запустити середовище Dev C++.

4.3.2. Скласти алгоритм програми до завдання 1 згідно свого варіанту.

4.3.3. Написати програму до завдання 1 згідно свого варіанту.

4.3.4. Скласти алгоритм програми для обробки текстового масиву (додаток 2) згідно свого варіанту.

4.3.5. Скласти програму для обробки текстового масиву згідно свого

варіанту.

Вимоги до програм

- вхідні дані ввести оператором введення (завдання1, 2);
- на друк вивести результати роботи програм згідно завдання.

4.4. Обладнання та програмне забезпечення

4.4.1. Персональний комп'ютер.

4.4.2. Програмне забезпечення: IDE Dev-C++

4.5. Контрольні запитання

4.5.1. Яким чином оголосити змінну символьного типу?

4.5.2. Який специфікатор формату використовується для введення та виведення символів?

4.5.3. Що таке масиви символів? Яким чином їх оголосити?

4.5.4. Як звернутися до певного елемента у масиві символів?

4.4.5. Що таке рядок символів? Яким чином його оголосити?

4.5.7.Що таке масив символьних рядків? Як його оголосити?

4.5.8.Які ви знаєте функції безформатного вводу-виводу рядків символів?

Завдання 1.

Результати екзаменаційної сесії студентів 1-го курсу подані у вигляді наступної таблиці

	Прізвище	Інформатика	Вища матем.	Фізика	Програмування
1.	Іванчук С.О.	4	3	3	4
2.	Панченко І.А.	5	4	4	5
3.	Заєць О.М.	3	4	4	4
4.	Вельбицький П.О.	4	3	3	3
5.	Сидоренко В.Р.	2	3	3	2
6.	Кравченко З.І.	3	5	4	5
7.	Якубів Р.Н.	5	4	4	3
8.	Зоренко П.М.	4	2	3	3
9.	Берестяк Г.С.	4	5	5	5
10.	Дячик Н.С.	5	5	4	4

Варіанти:

1. Надрукувати таблицю, що містить номери, прізвища та кількість “5”, “4”, “3”,”2” у кожного студента групи, а також підрахувати загальну кількість “5”, “4”, “3”, “2” в групі.
2. Надрукувати таблицю, що містить номери, прізвища, оцінки та середній бал тих студентів групи, середній бал яких більше 4, а також підрахувати кількість таких студентів у групі.
3. Надрукувати таблицю, що містить номери, прізвища та оцінки студентів, які мають хоча б одну “3”, а також підрахувати кількість таких студентів у групі.
4. Надрукувати таблицю, що містить номери, прізвища та оцінки студентів, які не мають жодної “5”. Підрахувати кількість таких студентів.
5. Надрукувати таблицю, що містить номери, прізвища та оцінки кожного студента, а в кінці вказати середній бал групи з кожної дисципліни.
6. Надрукувати таблицю, що містить номери, прізвища, оцінки та середній бал кожного студента групи.
7. Надрукувати таблицю, що містить номери, прізвища та оцінку студентів з вищої математики, а також підрахувати середній бал групи з цього предмета.
8. Надрукувати таблицю, що містить прізвища та оцінки тих студентів, які мають найбільший та найменший середній бал у групі.
9. Надрукувати таблицю, що містить номери, прізвища, оцінки та середній бал студентів групи, середній бал яких менше 4.
10. Надрукувати таблицю, що містить номери, прізвища, оцінки студентів, які мають тільки добрі та відмінні оцінки.
11. Надрукувати таблицю, що містить номери, прізвища, оцінки та кількість “3” в оцінках кожного студента.
12. Надрукувати таблицю, що містить номери, прізвища та оцінки тих студентів, які отримали з інформатики добрі та відмінні оцінки, а також підрахувати кількість таких студентів.
13. Надрукувати таблицю, що містить номери, прізвища та оцінки тих студентів, які отримали з вищої математики задовільну або незадовільну оцінку, а також підрахувати кількість таких студентів.
14. Надрукувати таблицю, що містить номери, прізвища та екзаменаційні оцінки студентів. В кінці вказати дисципліну, середній бал якої максимальний.
15. Надрукувати таблицю, що містить номери, прізвища та оцінки студентів, які отримали хоча б одну незадовільну оцінку.
16. Надрукувати кількість “2”, “3”, “4”, “5” з кожної дисципліни.

17. Надрукувати таблицю, що містить номери, прізвища і кількість “2”, “3”, “4”, “5” в оцінках кожного студента.
18. Надрукувати таблицю, що містить номери, прізвища і оцінки студентів з предметів “ Вища математика” і “ Інформатика”.
19. Надрукувати таблицю, яка містить середні екзаменаційні бали студента по кожному предмету.
20. Надрукувати таблицю, що містить номери, прізвища, оцінки та кількість позитивних оцінок кожного студента.
21. Надрукувати таблицю, що містить номери, прізвища та оцінки студентів, які мають позитивні оцінки з інформатики. Надрукувати кількість таких студентів.
22. Надрукувати таблицю, що містить номери, прізвища та оцінки студентів, які мають позитивні оцінки з вищої математики. Надрукувати кількість таких студентів.
23. Надрукувати таблицю, що містить номери, прізвища та оцінки студентів, які мають оцінки “добре” та “відмінно” з інформатики. Надрукувати кількість таких студентів.
24. Надрукувати таблицю, що містить номери, прізвища та оцінки студентів, які мають позитивні оцінки з фізики. Надрукувати кількість таких студентів.
25. Надрукувати таблицю, що містить номери, прізвища та оцінки студентів, які мають оцінки “добре” та “відмінно” з програмування. Надрукувати кількість таких студентів.
26. Надрукувати таблицю, що містить прізвища та оцінки тих студентів, які мають найбільший та найменший середній бал у групі.
27. Надрукувати таблицю, що містить номери, прізвища, оцінки та середній бал студентів групи, середній бал яких менше 3.
28. Надрукувати таблицю, що містить номери, прізвища, оцінки студентів, які мають тільки відмінні оцінки.
29. Надрукувати таблицю, що містить номери, прізвища, оцінки та кількість “2” в оцінках кожного студента.
30. Надрукувати таблицю, що містить номери, прізвища та оцінки тих студентів, які отримали з програмування добрі та відмінні оцінки, а також підрахувати кількість таких студентів.

Завдання 2

Варіанти:

1. Дано текстовий масив A(10). Знайти і надрукувати елементи найменшої довжини. Вивести на друк даний елемент, його порядковий номер і довжину (кількість символів).
2. В текстовому масиві B(12) відшукати елемент з найбільшою

довжиною, вивести його на друк разом з номером і довжиною.

3. В текстовому масиві С(15) знайти суму довжин елементів з найменшою та найбільшою довжиною.

4. З елементів текстового масиву В(20) сформувати масиви, елементи яких мають однакову довжину.

5. В текстовому масиві А(15) поміняти місцями елементи з найменшою та найбільшою довжинами.

6. В текстовому масиві А(13)поміняти місцями:1-й елемент з 13-м, 2-й з 12-м, і т. д. Вивести на друк початковий та перетворений масиви.

7. Дано масив А(10) вивести на друк елементи в зростаючому порядку їх довжини.

8. Масив В(10)містить прізвища студентів. Впорядкувати його в алфавітному порядку.

9. Дано текстовий масив: папір, вода, башта, канал, висота, об'єм. Злити 2-й і 4-й елементи масиву і отриману текстову змінну поставити на друге місце. 4-й елнемент масиву знищити.

10. Дано текстовий масив В(12). Відсортувати його в порядку сподання довжин його елементів.

11. Дано числовий масив оцінок: 3, 4, 4, 5, 2, 3, 3, 4. Сформувати текстовий масив оцінок, замінивши: 3 на задовільно, 4 на добре, і т. д. Надрукувати отриманий масив.

12. Дано результати екзаменаційної сесії студентів:

№	П. І. Б.	Вища матем.	Фізика	Чисельні методи
1.	Іваненко	задовільно	добре	незадовільно
2.	Петренко	добре	незадовільно	добре
3.	Сидоренко	відмінно	задовільно	добре
4.	Крук	відмінно	відмінно	відмінно
5.	Глуб	задовільно	добре	добре

Підрахувати середній бал студента по кожному предмету.

13. Умова та ж, що і в 12 підрахувати середній бал кожного студента.

14. Дано текстовий масив А(10). Вивести на друк його елементи в зростаючому порядку їх довжин

15. Дано текстовий масив А(10). Вивести на друк його елементи в спадаючому порядку їх довжин.

16. В текстовому масиві А(8) даних, що містить 8 слів, підрахувати суму довжин елементів що стоять на парних місцях.

17. В текстовому масиві F(10) підрахувати суму довжин перших 7-ми елементів.

18. В текстовому масиві з 9-ти елементів знайти суму довжин елементів з 2-го по 6-й.

19. Дано текстовий масив В$(12). Відсортувати його в порядкку

спадання довжин його елементів і записати отриманий масив у масив A$(12).

20. Дано масив текстових змінних B(10). Створити масив c(10), що містить елементи масиву B(12) записані у зворотньому порядку.

21. Дано текстовий масив A(10). Знайти і надрукувати елементи найбільшої довжини. Вивести на друк даний елемент, його порядковий номер і довжину (кількість символів).

22. В текстовому масиві C(15) знайти різницю довжин елементів з найменшою та найбільшою довжиною. Надрукувати ці елементи.

23. Дано текстовий масив B(12). Відсортувати його в порядку зростання довжин його елементів.

24. В текстовому масиві з 10-ти елементів знайти суму довжин елементів з 3-го по 9-й.

25. Дано масив текстових змінних B(10). Створити масив C(10), що містить елементи масиву B(12), збільшені на 3 кожний.

26. В текстовому масиві A(15) поміняти місцями елементи з найменшою та найбільшою довжинами.

27. В текстовому масиві A(15)поміняти місцями:1-й елемент з 15-м, 2-й з 14-м, і т. д. Вивести на друк початковий та перетворений масиви.

28. Дано текстовий масив A(20). Знайти і надрукувати елементи найбільшої довжини. Вивести на друк даний елемент, його порядковий номер і довжину (кількість символів).

29. В текстовому масиві B(14) відшукати елемент з найбільшою довжиною, вивести його на друк разом з номером і довжиною.

30. В текстовому масиві F(12) підрахувати суму довжин перших 6-ми елементів.

Лабораторна робота №5
Розробка програм з файловими змінними. Робота з файлами

5.1. Мета роботи
Вивчити функції та алгоритми для організації роботи з файлами.

5.2. Теоретичні відомості
5.2.1. Структура

У мові програмування С++ структура представляє колекцію змінних, об'єднаних загальним іменем, яка забезпечує зручний засіб зберігання спорі- днених даних в одному місці. Структура – сукупність різних типів даних, оскільки вони складаються з декількох різних, але логічно взаємопов'язаних Програмування мовою С++. З цих самих причин структури іноді називають складеними або конгломератними типами даних. Структура - це об'єднання одного або більше об'єктів (змінних, масивів, покажчиків, інших структур). Як і масив, вона являє собою сукупність даних, але відрізняється від нього тим, що до її елементів необхідно звертатися на ім'я, а її різні елементи не обов'язково має належати до одного типу.

Структури зручно використовувати там, де різноманітні дані, що відносяться до одного і того ж об'єкту, необхідно об'єднувати. Наприклад, учня середньої школи характеризують такі дані: прізвище, ім'я, дата народження, клас, вік.

Оголошення структури здійснюється за допомогою ключового слова struct, за яким слід її тип, список елементів, укладених у фігурні дужки. Її можна представити в наступному загальному вигляді:

struct тип {
тип елемента 1 ім'я елемента 1;
тип елемента n ім'я елемента n;
};

Іменем елемента може бути будь-який ідентифікатор. В одному рядку можна записувати через кому кілька ідентифікаторів одного типу.

```
struct date {
int day;
int month;
int year;
};
```

Слідом за фігурною дужкою, що закінчує список елементів, можуть записуватися змінні даного типу, наприклад:

struct date {...} a, b, c;

При цьому виділяється відповідна пам'ять.

Виведене ім'я типу можна використовувати для оголошення запису, наприклад: struct date day;. Тепер змінна day має тип date.

Дозволяється вкладати структури одна на іншу. Для кращого сприйняття структури використовуємо кирилицю (українські літери) в ідентифікаторах, в мові С++ цього робити не можна.

Наприклад:

struct УЧЕНЬ {char Прізвище [15];

ім'я [15];

struct DATA{ int ДЕНЬ, МІСЯЦЬ, РІК;} ДАТА НАРОДЖЕННЯ;

int клас, вік;};

Наведений вище тип DATA включає три елементи: День, Місяць, Рік, що містять цілі значення (int).

Запис УЧЕНЬ включає елементи: ПРІЗВИЩЕ [15]; ІМ'Я [15]; ДАТА НАРОДЖЕННЯ, КЛАС, ВІК. ПРІЗВИЩЕ [15] і ІМ'Я [15] - це символьні масиви з 15 компонент кожен. Змінна ДАТА НАРОДЖЕННЯ представлена складовим елементом (вкладеної структурою) ДАТА. Будь-яку дату народження відповідають день місяця, місяць і рік. Елементи КЛАС і ВІК містять значення цілого типу (int). Після введення типів ДАТА і УЧЕНЬ можна оголосити змінні, значення яких належать цим типам.

Наприклад:

struct УЧЕНЬ УЧНІ [50];

Масив УЧНІ складається з 50 елементів типу УЧЕНЬ.

У мові С++дозволено використовувати масиви структури; записи можуть складатися з масивів та інших записів.

Щоб звернутися до окремого компоненту структури, необхідно вказати її ім'я, поставити крапку і відразу за нею написати ім'я потрібного елемента.

Наприклад:

УЧНІ [1]. КЛАС = 3;

УЧНІ [1]. ДАТА НАРОДЖЕННЯ. ДЕНЬ = 5;

УЧНІ [1]. ДАТА НАРОДЖЕННЯ. МІСЯЦЬ = 4;

УЧНІ [1]. ДАТА НАРОДЖЕННЯ. РІК = 1979;

Перший рядок вказує, що 1-й учень навчається в третьому класі, а наступні рядки - його дату народження: 5.04.79.

Кожен тип елемента структури визначається відповідним рядком оголошення в фігурних дужках. Наприклад, масив УЧНІ має тип УЧЕНЬ, рік є цілим числом. Оскільки кожен елемент запису відноситься до певного типу, його складене ім'я може з'являтися скрізь, де дозволено використовувати значення цього типу.

5.1.2. Бібліотеки введення/виводу і робота з файлами мовою С++

Операції введення/виводу в мові С організовані за допомогою бібліотечних функцій. Потрібно сказати, що система Borland C++ наслідує стандарт ANSI, названому також буферизованим (buffered) або форматованим (formated) введенням/виводом.

В той же час система Borland C++ підтримує й інший метод вводу/виводу, так званий UNIX-подібний, або неформатований (небуферизований) ввід/вивід.

Ми приділимо головну увагу першому методу - стандарту ANSI.

Мова С++ підтримує ще і власний об’єктно-орієнтований ввід/вивід.

Важливо зрозуміти, що таке файл (file) і потік (stream) і яке розходження між цими поняттями. Система введення/виводу мови С підтримує інтерфейс, що не залежить від того, який в дійсності використовується фізичний пристрій вводу/виводу, тобто є абстрактний рівень між програмістом і фізичним пристроєм. Ця абстракція і називається потоком. Спосіб же збереження інформації на фізичному пристрої називається файлом.

Незважаючи на те що пристрої дуже різні (термінал, дисковід, магнітна стрічка й ін.), стандарт ANSI мови С зв'язує кожен з пристроїв із логічним пристроєм, названим потоком. Оскільки, потоки не залежать від фізичних пристроїв, то та сама функція може записувати інформацію на диск, на магнітну стрічку або виводити її на екран.

У мові С існує два типи потоків: текстовий (text) і двійковий (binary).

Текстовий потік – це послідовність символів. Однак взаємооднозначної відповідності між символами, що подаються в потоці і виводяться на екран, може не існувати.

Двійковий потік – це послідовність байтів, що взаємооднозначно відповідають тому, що знаходиться на зовнішньому пристрої.

Файл у мові С - це поняття, що може бути застосоване до усього від файла на диску до термінала. Потік може бути зв'язаний із файлом за допомогою оператора відкриття файла. Як тільки файл відкритий, то інформація може передаватися між ним і вашою програмою.

Не всі файли однакові. Для прикладу з файла на диску ви можете вибрати 5-ий запис або замінити 10-ий запис. У той же час у файл, зв'язаний із друкувальним пристроєм, інформація може передаватися тільки послідовно в тому ж порядку. Це ілюструє саме головне розходження між потоками і файлами: усі потоки однакові, що не можна сказати про файли.

Операція відкриття файла зв'язує потік із визначеним файлом. Операція закриття файла розриває цей зв'язок. Якщо потік був відкритий для виводу, то при виконанні операції закриття файла відповідний буфер записується на зовнішній пристрій. Якщо програма закінчила роботу нормальним способом, усі файли автоматично закриваються.

Кожний потік, зв'язаний із файлом, має керуючу структуру, названу FILE. Вона описана в заголовному файлі stdio.h.

Вказівник на файлову змінну

Сполучною ланкою між файлом і потоком у системі введення/виводу стандарту ANSI мови С є вказівник на файл (file pointer). Вказівник на файл - це вказівник на інформацію, що визначає різні сторони файла: ім'я, статус, поточну позицію. Вказівник файла визначає ім'я файла на диску і його використання в потоці, асоційованим з ним. Вказівник файла - це вказівник на структуру типу **FILE**, яка визначена у файлі STDIO.H. У файлі STDIO.H визначені також наступні функції:

Функція	Дія функції
fopen( )	Відкрити файл
fclose( )	Закрити файл
putc( )	Записати символ у потік
getc( )	Прочитати символ із потоку
fseek( )	Змінити вказівник позиції файла на зазначене місце
fprintf( )	Форматний запис у файл
fscanf( )	Форматне читання з файла
feof( )	Повертає значення "істинно", якщо досягнутий кінець файла
ferror	Повертає значення "хибно", якщо виявлена помилка
fread( )	Читає блок даних із потоку
fwrite( )	Пише блок даних у потік
rewind( )	Встановлює вказівник позиції файла на початок
remove( )	Знищує файл

Щоб оголосити вказівник на файл, використовується оператор
*FILE *fput;*

Розглянемо більш докладно перераховані вище функції.

Функція fopen( ) виконує дві дії: по-перше, відкриває потік і зв'язує файл на диску з цим потоком; по-друге, повертає вказівник, асоційований із цим файлом. Прототип функції

*FILE *fopen(char filename, char mode);*

де mode - це рядок, що містить режим що відкриває файл. Можливі режими відкриття файлів перераховані нижче:

Режим	Дія
"r"	Відкрити для читання
"w"	Створити для запису
"a"	Відкрити для додавання в існуючий файл
"rb"	Відкрити двійковий файл для читання
"wb"	Відкрити двійковий файл для запису
"ab"	Відкрити двійковий файл для додавання
"r"	Відкрити файл для читання і запису
"w+"	Створити файл для читання і запису
"a+"	Відкрити файл для додавання або створити для читання і запису
"r+b"	Відкрити текстовий файл для читання і запису
"w+b"	Створити двійковий файл для читання і запису
"a+b"	Відкрити двійковий файл для додавання або створити для читання і запису
"rt"	Відкрити текстовий файл для читання
"wt"	Створити текстовий файл для запису
"at"	Відкрити текстовий файл для додавання
"r+t"	Відкрити текстовий файл для читання і запису
"w+t"	Створити текстовий файл для читання і запису
"a+t"	Відкрити текстовий файл для додавання або створити для читання і запису

Якщо ви збираєтеся відкрити файл з ім'ям test для запису, то досить написати

```
FILE* fp;
fp=fopen("test","w");
```

Однак рекомендується використовувати наступний спосіб відкриття файла:

```
FILE *fp;
```

```
If((fp=fopen("test",""w""))==NULL)
 {
 puts("Не можу відкрити файл \n");
    exit(1);
}
```

Цей метод визначає помилку при відкритті файла. Константа NULL визначена в STDIO. H. Функція exit(), яку ми використовували, має прототип у файлі STDLIB.H

void exit(int val);

і припиняє виконання програми, а розмір val повертає в операційну систему (програму , що викликає). При цьому перед припиненням роботи програма закриває усі відкриті файли, звільняє буфери, зокрема виводячи всі необхідні повідомлення на екран. Крім цього, існує функція abort() із прототипом

void abort(int val);

Ця функція негайно припиняє виконання програми без закриття файлів і звільнення буферів. У потік stderr вона направляє повідомлення "abnormal program termination".

Якщо файл відкритий для запису, то існуючий файл знищується і створюється новий файл. При відкритті файла для читання потрібно, щоб він існував. У випадку відкриття для читання і запису існуючий файл не знищується, однак створюється, якщо його не існує.

Запис функції в потік робиться функцією putc() із прототипом:

*int putc(int ch, FILE *fptr);*

Якщо операція була успішною, то повертається записаний символ. У випадку виникнення помилки повертається EOF.

Функція getc( ) зчитує символ із потоку, відкритого для читання функцією fopen( ).Прототип функції getc( )

*int getc(FILE *fptr);*

Історично склалося так, що getc() повертає значення int. Те ж саме можна сказати про аргумент ch в описі функції putc().Використовується ж в обох випадках тільки молодший байт. Функція повертає символ EOF, якщо досягнутий кінець файла або відбулася помилка при читанні файла. Щоб прочитати текстовий файл, можна використовувати конструкцію

```
ch=getc(fptr);
while(ch! =EOF) { ch=gelc(fptr) };
```

Коли зчитується двійковий файл, то визначити наявність кінця файла,

так само як при читанні текстового файла, не вдасться. Для визначення кінця текстового файла служить функція feof( ) із прототипом

*int feof(FILE *fptr).*

Функція повертає значення "істинно", якщо кінець файла досягнутий, і "нуль" - якщо ні. Наступна конструкція читає двійковий файл до кінця файла:

while(! feof(fptr)) { ch=getc(fptr); }

Для текстових файлів ця конструкція також може бути застосована. Функція fclose( ), оголошена у виді

*int fclose(FILE * fptr);*

повертає "нуль", якщо операція закриття файла була успішно. Інше значення означає помилку. При успішній операції закриття файла відповідні дані з буфера зчитуються у файл, відбувається звільнення блока керування файлом, асоційованого з потоком, і файл стає доступним для подальшого використання.

Якщо відбулася помилка читання або запису текстового файла, то відповідна функція повертає EOF. Щоб визначити, що ж у дійсності відбулося, служить функція ferror( ) із прототипом

*int ferror(FlLE *fptr);*

який повертає значення "істинно" при виконанні останньої операції з файлами і значення "хибно" у противному випадку. Функція ferror( ) повинна бути виконана безпосередньо після кожної операції з файлами, інакше її повідомлення про помилку може бути втрачено.

Функція rewind( ) встановлює індикатор позиції файла на початок файла, визначеного як аргумент функції, прототип цієї функції має вид

*void rewind(FILE *fptr);*

Borland C++ визначає ще дві функції буферизованого введення /виводу: putw( ) і getw( ). Ці функції не входять у стандарт ANSI мови С. Вони використовуються для читання і запису цілих чисел. Ці функції працюють точно так само, як putc( ) і getc( ).

Стандарт ANSI мови С включає також функції fread( ) і fwrite( ), що використовуються для читання і запису блоків даних:

*unsigned fread(void *buf,int bytes, int c, FILE *fptr);*

*unsigned fwriie(void *buf,int bytes, int c, FILE *fptr);*

де buf - вказівник на область пам'яті, звідки буде відбуватися обмін інформацією; с - кількість одиниць запису, кожна довжиною bytes байтів буде зчитано (записано); bytes - довжина кожної одиниці запису в байтах; fptr - вказівник на відповідний файл.

Читання і запис у файл необов’язково робити послідовно, можна це робити безпосередньо доступом до потрібного елемента файла за допомогою функції fseek( ), що встановлює вказівник позиції файла в

потрібне місце. Прототип цієї функції -

*int fseek(FILE *fptr,long numbytes, int origin);*

де fptr - вказівник на відповідний файл; numbytes - кількість байтів від точки відліку для встановлення поточної позиції вказівника файла, origin - один із макросів, визначених у stdio.h:

Точка відліку	Макрос	Значення
Початок файла	SEEK SET	0
Поточна позиція	SEEK CUR	1
Кінець файла	SEEK END	2

Коли починається виконання програми, автоматично відкривається 5 визначених потоків. Перші три з них - стандартний (stdin), стандартний вивід (stdout) і стандартний потік помилок (stder). У звичайній ситуації вони зв'язані з консоллю, однак можуть бути перенаправлені на інший потік. Можна використовувати stdin, stdout і stder як вказівники файлів в усіх функціях, що застосовують тип FILE.

Крім того, Borland C++ відкриває потоки stdprn і stdaux, асоційовані відповідно з принтером і послідовним портом комп'ютера. Ці потоки відкриваються і закриваються автоматично.

Стандарт ANSI включає також функції fprintf( ) і fscanf( ), що працюють аналогічно функціям printf( ) і scanf( ), за тим винятком,що зв'язані з файлами на диску. Прототипи цих функцій відповідно

*int fprintf(FILE *fptr, const char*string,...);*

*int fscanf(FILE *fptr, const char*string,...);*

де fptr - вказівник на файл, що повертається функцією fopen( ).

Функція remove( ) знищує зазначений файл. Прототип цієї функції :

*int remove(char *filename);*

Функція повертає значення 0 при успішній операції і ненульове значення в іншому випадку.

Оскільки мова С пов'язана з операційною системою UNIX, то в системі Borland C++ створена друга система введення/виводу. Ця система відповідає стандарту UNIX. Прототипи функцій знаходяться у файлі IO.H. Цими функціями є:

read( ) - читає буфер даних,
write( ) - пише в буфер даних,
open( ) - відкриває файл,
close( ) - закриває файл,
fseek( ) - пошук визначеного байта у файлі,
unlink( ) - знищує файл.

Опис цих функцій можна знайти в документації посібників Borland C++.

Складаючи програми для роботи з файлами слід пам'ятати, що:

- в програмі, яка виконує операції читання з файла, або запису в файл повинен бути оголошений вказівник на тип FILE;
- для того, щоб файл був доступним, його необхідно відкрити, вказавши, для виконання якої дії відкривається файл: читання, запису чи поновлення даних, а також тип файла (двійковий чи текстовий)ж
- при роботі з файлами можливі помилки, тому рекомендується при допомозі функції ferror провіряти результат виконання операцій з файлами (fopen());
- читання даних з текстового файла можна виконувати з допомогою функції fscanf(), запис – fprintf();
- після завершення роботи з файлом потрібно його обов'язково закрити (функція fclose());

Приклад запису у файл та зчитування з файла масиву структур, що описують інформацію про студентів групи і зберегають імя студента, вік та назву групи.

```
#include <iostream>
#include <stdio.h>
using namespace std;
const int size = 10; // константа, розмір масивів
struct Anketa{  // оголошення структури, що описує анкету
студента
  char name[20];
  int age;
  char group[10];
} studArray[size], studArray2[size]; // оголощення двох масивів
структур

int main(int argc, char** argv) {
  FILE *fp;
        //заповнюємо масив структур
  for (int i=0; i<size; i++){
            cout<<"Name = ";cin>>studArray[i].name;
            cout<<"Age = "; cin>>studArray[i].age;
            cout<<"Group = ";cin>>studArray[i].group;
  }
  // Відкриваємо файл для запису.
```

```
  if ((fp=fopen("balance", "w"))==NULL){
  printf("can't open file");    return 1;            }
  // Одним викликомзберігаємо весь масив studArray.
  fwrite(studArray, sizeof studArray,1,fp);
  fclose(fp);
  // Відкриваємо файл для зчитування.
  if ((fp=fopen("balance", "r"))==NULL){
  printf("can't open file");
  return 1;
  }
        // Одним "махом" зчитуємо весь масив studArray2.
  fread(studArray2, sizeof studArray2,1,fp);
       // Відображаємо вміст масиву.
  for (int i=0; i<size; i++){
  cout<<"Student "<<i<<" "<<studArray2[i].name;
        cout<<" "<<studArray2[i].age<<"
"<<studArray2[i].group<<endl;
  }
  fclose(fp); return 0;
}
```

5.3 Програма роботи

5.3.1. Запустити середовище Dev C++.

5.3.2. Скласти алгоритм програми створення та читання текстових файлів (додаток 1) згідно свого варіанту.

5.3.1. Написати програму до завдання 1 (додаток 1) згідно свого варіанту. Для опису інформації у файлі використати масив структур.

Вимоги до програм

- вхідні дані ввести оператором введення;
- привести результат виконання програм згідно завдання.

5.4. Обладнання та програмне забезпечення

5.4.1. Персональний комп’ютер.

5.4.2. Програмне забезпечення: IDE Dev-C++

5.5. Контрольні запитання

5.5.1 Які методи введення/виводу підтримує C++.

5.5.2. Що називають файлом і потоком.

5.5.3. Які Ви знаєте типи потоків.

5.5.4. Які Ви знаєте функції введення, виведення.

5.5.5. В яких заголовних файлах знаходяться прототипи функцій введення/виводу.

5.5.6. Які функції здійснюють введення/вивід рядка символів.

5.5.7. Що таке вказівник на файл, його произначення.

5.5.8. Перерахуйте основні функції роботи з файлами.

5.5.9. Які Ви знаєте режими відкриття файлів, яким чином вони задаються.

5.5.10. Яке призначення функції exit().

5.5.11. Яка функція призначена для закривання файлів.

5.5.12. Які функції призначені для читання і запису блоків даних.

5.5.13. Яка функція встановлює доступ до конкретного елемента файла.

5.5.14. Що таке структура?

5.5.15. Які типи даних може містити структура?

5.5.16. Як оголошується структурований тип даних?

5.5.17. Для чого використовується структура?

Завдання 1

Варіанти:

1. Сформувати файл “BOOK”, який містить інформацію про книги вашої бібліотеки. Інформація повинна містити прізвище автора, назву книги, видавництво та рік видання. Використовуючи сформований файл, роздрукувати інформацію про книги, видані у видавництві “Просвіта”.

2. Сформувати файл, якій містить інформацію про побутові холодильники: назва холодильника, вартість, об’єм холодильної камери, завод виготовлювач. Використовуючи сформований файл, роздрукувати інформацію про холодильники, вартість яких понад 55000 гривень.

3. Під час футбольної гри формується файл, який включає прізвище гравця та кількість набраних за гру балів. Використовуючи сформований файл, роздрукувати прізвища 3 найрезультативніших гравців команди.

4. Сформувати файл, який містить інформацію про потяги, які слідують до Києва (номер потяга, повна назва, час в дорозі). Використовуючи сформований файл, роздрукувати інформацію про потяги, час перебування в дорозі яких не перевищує 6 годин.

5. Сформувати файл , який містить інформацію про потяги, які відправляються зі станції Здолбунів (враховуючи транзитні): номер потяга, станцію призначення, час відправки, час в дорозі. Використовуючи сформований файл, роздрукувати інформацію про потяги, які йдуть до Львову.

6. Сформувати файл, що містить інформацію про студентів, які народилися влітку (червень, липень, серпень).

7. Сформувати файл “stud”, який має наступну структуру: прізвище студента, рік народження, стать. Роздрукувати, використовуючи файл, список студентів чоловічого роду та вказати їх вік. В кінці списку надрукувати середній вік студентів.

8. Сформувати файл “EXAM” за результатами екзаменаційної сесії (три екзамени). Інформація про студентів вводиться у символьному вигляді в наступному порядку: прізвище **N1**N2**N3, де Ni-оцінка. Використовуючи файл роздрукувати результати сесії у вигляді таблиці. Передбачити друк шапки таблиці з назвами дисциплін.

9. Протокол лижних гонок записати у файл “SCI”. Для кожного учасника ввести: прізвище, час участі в гонці (година, хвилина, секунда). Використовуючи сформований файл, роздрукувати прізвища учасників, котрі виконали норми (час менше 30 хвилин).

10. Сформувати файл “CAR”, який містить інформацію про власників автомобілів: прізвище, марка, колір. Використовуючи сформований файл, роздрукувати відомості про власників, які мають автомобілі “Audi” сірого кольору.

11. Сформувати файл, який містить дані про книги вашої особистої бібліотеки: прізвище автора, назва книги, видавництво, рік видання кількість сторінок. Використовуючи сформований файл, роздрукувати інформацію про книги, випущені у видавництві “Мир”, а також підрахувати загальну кількість таких книг.

12. Сформувати файл, який містить інформацію про магнітофони: марка, його вартість, клас. Використовуючи сформований файл, роздрукувати інформацію про магнітофони першого класу.

13. Сформувати файл - телефонний довідник. Інформація повинна містити прізвище, ім’я та по-батькові абонента, номер телефону. Роздрукувати весь довідник.

14. Сформувати файл “FRIEND” із прізвищ і дат народження ваших друзів. Використовуючи сформований файл, роздрукувати прізвища тих, хто народився взимку.

15. У шаховому турнірі приймають участь 10 шахістів. Сформувати файл який включає прізвища та результати ігор (перемога - 1, нічия -1, програш - 0). Використовуючи сформований файл, обробити результати чемпіонату і роздрукувати назви команд, які зайняли призові місця а також кількість перемог кожної команди.

16. В чемпіонаті з футболу приймає участь 16 команд. Сформувати файл команд і результатів матчу (виграш - 2 очка, нічия -1, програш - 0). Використовуючи сформований файл, обробити результати чемпіонату і роздрукувати назви команд, які зайняли призові місця, а також кількість перемог кожної команди.

17. В журналі обліку відвідувань щодня по кожному з предметів відмічають пропуск заняття студентами. Сформувати файл, який включає прізвище та дату відвідування занять з одного предмета (1 - присутній, 0 - відсутній) кожним студентом групи. Використовуючи сформований файл, сформувати список тих студентів, які мають більше 5 пропусків.

18. Сформувати файл, який включає прізвища та оцінки студентів на протязі семестру з дисципліни “Обчислювальна техніка”. Використовуючи сформований файл, роздрукувати прізвища тих студентів, в котрих середній бал з дисципліни - 4.

19. До 20 спортивних журналістів звернулись із проханням назвати 3 кращих футболістів сезону. Сформувати файл, який включає прізвища футболістів, кількість набраних очків від кожного журналіста (3 - перше місце, 2 - друге місце, 1 - третє місце). Використовуючи сформований файл, визначити 3 кращих гравців.

20. Сформувати файл, який включає прізвища та посади викладачів, які викладають дисципліну “Обчислювальна техніка” на всіх факультетах інституту. Використовуючи сформований файл, роздрукувати прізвища тих викладачів, які починаються на букву “В”.

21. Сформувати файл, який містить прізвища та імена студентів своєї групи. Використовуючи сформований файл, роздрукувати прізвища тих студентів, які починаються на букву “К”.

22. Сформувати файл, який містить прізвища та ініціали викладачів, які викладають у вашій групі, та відповідні предмети. Роздрукувати прізвища викладачів з кафедри електротехніки і автоматики.

23. Сформувати файл, який містить результати вашої атестації. Використовуючи створений файл, роздрукувати ті предмети, з яких ви отримали “5”. Якщо ви не отримали жодної п‘ятірки, то роздрукуйте предмети, з яких ви отримали оцінку “4” або “3”).

24. Сформувати файл “Journal”, який містить інформацію про журнали бібліотеки. Інформація повинна містити прізвище автора, назву статті, назву журналу, видавництво та рік видання. Використовуючи сформований, роздрукувати інформацію про журнали, видані у видавництві “ файл НУВГП”.

25. Сформувати файл, якій містить інформацію про ноутбуки: назва виробника, вартість, діагональ екрану. Використовуючи сформований файл, роздрукувати інформацію про ноутбуки, вартість яких понад 25000 гривень.

26. Сформувати файл “GROUP” із прізвищ і дат народження студентів групи. Використовуючи сформований файл, роздрукувати прізвища тих, хто народився влітку.

27. В журналі обліку відвідувань щодня по кожному з предметів

відмічають пропуск заняття студентами. Сформувати файл, який включає прізвище та дату відвідування занять з одного предмета (1 - присутній, 0 - відсутній) кожним студентом групи. Використовуючи сформований файл, сформувати список тих студентів, які мають більше 10 пропусків.

28. Сформувати файл, який включає прізвища та оцінки студентів на протязі семестру з дисципліни “Вища математика”. Використовуючи сформований файл, роздрукувати прізвища тих студентів, в котрих середній бал з дисципліни - 3.

29. Протокол змагання з плавання у файл “Competition ”. Для кожного учасника ввести: прізвище, час участі в змаганні (хвилин, секунда). Використовуючи сформований файл, роздрукувати прізвища учасників, котрі виконали норми (час менше 2 хвилини).

30. Сформувати файл “CAR”, який містить інформацію про власників автомобілів: прізвище, марка, колір. Використовуючи сформований файл, роздрукувати відомості про власників, які мають автомобілі “Ford”.

Лабораторна робота 6
Розробка програм з користувацькими класами. Робота з класами та об‘єктами.

6.1. Мета роботи

Формування навиків роботи із класами, алгоритмами їх оголошення та обробки. Набути навиків об‘єктно-орієнтованого програмування.

6.2. Теоретичні відомості

6.2.1. Клас

Головною відмінністю C++ від C є можливість опрацьовувати новий тип даних - ***клас. Клас*** – визначений користувачем тип даних (змінні будь-якого типу, інші класи чи вказівники), який створюється для опису абстрактної множини об‟єктів – членів класу. Ідея класу полягає в об'єднанні даних і алгоритмів їх опрацювання. Дані називаються ***полями класу,*** алгоритми - ***методами,*** а власне об'єднання - *інкапсуляцією.* Методи опрацьовують поля і зовнішні дані, власне вони реалізують ідею класу. Класи володіють властивістю ***наслідування,*** яка забезпечує використання класом-нащад-ком (надалі — похідним класом) полів і методів базового класу (предка, батьківського класу). Кожний клас може мати довільну кількість нащадків, що дає змогу створювати ієрархічні дерева успадкування. Нащадок може перекривати деякі методи предка, і тоді метод з одним іменем для різних класів виконуватиметься порізному. Це називають ***поліморфізмом*** методів.

6.2.2. Інкапсуляція

Створення нового класу аналогічне до створення нової структури:

class <назва класу>
{
<специфікатор доступу>:
<тип поля 1> <назви полів 1>;

<тип поля N> <назви полів N>;
<декларації чи описи методів класу>;
};

Методами класу є функції, які визначені для полів чи зовнішніх змінних.

В цілому поняття класу нагадує поняття структури в С++ за виключенням ряду моментів: - він містить в собі ряд специфікацій доступу до об‟єктів класу; - як правило він включає в себе елементи - функції, що задають правила обробки об‟єктів класу; - для створення-знищення об‟єктів класу використовуються спеціальні функції – конструктор та деструктор. На відміну від полів структури, які є доступними завжди, в класі члени та методи можуть мати різний рівень доступу.

Специфікатори доступу описуються так:

Специфікатор доступу	**Опис**
private	Доступність лише для методів класу
protected	Доступність лише для методів класу та методів похідних класів
public	Доступність для будь-якої зовнішньої функції

В описі класу специфікатор доступу може бути відсутній. Тоді за замовчуванням активним є специфікатор **private,** поки явно не задано інше. Зазначимо, що структура в усьому аналогічна до класу, крім того, що за замовчуванням її поля та методи загальнодоступні **(public).** Від структур не можна успадкувати класи-нащадки.

Створимо, наприклад, клас TPoint, який міститиме координати точки і такі методи: засвічування, гасіння й переміщення точки. Опис класу TPoint має такий вигляд:

```
class TPoint
{ protected:
int x,y; // Координати
public:
TPoint(int a, int b);            // Ініціалізує поля координат числами
a і b
void On()                        // Рисує точку поточним кольором
{Draw(getcolor());}
void Off()        // Витирає точку - малює її кольором фону
{Draw(getbkcolor());}
virtual void Draw(int color)  // Рисує точку кольором color
{putpixel(x, y, color);}
// Переміщає точку на екрані на dx вправо і на dy вниз
void Move (int dx, int dy);
};
```

Функція TPoint створює екземпляр класу і заповнює його поля конкретними значеннями. Такий метод класу називається **конструктором.**

Конструктор *завжди* має назву класу. Значення ключового слова **virtual** в описі методу Draw() буде пояснено нижче.

В класі може бути визначено кілька конструкторів, які відрізняються списком параметрів. При створенні об‟єктів класу викликатися буде (перевантаження конструктора) конструктор із відповідною кількістю та типом вхідних параметрів.

Конструктор може мати вхідні параметри, а може їх і не мати. Конструкттор, який не має вхідних параметрів називається конструктором за замовчуванням.

Клас може і не мати конструктора у явному вигляді. Тоді розподіл пам’яті та ініціалізація об’єкту виконуються системою автоматично стандартним шляхом. Стандартна процедура не враховує особливостей класу,тому може бути некоректною і приводити до помилок в роботі програми.

Деструктор – це функція-метод класу, яка відповідає за коректневивільнення пам‟яті при знищенні об‟єкту. Деструктор завжди має ім’я класу, перед яким ставиться символ „~”. Якщо деструктор у програмі не визначено, то він генерується на етапі компіляції програми автоматично.

Локальні об‟єкти видаляються деструктором тоді, коли вони виходять за межі області видимості, о глобальні об‟єкти – по завершенні роботи функції main(). Деструктор, при необхідності, може виконувати і будьякі інші дії – вивід кінцевих значень елементів даних чи текстових повідомлень при налагодженні програми.

Конструктори і деструктори не можуть повертати значень і тому при їх описі відсутній тип результату.

Зазначимо, що значення полів класу бажано змінювати за допомогою методів. Наприклад, змінити розміщення точки можна за допомогою методу Move.

Поза описом класу заголовок методу має такий вигляд:

<назва класу>::<назва методу>(<список формальних параметрів>);

Опишемо методи створеного класу:

```
TPoint::TPoint(int a, int b)
{
x = a; y = b;
} void TPoint::Move(int dx, int dy)
{
Off();
```

```
x+=dx; y+=dy;
On();
}
```

Змінну (об'єкт) класу оголошують аналогічно до інших класів:

TPoint n(12,24), m(100,200);

Метод класу викликають так:

<назва об'єкта>.<назва методу>(<список фактичних параметрів>);

Оголосити й використати екземпляр Point класу TPoint можна, наприклад, так:

```
TPoint Point(50,50);
Point.On();
Point.Move(35, 70);
Point.Of();
```

або за допомогою динамічних змінних:

```
TPoint* PointPtr = new Point(100,100);
PointPtr->On();
PointPtr->Move(35, 70);
PointPtr->Off();
```

6.2.3. Перевантаження операції

Для класів визначений спеціальний метод **operator,** а саме:

<тип>operator<символ>(<формальні параметри>)
{<тіло методу>}

У цьому разі як символ можна використовувати усі арифметичні операції, команду присвоєння, команди присвоєння, суміщені з арифметичними операціями та різні пари дужок, наприклад: **operator+, operator/=, operator=, operator()** тощо. Правила опису власних оператор-методів аналогічні до правил створення звичайних функцій чи методів.

Використовуючи клас TPoint та operator() нарисуйте 1000 точок, випадково розміщених на екрані.

```
#include <graphics.h>
#include <conio.h>
#include <stdlib.h>
class TPoint
{
protected: int x, y;
public:
TPoint(int a = 0, int b = 0)
{
x = a; y = b;
}
void On()
{Draw(getcolor());}
void Off()
{Draw(getbkcolor());)
virtual void Draw(int color)
{putpixel(x, y, color);)
TPoint& operator()(int i,int j)
{
x = i,y = j;
return *this;
}
};
void main()
{
int gdriver = DETECT, gmode, errorcode;
initgraph(&gdriver, &gmode,"");
TPoint P;
randomize();
for(int i = 0; i < 1000; i++) P(random(i), random(i) ).On();
getch();
closegraph();
}
```

У цій програмі для задання координат точки використовуємо operator(). Щоб можна було застосувати, наприклад команди P(30,80).On() або P(100,200).Off(), operator() має належати типу **Tpoint.** Тому в описі оператор-методу operator() використовуємо посилання TPoint&. Команда **return** має повертати екземпляр (значення змінної) типу TPoint. Оскільки на момент опису класу ім'я змінної цього класу невідоме, у C++ існує

спеціальне ключове слово **this** - вказівник на цю змінну.

6.2.4. Приклад перевантаження операторів

Програма для роботи із комплексними числами. Членами класу є дійсна та уявна частина комплексного числа. Методами класу є: вивід комплексного числа на екран; конструктор для ініціалізації комплексних чисел; деструктор; обчислення та вивід модуля комплексного числа; обчислення та вивід аргументу комплексного числа.

Для ініціалізації комплексного числа створено перевантажений конструктор:

complex() ініціалізація комплексного числа виду a +i·a (якщо параметр в дужках не задано, то по замовчуванню ініціалізується комплексне число 1+i);

complex(double, double) ініціалізація комплексного числа виду a +i·b.

Деструктор ~**complex()** створено також, хоча для нашого класу задач його використання не є обов‟язковим.

Для роботи із комплексними числами описано перевантажені операції:

bool operator ==(complex) порівняння двох комплексних чисел;
complex **operator +(complex)** додавання двох комплексних чисел;
complex operator -(complex) віднімання двох комплексних чисел;
complex operator *(complex) множення двох комплексних чисел;
complex operator /(complex) ділення двох комплексних чисел;
complex operator =(complex) операція присвоєння для комплексних чисел;
void operator ++() префіксне збільшення на 1 дійсної частини;
void operator ++(int) постфіксне збільшення на 1 уявної частини;
void operator --() префіксне зменшення на 1 дійсної частини;
void operator --(int) постфіксне зменшення на 1 уявної частини.

```
#include <iostream>
#include <math.h>
#include <windows.h>
#define pi 3.1415926
using namespace std;

class complex {
private:
   double x;
   double y;
```

```
public:
   complex();
   complex(double, double);
   ~complex();

   double modul();
   double argument();
   void show_complex();

   void operator++();
   void operator++(int);
   void operator--();
   void operator--(int);
   complex operator+(complex);
   complex operator-(complex);
   complex operator*(complex);
   complex operator/(complex);
   complex operator=(complex);
   bool operator==(complex);
};

bool complex::operator==(complex chislo)
// Порівняння двох комплексних чисел
{
   if (x == chislo.x && y == chislo.y) {
      return true;
   }
   return false;
};

complex complex::operator+(complex a)
//Додавання
{
   complex temp;
   temp.x = x + a.x;
   temp.y = y + a.y;
   return temp;
};

complex complex::operator-(complex a)
//Віднімання
```

```
{
   x = x - a.x;
   y = y - a.y;
   return *this;
};

complex complex::operator*(complex a)
//Множення
{
   x = x * a.x - y * a.y;
   y = y * a.x + x * a.y;
   return *this;
};

complex complex::operator/(complex a)
//Ділення
{
   x = x * a.x + y * a.y;
   y = y * a.x - x * a.y;
   return *this;
};

complex complex::operator=(complex a)
//Операція присвоєння
{
   x = a.x;
   y = a.y;
   return *this;
};

//Префіксне збільшення на 1 дійсної частини
void complex::operator++()
//Префіксне збільшення на 1 дійсної  частини
{
   x++;
};

void complex::operator--()
//Префіксне зменшення на 1 дійсної частини
{
   x--;
```

```
};

void complex::operator++(int)
//Постфіксне збільшення на 1 уявної частини
{
   y++;
};

void complex::operator--(int)
//Постфіксне зменшення на 1 уявної частини
{
   y--;
};

double complex::modul()
{
   return pow(x * x + y * y, 1 / 2.0);
};

double complex::argument()
{
   return atan2(y, x) * 180 / pi;
};
void complex::show_complex()
{
   if (y >= 0)
      cout << x << "+" << y << "i" << endl;
};

complex::complex()
//Конструктор 1
{
   x = 1;
   y = 1;
};

complex::complex(double x0, double y0)
//Конструктор 2
{
   x = x0;
   y = y0;
```

```
};

complex::~complex()
  //Деструктор
  {};

int main()
{
  SetConsoleOutputCP(1251);
  SetConsoleCP(1251);
  /*Ввід комплексного числа c1 за допомогою конструктора 1 */
  complex c1;
  cout << "Комплексне число c1=\t";
  c1.show_complex();

  /*Ініціалізація числа c2 за допомогою конструктора 2 */
  complex c2(-1.0, 2.0);
  cout << "Комплексне число c2=\t";
  c2.show_complex();
  cout << endl;
  c2++;
  cout << "Комплексне число c2 після префіксної операції ++
=\t";
  c2.show_complex();

  ++c2;
  cout << "Комплексне число c2 після постфіксної операції ++
=\t";
  c2.show_complex();

  c2 = c2 - c1;
  cout << "Комплексне число c2 після віднімання віднього c1=\t";
  c2.show_complex();

  cout << endl;
  complex c3;
  c3 = ((c2 + c1) / (c2 + c2)) * c1;
  cout << "Обчислення виразу c3=((c2+c1)/(c2+c2))*c1 : c3=\t";
  c3.show_complex();

  cout << "Порівняння чисел c2 та c3 -\t";
```

```
        if (c2 == c3)
           cout << "Числа рівні \n";
        else
           cout << "Числа не рівні між собою\n";

        cout << "Модуль комплексного числа c3=\t" << c3.modul() <<
endl;
        cout << "Аргумент комплексного числа c3=\t" <<
c3.argument();

        return 0;
    }
```

6.4. Програма роботи

6.4.1. Запустити середовище програмування Dev-C++.

6.4.2. Написати програму для розв'язку задачі згідно свого варіанту (додаток 1).

Вимоги до програм

- Створити клас згідно варіанту. Доповнити клас перевантаженими конструкторами (один - без параметрів, ініціалізує поля класу значеннями за замовчуванням, напр. 0, другий - параметризований, ініціалізує поля класу даними, введеними користувачем з клавіатури) та деструктором(виводить повідомлення про знищення обє'кта).

6.4. Обладнання та програмне забезпечення

6.4.1. Персональний комп'ютер.

6.4.2. Програмне забезпечення: IDE Dev-C++

6.5. Контрольні запитання

6.5.1. Що називають полями класу?

6.5.2. Що називають методами класу?

6.5.3. Якими властивостями володіють класи?

6.5.4. Що таке наслідування класів?

6.5.5. Що називають поліморфізмом методів?

6.5.6. Як в C++ реалізується перевантаження операторів?

6.5.7. Для чого призначений конструктор класу?

Завдання 1

Варіанти:

1. Створити клас дробів. Членами класу є чисельник та знаменник. Методами класу є: ввід дробу з клавіатури; вивід дробу на екран; обчислення та вивід значення дробу. Доповнити клас перевантаженими операціями "+", "-", "*", "/". Написати програму, що демонструє роботу з класом.
2. Створити клас векторів. Членами класу є декартові координати початку та кінця вектора в просторі. Методами класу є: ввід вектора з клавіатури; вивід вектора на екран; обчислення та вивід довжини вектора. Доповнити клас перевантаженими операціями "+", "-". Написати програму, що демонструє роботу з класом.
3. Створити клас матриць розміру 3×3 . Членами класу є елементи матриці. Методами класу є: ввід матриці з клавіатури; вивід матриці на екран; обчислення та вивід визначника матриці. Доповнити клас перевантаженими операціями "+", "-". Написати програму, що демонструє роботу з класом.
4. Створити клас відрізків на площині. Членами класу є координати кінців відрізка. Методами класу є: ввід відрізка з клавіатури; вивід відрізка на екран; обчислення та вивід на екран довжини відрізка. Доповнити клас перевантаженими операціями "+" (додавання відрізків згідно правил додавання векторів), "-" (віднімання відрізків згідно правил віднімання векторів). Написати програму, що демонструє роботу з класом.
5. Створити клас для роботи із часом в межах доби. Членами класу є години, хвилини та секунди. Методами класу є: ввід часу з клавіатури; вивід часу на екран; обчислення та вивід на екран часу, що залишився до завершення доби. Доповнити клас перевантаженими операціями "+", "-". При всіх операціях передбачити перевірку виходу отриманого значення за допустимі межі. Написати програму, що демонструє роботу з класом.
6. Створити клас для роботи із колом. Членами класу є радіус кола та координати його центру. Методами класу є: ввід кола із клавіатури; вивід кола на екран; обчислення площі круга та довжини кола і вивід результату на екран. Доповнити клас перевантаженими операціями "*" (множення радіусу на число); "/" (ділення радіусу на число). Написати програму, що демонструє роботу з класом.
7. Створити клас прямокутників. Членами класу є довжини сторін прямокутника. Методами класу є: ввід прямокутника із клавіатури; вивід прямокутника на екран; обчислення периметра та його площі і вивід результату на екран. Доповнити клас перевантаженими операціями префіксний ++ збільшує довжини сторін прямокутника на 1; префіксний -- зменшує довжини сторін прямокутника на 1. Написати програму, що

демонструє роботу з класом.

8. Створити клас трикутників. Членами класу є довжини сторін трикутника. Методами класу є: ввід трикутника із клавіатури; вивід трикутника на екран; обчислення периметра та площі і вивід результату на екран. Доповнити клас перевантаженими операціями префіксний ++ збільшує довжини сторін трикутника на 1; префіксний -- зменшує довжини сторін трикутника на 1. Написати програму, що демонструє роботу з класом.

9. Створити клас для роботи із датою. Членами класу є рік, місяць та день місяця. Методами класу є: ввід дати з клавіатури; вивід дати на екран; обчислення та вивід на екран пори року, що відповідає даній даті. Доповнити клас перевантаженими операціями "+" (додавання відповідно років, місяців та днів); "-" (віднімання відповідно років, місяців та днів). При всіх операція передбачити перевірку виходу отриманого значення за допустимі межі. Написати програму, що демонструє роботу з класом.

10. Створити клас матриць розміру 4×4 . Членами класу є її елементи. Методами класу є: ввід матриці; вивід матриці на екран; обчислення та вивід на екран максимального елементу матриці. Доповнити клас перевантаженими операціями "+", "-". Написати програму, що демонструє роботу з класом.

11. Створити клас точок. Членами класу є координати точки на площині. Методами класу є: ввід точки з клавіатури; вивід координат точки на екран; обчислення та вивід номера квадранта системи координат, в якому точка знаходиться. Доповнити клас перевантаженими операціями "+" (додаються відповідні координати двох точок); "-" (віднімаються відповідні координати двох точок). Написати програму, що демонструє роботу з класом.

12. Створити клас поліномів розмірності 4. Членами класу є коефіцієнти полінома. Методами класу є: ввід коефіцієнтів полінома; вивід полінома на екран; обчислення та вивід значення полінома для заданого значення x. Доповнити клас перевантаженими операціями "+" (почленне додавання поліномів); "-" (почленне віднімання поліномів). Написати програму, що демонструє роботу з класом.

13. Створити клас тріад чисел. Членами класу є три дійсних числа. Методами класу є: ввід трьох чисел; вивід чисел на екран; обчислення та вивід найбільшого та найменшого числа. Доповнити клас перевантаженими операціями "+" (почленне додавання тріад чисел); "-" (почленне віднімання тріад чисел). Написати програму, що демонструє роботу з класом.

14. Створити клас еліпсів. Членами класу є довжини півосей еліпса. Методами класу є: ввід еліпса із клавіатури; вивід еліпса на екран;

обчислення площі та периметра еліпса і вивід результату на екран. Доповнити клас перевантаженими операціями "+" (додаються відповідні півосі еліпсів); "*" (множення півосей еліпса на число). Написати програму, що демонструє роботу з класом.

15. Створити клас точок в просторі. Членами класу є декартові координати точки. Методами класу є: ввід точки з клавіатури; вивід координат точки на екран; обчислення та вивід полярних координат точки. Доповнити клас перевантаженими операціями "+" (додаються координати точок); "-" (віднімаються координати точок). Написати програму, що демонструє роботу з класом.

Лабораторна робота 7
Розробка програм з користувацькими класами. Робота з класами та об‘єктами.

7.1. Мета роботи

Навчитися працювати з класами та об‘єктами. Набути навиків об‘єктно-орієнтованого програмування.

7.2. Теоретичні відомості

7.2.1. Наслідування

Для оголошення успадкування використовується загальна форма:

class <назва нащадка> : < доступ> <назва предка>
{
<додані поля класу>;
<декларації чи описи доданих і перевизначених методів>;
}

Клас, властивості якого успадковуються, називається **базовим**, а клас, який успадковує властивості базового, - **похідним**. Тут *доступ* – одне з трьох ключових слів: *public, private* або *protected.*

7.2.2. Доступ до елементів базового класу

Специфікатор доступу визначає, як елементи базового класу успадковуються похідним класом. Якщо специфікатором доступу успадкованого базового класу є *public,* то всі відкриті члени базового класу стають відкритими і в похідному. Якщо специфікатором доступу успадкованого базового класу є *private*, то всі відкриті члени базового класу стають закритими в похідному. В обох випадках всі закриті члени базового класу залишаються закритими і недосяжними для похідного класу. Якщо специфікатором доступу є *private*, то відкриті члени базового класу стають закритими в похідному, проте ці члени залишаються доступними для функцій-членів похідного класу.

Іноді можлива ситуація, коли члени базового класу, залишаючись закритими, були досяжні для похідного класу. Для реалізації цієї ідеї в C++ уведено специфікатор доступу *protected* (захищений). Специфікатор доступу *protected* еквівалентний *private* з єдиною різницею: захищені члени базового класу досяжні для членів усіх похідних класів цього базового класу. Поза базовим класом або похідними класами захищені члени

недосяжні. Специфікатор доступу *protected* може бути у будь-якому місці оголошення класу, хоча, як правило, його розміщують після оголошення закритих членів та перед оголошенням відкритих членів. Повна загальна форма оголошення класу має вигляд:

```
class ім'я_класу
{
... // закриті члени
protected:
... // захищені члени
public:
... // відкриті члени
};
```

7.2.3. Конструктори, деструктори та успадкування

Базовий клас, похідний клас або обидва можуть мати конструктори та/або деструктори. Якщо і у базового, і у похідного класів є конструктори та деструктори, то конструктори виконуються у порядку успадкування, а деструктори – у зворотному порядку. Таким чином, конструктор базового класу виконується раніше, ніж конструктор похідного класу. Для деструкторів правильний зворотний порядок: деструктор похідного класу виконується раніше від деструктора базового класу.

Розглядаючи поняття успадкування, необхідно звернути увагу на особливості передачі аргументів для конструкторів похідного та базового класів. Коли ініціалізація виконується тільки в похідному класі, аргументи передаються звичайним шляхом. Проте, під час передавання аргумента конструктору базового класу виникають деякі складності. Передусім усі необхідні аргументи базового та похідного класів передаються конструктору похідного класу. Потім завдяки розширеній формі оголошення конструктора похідного класу відповідні аргументи передаються далі в базовий клас.

Синтаксис передавання аргументів з похідного в базовий клас такий:

```
ім'я_похідного_класу ( список_арг1 ) :
ім'я_базового_класу ( список_арг2 )
{
... //тіло конструктора похідного класу
}
```

Для базового та похідного класів допустимо використовувати одні й ті самі аргументи. Для похідного класу допустимим є також ігнорування всіх аргументів та передача їх напряму в базовий клас.

У більшості випадків конструктори базового та похідного класів не використовують один і той самий аргумент. Тоді у разі потреби передавання кожному конструктору класу одного або більше параметрів слід передати конструктору похідного класу всі аргументи, необхідні конструкторам цих двох класів. Потім конструктор похідного класу просто передає конструктору базового ті аргументи, які йому потрібні.

7.2.4. Множинне успадкування

Є два способи, завдяки яким похідний клас може успадковувати більше, ніж один базовий клас. По-перше, похідний клас може використовуватися як базовий для іншого похідного класу, створюючи багаторівневу ієрархію класів. В цьому випадку вихідний базовий клас є **непрямим** (inderect) **базовим класом** для іншого похідного класу. По-друге, похідний клас може прямо успадковувати більше, ніж один базовий клас. У такій ситуації комбінація двох або більше базових класів допомагає створенню похідного класу.

Коли клас використовується як базовий для похідного, який, у свою чергу, є базовим для іншого похідного класу, конструктори цих трьох класів викликаються в порядку успадкування. Деструктори викликаються у зворотному порядку.

Якщо похідний клас напряму успадковує множину базових класів, використовується таке розширене оголошення:

```
class ім'я_похідного_класу :
спец_доступу1 ім'я_базового_класу1,
спец_доступу2 ім'я_базового_класу2,
. . .
спец_доступуN ім'я базового_класуN
{
... // тіло класу
        }
```

Коли успадковується множина базових класів, конструктори використовуються зліва направо у порядку, що задається в оголошенні похідного класу. Деструктори виконуються у зворотному порядку. Коли клас успадковує множину базових класів, конструкторам яких необхідні аргументи, похідний клас передає ці аргументи, використовуючи розширену форму оголошення конструктора похідного класу:

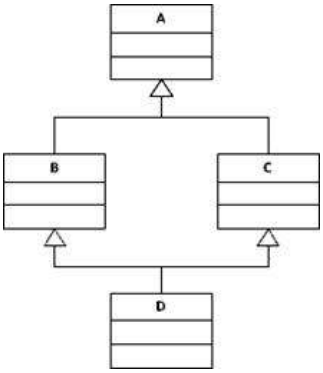
```
ім'я_похідного_класу ( список_арг ):
ім'я_базового_класу1( список_арг1 ),
ім'я_базового_класу2( список_арг2 ),
. . .
```

ім'я_базового_класуN(список_аргN)
{
... // тіло конструктора похідного класу
}

Коли похідний клас успадковує ієрархію класів, кожний похідний клас повинен передавати попередньому базовому класу по ланцюжку необхідні аргументи.

7.2.5. Віртуальні базові класи

У разі багаторазового прямого успадкування похідним класом одного і того самого базового класу може виникнути проблема. Розглянемо такий варіант:



Тут базовий клас *D* успадковується похідними класами *B* і *C*. Похідний клас *A* прямо успадковує ці класи. Це означає, що клас *D* фактично успадковується класом *A* двічі – через класи *B* і *C***.** Однак, якщо член класу *Базовий* буде використовуватись у класі *A*, це зумовить неоднозначність, оскільки в ньому буде дві копії класу *D***.** Для попередження такої ситуації в C++ включений механізм віртуального базового класу (virtual base class). У цьому випадку перед специфікатором доступу базового класу необхідно поставити ключове слово *virtual*, наприклад:

class derived2: ***virtual*** *public base*

7.3. Програма роботи

7.3.1. Запустити середовище програмування Dev-C++.

7.3.2. Написати програму для розв'язку задачі згідно свого варіанту (додаток 1).

Вимоги до програм

Створити клас у відповідності з вказаною предметною областю. Передбачити можливість роботи з довільним числом записів, а також реалізувати:

– конструктори без параметрів та з параметрами;

– функцію для виведення інформації про об'єкт на екран;

– функцію пошуку потрібної інформації за конкретною ознакою;

– функцію редагування записів.

При розробці програми слід здійснити захищення даних (опис з моди-фікатором *private*) для ізоляції елементів-даних класу від підпрограм, в яких цей клас використовується.

7.4. Обладнання та програмне забезпечення

7.4.1. Персональний комп'ютер.

7.4.2. Програмне забезпечення: IDE Dev-C++

7.5 Контрольні запитання

7.5.1. Поясніть принцип ООП — успадкування. Наведіть приклади.

7.5.2. Які ви знаєте специфікатори доступу?

7.5.3. Що відбувається з відкритими та закритими членами базового класу, якщо базовий клас успадковується як відкритий похідним?

7.5.4. Що відбувається з закритими та відкритими членами базового класу, якщо базовий клас успадковується як закритий похідним?

7.5.5. Поясніть, навіщо потрібна категорія захищеності *protected*?

7.5.6. Якщо один клас успадковується іншим, яким порядок виклику конструкторів та деструкторів? Наведіть приклади.

7.5.7. Що таке множинне успадкування?

Завдання 1

Варіанти:

1. Створити базовий клас ФІГУРА, в якому визначено координати однієї з вершин геометричної фігури. Створити похідні класи: ПРЯМОКУТНИК (додатково визначаються довжини двох сторін), КОЛО (додатково визначається радіус), ПРЯМОКУТНИЙ ТРИКУТНИК (додатково визначаються довжини катетів).

2. Створити клас ФУНКЦІЯ обчислення залежності y від x. Створити похідні класи: ЛІНІЙНА ФУНКЦІЯ ($y(x)=ax+b$, додаються два дійсних значення - коефіцієнти a і b), КВАДРАТИЧНА ФУНКЦІЯ ($y(x)=cx^2+ax+b$, додаються три дійсних значення a, b, c), КУБІЧНА ФУНКЦІЯ ($y(x)=dx^3+cx^2+ax+b$, додаються чотири дійсних значення a, b, c, d).

3. Створити базовий клас ПРОГРЕСІЯ з одним дійсним полем даних - перший член прогресії. Створити похідні класи: АРИФМЕТИЧНА ПРОГРЕСІЯ і ГЕОМЕТРИЧНА ПРОГРЕСІЯ. Кожен клас має додткове поле типу double - постійна різниця для арифметичної і постійне

відношення для геометричної прогресії. Визначити функції обчислення суми та n-го елемента.

4. Створити базовий клас - геометрична ЧОТИРИКУТНИК, в якому визначено координати нижньої лівої вершини геометричної фігури, і похідні класи - КВАДРАТ (має додаткове поле типу double – довжина сторони), ПРЯМОКУТНИК (має два додаткові поля типу double – довжини двох сторін), ТРАПЕЦІЯ (задано довжини чотирьох сторін).

5. Створити базовий клас ФІГУРА, в якому визначено координати однієї з вершин геометричної фігури. Створити похідні класи: КУБ (має додаткове поле типу double – довжина ребра), ТЕТРАЕДР (правильний, має додаткове поле типу double – довжина ребра), КУЛЯ (має додаткове поле типу double – радіус).

6. Створити абстрактний клас КРИВІ обчислення залежності y від x. Створити похідні класи: ПРЯМА(y(x)=ax+b, додаються два дійсних значення - коефіцієнти a і b), ПАРАБОЛА ($y(x)=ax^2+b$, додаються дійсні значення a, b), ГІПЕРБОЛА ФУНКЦІЯ ($y(x)=ax^3+ b$, додаються дійсні значення a, b).

7. Створити базовий клас ФІГУРА, в якому визначено координати однієї з вершин геометричної фігури. Створити похідні класи: КОНУС (має додаткові поля типу double – радіус основи і вистота), ЦИЛІНДР (має додаткові поля типу double – радіус основи і вистота), КУЛЯ (має додаткове поле типу double –радіус).

8. Створити базовий клас РІВНЯННЯ обчислення значення x. Створити похідні класи: клас ЛІНІЙНИХ РІВНЯНЬ (ax+b=0, додаються два дійсних значення - коефіцієнти a і b) і клас КВАДРАТНИХ РІВНЯНЬ ($cx^2+ax+b=0$, додаються три дійсних значення a, b, c).

9. Створити абстрактний клас ТРИКУТНИК. Трикутник задається двома сторонами та кутом між ними. Визначити похідні класи ПРЯМОКУТНИЙ, РІВНОБЕДРЕНИЙ трикутник. Додати методи для обчислення площі.

10. Створити клас геометричне ТІЛО, в якому визначено координати однієї з вершин геометричної фігури. Створити похідні класи ПІРАМІДА (має додаткове поле типу double – висота) та КУЛЯ (має додаткове поле типу double – радіус). Клас піраміда має похідні класи ТРИКУТНА (має додаткове поле типу double – площа основи) ТА ЧОТИРИКУТНА (в основі прямокутник, має додаткові поля типу double – довжини сторін).

Лабораторна робота 8
Розробка віконного інтерфейса. Робота у середовищі C++ Builder

8.1. Мета роботи

Навчитись програмувати в середовищі C++ Builder. Ознайомитися з такими об'єктами: форма (Form), текстове поле (Label), зображення (Image), кнопка (Button) та їхніми ос¬новними властивостями: підпис (Caption), колір (Color), шрифт (Font), видимість (Visible), ширина (Width), висота (Height) та ін.

8.2. Теоретичні відомості

Об'єкт **Form** використовують для створення програмою нового вікна. Розглянемо такі властивості форми:

Властивість	Опис властивості	Приклади значень
ActiveControl	Для задання активного об’єкта (фокуса) у формі	Button1, Edit2
AutoScroll	Наявність у формі смуг прокручування	True, False
BorderStyle	Можливість змінювати розміри вікна	bsSizeable – вікно з довільними розмірами, bsDialog, bsNone – вікно з фіксованими розмірами
Width, Height	Ширина і висота вікна у пікселях	503, 204 (числове значення)
Font	Шрифт	Комплексна властивість, задається в діалоговому вікні
HorizScrollBar VertScrollBar	Параметри смуг прокручування	Комплексна властивість
Icon	Задає піктограму, яка буде в заголовку форми під час виконання	None – стандартна піктограма для C++ Builder , або завантажена з певного файлу *.ico
Name	Ім’я форми	Form1(ідентифікатор)
Caption	Заголовок форми	Довільний рядок символів

Color	Колір фону форми	clGreen, clinfoBk (перелічений тип) чи $004525B1 (числове значення – задається в діалоговому вікні)
Cursor	Вигляд вказівника миші на формі під час виконання проекту	crDrag, crCross, crHelp, crArrow (перелічений тип)
Enabled	Доступність для дій об’єктів у формі під час виконання	True, False
Left, Top	Координати лівого верхнього кутка вікна у пікселях	200, 108 (числове значення)
Position	Розміщення і розміри вікна у момент запуску програми	poScreenCenter, poDesigned
WindowState	Стан вікна у момент запуску програми	wsNormal, wsMaximized, wsMinimized

Об єкт Label використовують для створення текстових полів (написів) у вікні програми. Крім аналогічних до наведених у вищезазначеній таблиці властивостей Width, Height, Font, Color, Name, Caption, Cursor, Enabled, Left, Top, він володіє ще й такими:

Властивість	Опис властивості	Приклади значень
Align	Вирівнювання поля відносно об’єкта, що його містить (форми)	alBottom, alClient, alLeft, alNone
Aligment	Вирівнювання тексту в межах поля	taCenter, taLeftJustify, taRightJustify
AutoSize	Приведення меж поля до границь тексту	True, False
Visible	Видимість об’єкта	True, False
WordWrap	Перенесення слів тексту у новий рядок	True, False

Об'єкт Image використовують для вставляння графічних об'єктів із файлів типу *.bmp, *.emf, *.ico, *.wmf у форму. Крім відомих властивостей

Align, Width, Height, Name, Cursor, Enabled, Left, Top, Visible, використовують такі:

Властивість	Опис властивості	Приклади значень
Center	Вирівнювання малюнка до центру відносно поля, що його містить	True, False
Picture	Ім’я графічного файлу	True, False
Stretch	Приведення розміру зображення до заданий розмірів об’єкта	Задається в діалоговому вікні
AutoSize	Приведення розмірів об’єкта до реальних розмірів зображення	True, False

Об'єкт Button використовують для створення кнопок на формі. Кнопки мають такі властивості: Visible, Width, Height, Font, Color, Name, Caption, Cursor, Enabled, Left, Top та ін.

Об'єкт Edit використовують для введення користувачем рядка символів із клавіатури. У разі необхідності для перетворення одержаного рядка (властивість Text) у число і навпаки використовують функції C++ Builder StrToFloat та FloatToStr. Окрім відомих вам властивостей, поля редагування Edit володіють такими:

Властивість	Опис властивості	Приклади значень
CharCase	Вигляд символів, які набиратимуться в полі редагування	ecNormal, ecUpperCase, ecLowerCase
PasswordChar	Символ для введення пароля	#0 – пряме відображення тексту, * - текст відображатиметься зірочками
ReadOnly	Можливість змінити текст (доступність поля)	True – текст не можна змінити, False
Hint	Текст підказки, яка висвітлюється, якщо навести курсор миші	Довільний рядок символів
ShowHint	Висвітлювати чи ні підказку	True, False

Text	Текст у полі редагування	Довільний рядок символів

Об'єкти RadioButton використовують для створення у формі засобу для вибирання однієї альтернативної можливості серед декількох. Розглянемо такі властивості перемикачів:

Властивість	Опис властивості	Приклади значень
Checked	Стан перемикача	True – вибраний, False – не вибраний
TabOrder	Порядок вибору клавішею Tab	0 – перший, 3 - четвертий
TabStop	Доступ до певного об’єкта табулятором	True, False

8.3 Програма роботи

8.3.1. Створити форму "Анкета студента" з даними про себе і двома фотографіями (портретною і художньою), які перекривають одна одну і мають з'являтися в результаті натискання на кнопки.

8.3.2. Створити форму з назвою "Обмін валюти", на якій можна змоделювати операції обміну валюти в обмін¬ному пункті. Застосувати поля редагування (Edit) та переми¬качі (RadioButton, дослівно радіокнопка), а також кнопки для виконання обчислень і закінчення роботи програми.

8.4. Обладнання та програмне забезпечення

8.4.1. Персональний комп’ютер.

8.4.2. Програмне забезпечення: C++ Builder (RAD Studio).

8.5. Порядок виконання роботи і опрацювання результатів

8.5.1. Завантажте середовище візуального програмування Borland C++ Builder.

Запуск системи візуального програмування C++ Builder виконують клацанням на піктограмі И або за допомогою каскадного меню Start (Пуск) => Programs (Програми) => Borland C++ Builder x.O => C++ Builder x.O, де x — версія програми. Отримаємо чотири вікна.

8.5.2. Дослідіть способи активізації чотирьох вікон C++ Builder:

- **головного вікна** C++ Builder x.O - Projectl, де є панель інструментів, палітра компонентів і головне меню;
- **вікна інспектора об'єктів Object Inspector** зі значеннями властивостей активного об'єкта;

- **вікна форми Forml,** в якому будуть розташовані результати роботи майбутньої програми;
- **вікна тексту програми** (Unitl.cpp).

Вікно тексту програми може частково перекриватися вікном форми. Активізувати вікна (а також змінювати їхні розміри чи розташування) можна за допомогою миші або використовуючи функціональні клавіші на клавіатурі:

F10 -для активізації головного меню (після цього натисніть на клавішу Esc);

F11 - для активізації вікна інспектора об'єкта;

F12 - для переходу між вікнами форми та коду програми.

8.5.3. Запустіть програму Projectl на виконання і розгляньте вікно порожньої поки що форми. Поекспериментуйте з вікном форми.

Запустити програму можна декількома способами:

- виконати команду Run => Run головного меню;
- клацнути на кнопці Run *Ш* панелі інструментів;
- натиснути на функціональну клавішу F9.

Виконайте такі дії: максимізуйте вікно, відновіть його попередній розмір, мінімізуйте та знову розгорніть вікно, пересуньте на робочому столі та змініть його розміри, викличте системне меню (Alt + пропуск). Виконайте ті самі дії за допомогою команд Move, Size та інших і клавіатури.

8.5.4. Закрийте вікно програми Forml, мінімізуйте головне вікно C++ Builder і створіть на робочому диску папку з іменем групи, а у ній власну папку, названу вашим прізвищем. Знову активізуйте C++ Builder.

8.5.5. Збережіть створену програму у своїй папці.

Для цього виберіть команду головного меню File => Save All (Зберегти Все) або натисніть на кнопку Save All на панелі інструментів. У першому рядку вікна, яке з'явиться ("Save Unitl As") під заголовком "Save in:" (Зберегти в:), за допомогою випадаючого меню виберіть ім'я робочого диска, після чого знайдіть і відкрийте свою власну папку. Задайте назву для файлу тексту програми, попередньо витерши запропоновану комп'ютером назву Unitl.cpp, => Save. У наступному вікні "Save Project 1 As" дайте назву файлові проекту, витерши запропоновану комп'ютером назву Projectl.bpr => Save. Зверніть увагу: файли проекту і тексту програми повинні мати різні назви.

8.5.6. Візуально ознайомтеся з властивостями форми Left, Top, Width та Height.

Перемістіть за допомогою миші форму Forml. Зверніть увагу, що зміна

розташування форми веде до зміни її властивостей Left та Top - координат лівого верхнього кута форми у вікні Object Inspector. Змініть розміри форми. Переконайтесь, що тепер змінюються властивості Width (ширина) та Height (висота) форми у вікні інспектора об'єкта.

8.5.7. Дослідіть, як зміна значень властивостей Left, Top, Width чи Height форми у вікні Object Inspector призводить до зміни розташування чи розміру форми.

Введіть відповідне значення у пікселях і натисніть на клавішу Enter.

8.5.8. Змініть колір фону форми.

Для цього у вікні властивостей форми Object Inspector у рядку Color виберіть значення кольору фону *двома* способами:

- викличте вікно вибору кольору подвійним клацанням мишею на поточному значенні властивості Color. Виберіть один із базових кольорів (Basic colors) або встановіть свій власний (Define Custom Colors) колір. Підтвердіть вибір (Ok).
- за допомогою випадаючого меню поекспериментуйте з різними значеннями властивості Color. Задайте початкове значення кольору — clBtnFace.

8.5.9. Виконайте програму ще раз.

8.5.10. Вставте у форму текстове поле (об'єкт типу Label або TLabel) з текстом "Анкета студента".

Двічі клацніть мишею на піктограмі Label на закладці Standard палітри компонентів головного вікна C++Builder. Розташуйте вставлений об'єкт, наприклад, переміщуючи його мишею. Якщо об'єкт Label невиокремлений, активізуйте його і у вікні Object Inspector змініть значення властивості Caption з Label 1 на текст "Анкета студента" без лапок. Змініть значення властивості Font (шрифт) цього текстового поля на такі:

Font : Times New Roman Cyr;
Font style : Bold;
Size :16;
Color : Purple.

У вікні Object Inspector відображається список властивостей лише активного у певний момент об'єкта.

8.5.11. Аналогічно вставте у форму ще декілька текстових полів з вашими біографічними даними. Для того, щоб надпис міг розміщуватися на декількох рядках необхідно встановити WordWrap = true і AutoSize = false.

8.5.12. Вставте у форму об'єкт типу Image (TImage) (зображення).

Для цього клацніть один раз лівою клавішею миші на піктограмі Image закладки Additional (додаткові) палітри компонентів і, наприклад, у

нижньому правому куті форми обведіть контур для майбутнього зображення (фотографії). Якщо потрібно, змініть розмір форми чи вставленого об'єкта та добийтеся якнайкращого розташування на ній створених раніше об'єктів. Змінювати розміри об'єкта можна методом їх "розтягування" за маркери (чорні габаритні квадратики). Запам'ятайте назву, яку Builder присвоїть цьому об'єкту (значення властивості Name) або замініть її на свій розсуд. За замовчуванням цей об'єкт матиме стандартну назву Image.

8.5.13. Вставте свою портретну фотографію за допомогою властивості Picture (ілюстрація) об'єкта Image1.

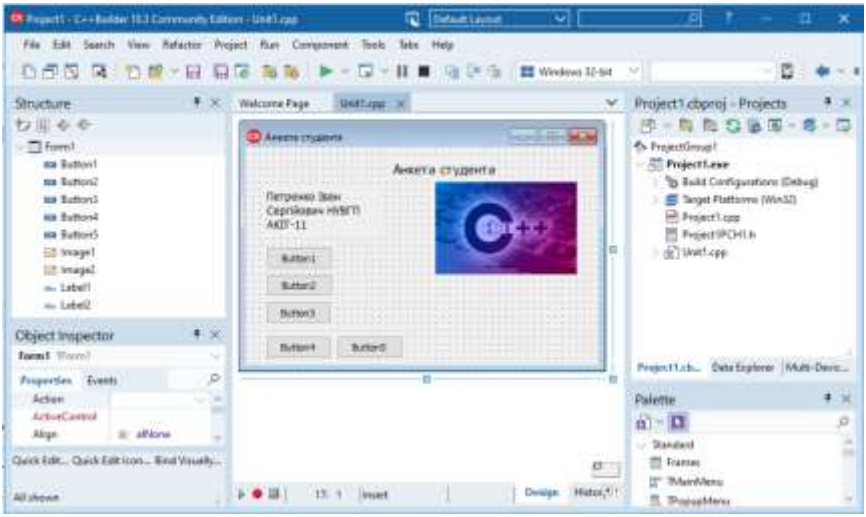


Рис. 8.1. Створення форми

Для цього виокремте об'єкт Image та активізуйте рядок Picture у вікні Object Inspector. Клацнувши на кнопці '…', викличте діалогове вікно вибору малюнка Picture Editor. Клацніть на кнопці Load (завантажити) і у вікні Load picture зазначте шлях до файлу з фотографією. *Якщо такого файлу немає,* скористайтесь будь-якою фотографією з дефолтної бібліотеки. Виберіть будь-який файл => Open. Підтвердіть свій вибір у вікні Picture Editor натисканням на клавішу Ok. Задайте властвість Stretch для об'єкта Image як True.

8.5.14. Вставте свою художню фотографію у форму поверх наявної, скориставшись ще одним об'єктом типу Image.

Один із варіантів розташування фотографії показаний на рис. 8.3. Вважатимемо, що цей об'єкт має назву Image2.

Під час накладання об'єктів може виникнути потреба використати команди Send To Back (переслати назад) чи Bring To Front (перенести наперед), які є в їх контекстових меню чи в головному меню Edit.

8.5.15. Поекспериментуйте з властивістю Visible (видимість) обох зображень, кожного разу виконуючи програму (див. пункт 8.5.3). Після цього виберіть значення властивості Visible у False для обох зображень.

8.5.16. Вставте у форму кнопки для засвічування фотографій — два об'єкти типу Button з назвами Button1 і Button2. Піктограма об'єкта типу Button (кнопка) знаходиться на закладці Standard палітри компонентів головного вікна C++ Builder. Поміняйте підписи на кнопках (змініть властивість Caption) на "Портретна фотографія" та "Художня фотографія" відповідно. Виберіть найкращий, на ваш розсуд, кирилізований шрифт для підписів. Якщо використано картинки із стандартної бібліотеки Borland, виберіть для кнопок цікаві підписи.

8.5.17. Запрограмуйте кнопку "Портретна фотографія" так, щоб після її натискання у формі з'являлась портретна фотографія.

Для програмування кнопки Button1 необхідно двічі клацнути на ній лівою клавішею миші. У результаті активізується вікно тексту програми із заготовкою функції Button1Click, яка опрацьовуватиме подію клацання на кнопці Button1:

```
void __fastcall TForm1::Label2Click(TObject *Sender)
    {

    }
```

У заготовку необхідно вставити текст програми реакції на цю подію. Процедура матиме такий вигляд:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
    {
     Image1->Visible = true;//  Портретна фотографія стає видимою
     Image2->Visible = false;// Художня фотографія стає невидимою
    }
```

За допомогою цієї функції властивість видимості для об'єкта Image1 задаємо і цю ж властивість для об'єкта Image2 забираємо. Для кнопки "Художня фотографія" дії будуть протилежні. Зверніть увагу на використання складених імен, наприклад Image1 .Visible, в яких назва об'єкта від його властивості відокремлюється крапкою. Такі складені імена дають доступ до значення конкретної властивості деякого об'єкта.

8.5.18. Запрограмуйте кнопку "Художня фотографія" відповідно до її

призначення (див. п. 8.5.17).

Текст функції для цієї кнопки матиме такий вигляд:

```
void __fastcall TForm1::Button2Click(TObject *Sender)
  {
   Image1->Visible = false;//Портретна фотографія стає видимою
   Image2->Visible = true; // Художня фотографія стає невидимою
  }
```

Щоб створити таку функцію швидко, можна скопіювати дві команди присвоєння з попередньої функції у нову і поміняти вирази праворуч.

8.5.19. Виконайте програму і переконайтесь, що кнопки виконують всі функції. Закрийте вікно програми "Анкета студента".

8.5.20. Збережіть створену програму у своїй папці.

Виберіть елемент головного меню File => Save All (Зберегти все) або натисніть на кнопку Save All на панелі інструментів.

8.5.21. Створіть exe-файл програми.

Виконайте команду головного меню Project => Make Project1 (Сконструювати все).

8.5.22. Закрийте C++ Builder, виконайте створену програму і проекспериментуйте з побудованими кнопками.

Запустіть exe-файл з іменем проекту і піктограмою зі своєї власної папки.

8.5.23.Вставте у форму третю фотографію (фото вашого будинку чи машини) і ще одну кнопку з відповідним підписом, яка її висвітлюватиме. Якщо файлу з такою фотографією немає, скористайтесь будь-яким файлом з бібліотеки Ibcolor (див. п. 8.5.13).

8.5.24. Поміняйте підписи до кнопок на такі: "Змінити фотографію" та "Забрати фотографію", перепрограмувавши кнопки відповідно до нового призначення. Запишіть фрагменти зміненого програмного коду у звіт. Виконайте програму і переконайтесь у правильності її роботи.

У тексті функцій, що описують роботу кнопок, можна скористатися такими командами:

if (Image1->Visible = True) ...// *Якщо видимість = True*

або рівносильною командою

if (Image1.Visible)... //*Тут умова істинна, якщо видимість увімкнена*

8.5.25. Поміняйте сценарій роботи програми з п. 8.5.2 на такий:

- відразу після запуску програми фотографій не видно, є дві кнопки "Портретна фотографія" і "Забрати фотографію" , доступною є лише перша кнопка;
- після клацання на кнопці "Портретна фотографія" у формі

з'являється портретне фото, підпис на першій кнопці змінюється на "Художня фотографія", стає доступною кнопка "Забрати фотографію";

- після клацання на кнопці "Художня фотографія" фотографія у формі змінюється на художню, а підпис на цій кнопці змінюється на "Третя фотографія";
- після клацання на кнопці "Третя фотографія" фотографія у формі змінюється на третю, а підпис на цій кнопці змінюється на "Портретна фотографія";
- після клацання на кнопці "Забрати фотографію" фотографія зникає і ця кнопка стає недоступною.
- Запишіть фрагменти програмного коду у звіт. Виконайте програму і переконайтесь у правильності її роботи.

У тексті функцій, які описують роботу кнопок, можна скористатися командами, що змінюють властивості кнопок Caption (підпис), Visible (видимість), Enabled (доступність).

8.5.26. В умовах задачі п. 8.5.3 після клацання на кнопці "Забрати фотографію" ця кнопка стає не лише недоступною, але і невидимою.

8.5.27. Змініть програмний код розв'язування задачі з п. 8.5.4 так, щоб після вимкнення фотографій напис на першій кнопці завжди відповідав фотографії, яка повинна з'явитися після її натискання.

8.5.28. Виходячи з умови задачі з п. 8.5.5, добийтеся того, щоб послідовність перемикання фотографій не порушувалася внаслідок їх вимкнення, а також додайте текстовий підпис з назвою фотографії, видимою у поточний момент.

8.5.29. Продемонструйте створену форму викладачеві. Закінчіть роботу.

8.5.30. Створіть програму «Обмін валют». Для цього Завантажте середовище візуального програмування C++ Builder.

8.2.31. Відмовтесь від можливості змінювати розміри вікна програми, надавши властивості форми BorderStyle значення bsDialog.

Задавши це значення, виконайте програму і переконайтеся, що не можна змінити розмір форми. Зверніть увагу на відсутність у вікні кнопок мінімізації і максимізації, а також системного меню. Завершіть роботу програми.

8.5.32. Вставте у форму два об'єкти типу RadioButton (перемикачі).

8.5.33. Для цього клацніть на піктограмі об'єкта типу RadioButton (перемикач), яка розміщена на закладці Standard палітри компонентів головного вікна C++ Builder, а після цього клацніть на потрібному місці на формі. Повторіть ці дії, щоб вставити другу радіокнопку.

8.5.34. Задайте початкове значення другого перемикача як активне.

Для цього клацніть на правому перемикачі і значення його властивості

Checked (контроль вибору) задайте як True.

8.5.35. Вставте у форму два поля редагування — об'єкти Edit1 та Edit2. Для цього клацніть на піктограмі об'єкта типу Edit (редагування), яка знаходиться на закладці Standard палітри компонентів головного вікна C++ Builder, а потім клацніть у потрібному місці на формі. Вставте другий об'єкт (рис. 8.2). Запустіть програму і поекспериментуйте зі вставленими об'єктами: клацніть у полі редагування, введіть деяке число, вилучіть його. Закрийте вікно програми.

8.5.36. Розташуйте у формі два текстові поля — об'єкти Label1 та Label2 (див. рис.8.2).

8.5.37. Вставте у форму два поля редагування — об'єкти Edit3 та Edit4 (див. рис. 8.2).

8.5.38. Вставте у форму дві кнопки - об'єкти типу Button (див. рис. 8.2).

8.5.39. Вставте у форму ще два текстові поля - об'єкти Label3 і Label4 (див. рис. 8.2).

8.5.40. Збережіть створену у цей момент форму у своїй папці.

File => Save All. Файли тексту програми та проекту назвіть різними іменами. Імена занотуйте у звіт. У подальшому періодично, зокрема перед черговими запусками проекту на виконання, зберігайте файли програми (File => Save All, вводити імена файлів вже не потрібно).

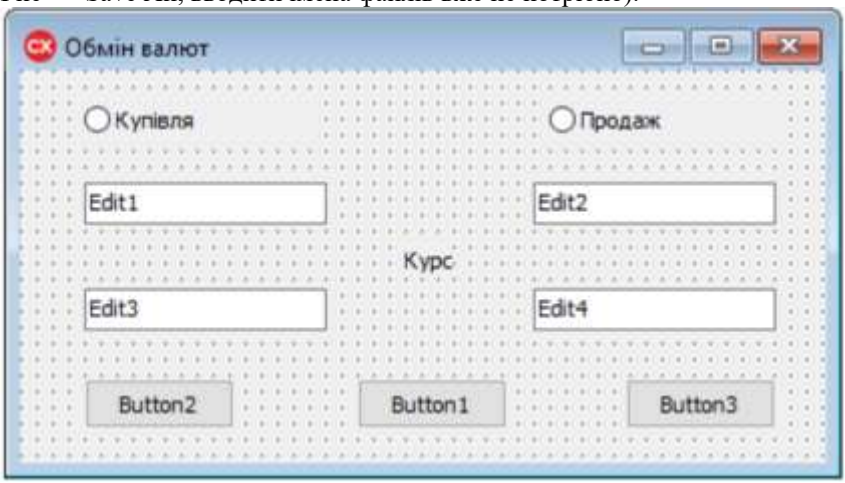


Рис. 8.2. Об’єкти на формі

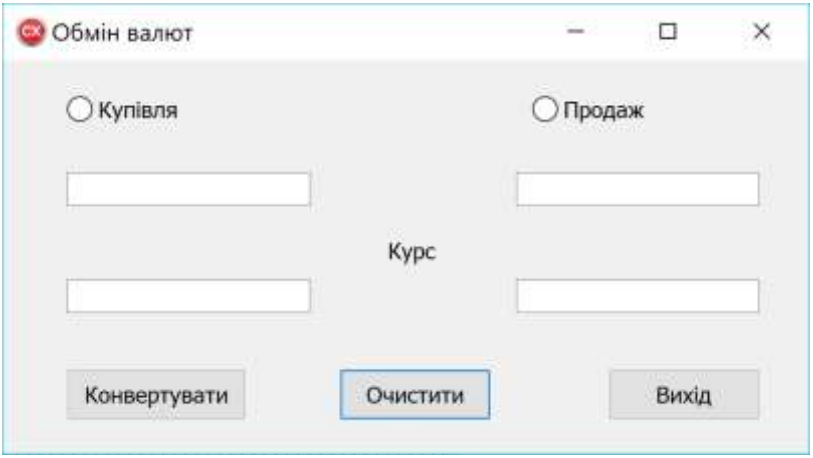


Рис. 8.3. Готова форма

8.5.41. Змініть назву форми з "Form1" на "Обмін валюти".

Для цього змініть значення властивості Caption форми. Клацніть на формі і на рядку Caption у вікні Object Inspector. Введіть назву форми без лапок. Зверніть увагу на те, що для об'єктів багатьох типів (зокрема, Label, Button, Form, RadioButton, CheckBox та інших) значення властивостей Caption та Name збігаються.

8.5.42. Змініть підписи Caption на об'єктах типу RadioButton, Label та Button так, як показано на рис. 8.3.

Для цього по черзі вибирайте об'єкти (клацайте на них) і змінюйте значення властивості Caption.

8.5.43. Задайте однакові розміри для всіх текстових полів, полів редагування та кнопок і вирівняйте їх на формі.

Для цього одночасно виокремте п'ять об'єктів лівого стовпця одним із способів:

- тримаючи натиснутою клавішу Shift, почергово активізуйте об'єкти, клацаючи на них лівою клавішею миші;
- обведіть навколо цих об'єктів контур, утримуючи натиснутою ліву клавішу миші.

У вікні Object Inspector задайте спільні для цих об'єктів значення властивостей Width (ширина), Height (висота) та Left (відступ від лівої межі вікна) у пікселях. *Зверніть увагу, що після активізації відповідної властивості її значення – це значення властивості першого виокремленого об'єкта створеної групи.* Можете змінити стиль, колір чи розмір шрифта одночасно для усіх виокремлених об'єктів (властивість Font). Зніміть

виокремлення, клацнувши на вільному місці форми. Аналогічно виконайте вирівнювання правого стовпця об'єктів. Вирівняйте вставлені поля попарно у горизонтальному напрямку. Для цього змініть властивість Top (відступ від верхньої межі вікна у пікселях) для відповідних груп об'єктів. Збережіть роботу (Save All).

8.5.44. Задайте значення курсів купівлі та продажу валюти, а також кількість валюти, яку кантор купує чи продає.

Для цього введіть потрібне число, наприклад 5, як значення властивості Text об'єкта Edit1. Повторіть це для об'єкта Edit2 (значення 5.2) та Edit3 (значення 20). Для набору символа десяткової крапки використайте символ, передбачений операційною системою комп'ютера.

8.5.45.Очистіть поле редагування Edit4.

Для цього вилучіть значення властивості Text для об'єкта Edit4. Не сплутайте значення властивостей Name та Text цих об'єктів.

8.5.46. Заблокуйте можливість введення даних для поля Edit4, задавши його властивість Readonly як True, оскільки це поле міститиме результат.

Змінювати значення певної властивості можна подвійним клацанням на ній лівою клавішею миші. Збережіть роботу (Save All). Виконайте програму і переконайтеся, що не можна ввести чи редагувати дані у полі Edit4.

8.5.47. Запрограмуйте радіокнопки так, щоб напрямок стрілки показував на вид операції: купівля чи продаж. Зробіть активним поле Edit3.

Клацніть *двічі* на правому перемикачі RadioButton2 (Продаж). Отримаєте заготовку функції RadioButton2Click. У тілі цієї функції опишіть дії, які мають відбутися у результаті клацання на правому перемикачі RadioButton2:

```
void   fastcall TForm1 ::RadioButton2Click(TObject "Sender)
{
Label3->Caption = "=>"; // Змінюємо напрямок стрілки
//Активізуємо поле Edit3
Edit3 -> SetFocus()
```

Аналогічно запрограмуйте подію Click клацанням на лівому перемикачі RadioButtoni, врахувавши, що стрілка має показувати на ліве поле ('<='). *Фрагмент програмного коду створеної функції запишіть у звіт.*

8.5.48. Запустіть програму і переконайтесь, що перемикач виконує свої функції згідно з п. 8.5.47.

8.5.49. Запрограмуйте кнопку "Вихід".

Скористайтесь функцією закінчення роботи програми exit():

```
void    fastcall TForm1 ::Button2Click(TObject 'Sender)
{
exit(1);                    // Закінчуємо роботу програми
}
```

8.5.50. Запрограмуйте кнопку "Обчислити".

Дане у полі редагування - це значення властивості Text об'єкта типу рядок. Для перетворення цього даного у числовий дійсний тип (float) скористайтесь функцією StrToFloat(), а навпаки - функцією FloatToStr(). Опишіть основні (kurs, suma) і додаткові (cod, ed) змінні.

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
 float f1 = StrToFloat(Edit1 ->Text);  //Одержуємо значення
курсу купівлі
 float f2 = StrToFloat(Edit2 ->Text);  // Одержуємо значення
курсу продажу
float f3 = StrToFloat(Edit3 ->Text);          // Одержуємо
числове значення суми в USD
 float f4;                                  //Змінна для суми в
гривнях
 if (RadioButton1 -> Cheked) f4 = f1* f3;
else  f4 = f2*f3;
Edit4  -> Text = FloatToStr(f4);
// Одержане число суми в гривнях перетворюємо у текстовий
формат і результат присвоюємо властивості Text поля Edit4
 }
```

8.5.51. Збережіть роботу (Save All).

8.5.52. Виконайте програму і проекспериментуйте з різними грошовими сумами й операціями купівлі чи продажу. Закрийте вікно програми "Обмін валюти".

Для переривання роботи програми у випадку неправильного введення вхідних даних виконайте пункт головного меню Run => Program Reset.

8.5.53. Змініть розміри та кольори символів (зокрема об'єктів Label3 і Edit3), розташування об'єктів, фон форми (властивість Color) так, щоб форма виглядала якнайкраще.

8.5.54. Забезпечте появу підказки "Введіть суму в доларах" після переміщення вказівника миші до поля Edit3.

Виберіть об'єкт Edit3 і встановіть властивість ShowHint у True, а як значення властивості Hint введіть текст підказки. Збережіть роботу, запустіть програму і переконайтеся, що підказка з'являється.

8.5.55. Поміняйте вигляд стрілки з => на =>, а <= на <=.

Для цього виберіть об'єкт Label3 і як значення властивості Caption введіть українську букву р, після чого, активізувавши властивість Font, виберіть назву шрифта Wingdings. Двічі клацніть на правому перемикачі і в його функції введіть українську букву р замість =>. У функції для лівого перемикача символи <= замініть буквою п. Збережіть роботу, запустіть програму і переконайтеся, що стрілка змінила свій вигляд.

8.5.56. Створіть exe-файл вашої програми.

Виконайте пункт головного меню Project => Make Project.

8.5.57. Закрийте C++ Builder, запустіть створену програму і виконайте обчислення для різних початкових даних.

Запустіть exe-файл з іменем проекту і відповідною піктограмою зі своєї папки.

Зауваження 1. Зверніть увагу на використання коми чи крапки у вхідних даних. У числах, які стосуються курсу валюти, гривневої чи доларової сум для десяткової крапки використайте символ, передбачений операційною системою вашого
комп'ютера (див. Start (Пуск) => Settings (Налаштовування) => Control Panel (Панель керування) => Regional Settings (Місцеві параметри) =-. закладка Number (Числа), рядок Decimal symbol (Символ десяткової крапки)).

8.5.58. Передбачте у створеній програмі ще одну кнопку для очищення полів грошових сум. Виконайте програму і переконайтесь у правильності її роботи.

Підказка. Для об'єктів Edit3, Edit4 у функції опрацювання події натискання на цю кнопку використайте команду присвоєння їхнім властивостям Text порожнього рядка ('').

8.5.59. Забезпечте появу підказки "Введіть курс купівлі" та "Введіть курс продажу" після переміщення вказівника миші до полів Edit1 та Edit2 відповідно (див. п. 8.5.54).

8.5.60. Модифікуйте програму, передбачивши додаткову можливість зміни типу операцій (купівля, продаж) унаслідок клацання мишею на стрілці. Запишіть у звіт фрагмент програмного коду, який реалізує цю можливість. Виконайте програму.

Підказка. Для цього двічі клацніть на текстовому полі стрілки. Відкриється вікно програмного коду із заготовкою функції Label3Click (опис дій у випадку клацання на об'єкті Label3). У тілі цієї функцій можна скористатися командами вигляду:

if (RadioButton1.Checked=True)
// Встановлюємо перемикач у праве положення, змінюючи
// значення властивості Checked об'єкта RadioButton2
else
//Встановлюємо перемикач у ліве положення, змінюючи
//значення властивості Checked об'єкта RadioButtoni

8.5.61. Спростіть форму (вилучіть зайві об'єкти) та змініть код кнопки "Обчислити" так, щоб її можна було використати для переведення миль у кілометри чи навпаки залежно від положення перемикача (1 миля =1,609344 км).

8.5.62. У створену для задачі у п. 8.5.31. форму вставте групу перемикачів (об'єкт RadioGroup) для вибору типу милі з двох можливих значень: морської чи звичайної (1 морська миля = 1,852 км).

Підказка. Для задання підписів до перемикачів використайте властивість Items групи перемикачів RadioGroup, а для контролю вибору певного перемикача скористайтесь властивістю Itemlndex (дорівнює: 1, якщо жоден не вибрано, 0 - якщо вибрано перший перемикач групи, 1 - якщо другий і т. д.)

8.6 Контрольні запитання

8.6.1. З чого складається середовище програмування C++ Builder?
8.6.2. Як створити новий проект в середовищі C++ Builder?
8.6.3. Які можливі типи програм в середовищі C++ Builder?
8.6.4. Які елементи можна розміщувати на формі проекту?

Лабораторна робота 9
Програмування циклів. Об'єкти: Memo, MainMenu, PopupMenu, CheckBox, GroupBox

8.1. Мета роботи
Набути навиків роботи з об’єктомтипу MainMenu) з та його командами.

8.2. Теоретичні відомості
8.2.1.Об'єкти Memo, MainMenu, PopupMenu, CheckBox, GroupBox

Об'єкт Мемо застосовують для створення багаторядкового редактора тексту. Крім звичайних властивостей, поле редагування Мемо володіє ще такими:

Властивість	Опис властивості	Приклади значень
HideSelection	Збереження виокремлення фрагмента тексту у момент втрати фокусу	True – виокремлення не зберігається False – зберігається
Lines	Задання початкового тексту у полі редагування	Комплексна властивість (задається у діалоговому вікні)
MaxLength	Максимальна можлива кількість уведених символів	Наприклад, 50 – п’ятдесят символів, 0 – без обмежень
ScrollBars	Наявність смуг прокручування	ssNone – відсутні ssHorizontal – горизонтальна ssBoth – обидві.

Об'єкт CheckBox використовують для створення незалежного дво- чи трипозиційного прапорця: увімкнено/вимкнено (/недоступний). Для цього об'єкта визначені такі дві нові властивості:

Властивість	Опис властивості	Приклади значень
AllowGrayed	Наявність третьої позиції (сірий увімкнений)	True – трипозиційний прапорець, False – двопозиційний прапорець
State	Стан прапорця	cbGrayed – сірий, cbUnchecked – вимкне-ний, cbChecked – увімкнений

Панель групи об'єктів GroupBox призначена для розміщення на ній групи із кількох об'єктів. Панель групи використовують для покращення дизайну вікна програми. Властивості цього об'єкта аналогічні до описаних вище.

За допомогою об'єкта MainMenu створюють головне меню програми. Ось деякі властивості головного меню:

Властивість	Опис властивості	Приклади значень
Items	Команди меню	Комплексна властивість (задається у діалоговому вікні)
Tag	Допоміжна змінна, використовується в тексті програми	0; 8 (ціле число)

За допомогою об'єкта PopupMenu створюють контекстове меню деякого компонента. Для "прив'язування" контекстового меню до конкретного об'єкта необхідно його властивості Popup-Menu надати значення імені (Name) конкретного контекстового меню. Розглянемо деякі властивості контекстового меню:

Властивість	Опис властивості	Приклади значень
Alignment	Ворівнювання меню відносно точки клацання правою клавішею миші	paCenter – по центру, paLeft – зліва, paRight – справа
AutoPopup	Автоматичний виклик контекстного меню	True – викликається клацанням правою клавішею, False – викликається за допомогою методу Popup

Конкретна команда меню (головного чи контекстового) може мати такі властивості:

Властивість	Опис властивості	Приклади значень
Break	Розбиття меню у горизонтальному напрямку	mbNone – без розбиття, mbBarBreak – розбиття з вертикальною рискою, mbBreak –розбиття без вертикальної риски

ShortCut	Комбінація «гарячих» клавіш для виклику команди меню	None –відсутня, CTRL+A, F8, CTRL+F10, Shift+F3

Таймер (Timer, піктограма на закладці System) використовують для повторення фрагмента коду програми з певною періодичністю. Відповідний фрагмент розташовують у тілі функції опрацювання події OnTimer таймера. Періодичність вмикання таймера у мілісекундах задають властивістю Interval. Геометрична фігура (Shape, піктограма на закладці Additional) призначена для зображення елементарних геометричних фігур і має, зокрема, такі властивості:

Властивість	Опис властивості	Приклади значень
Brush	Характеристики кольору (Color) і стилю (Style) заливки	Brush-Color: clMaroon Brush-Style: bsSolid, bsVertical
Shape	Форма фігури	stRoundRect – прямокутник зі округленими краями stEllipse, stSquare
Pen	Характеристики границі фігури	Комплексна властивість

Мультимедійний програвач (MediaPlayer, піктограма на закладці System) призначений для програвання відео- та аудіофайлів. Керування програвачем може здійснюватися як за допомогою традиційних кнопок Play, Pause, Stop, Next тощо на етапі виконання програми, так і з програмного коду шляхом виконання методів цього об'єкта, наприклад:

```
MediaPlayer1->FileName="повнее ім'я відео- чи аудіофайлу";
MediaPlayer1 ->Open();
MediaPlayer1 ->Play();
```

Спарена кнопка з полем редагування (CSpinEdit, піктограма на закладці Samples) призначена для введення та корекції цілочислового значення деякої величини під час роботи програми. Властивості кнопки:

Властивість	Опис властивості	Приклади значень
EditirEnabled	Можливість прямого редагуваня	True, False

Enabled	Можливість будь-якого редагування	True, False
Increment	Крок зміни	1; 4 (ціле число)
MaxValue	Максимальне значення	Ціле число
MinValue	Мінімальне значення	Ціле число

Індикатор стану (CGauge, піктограма *W* на закладці Samples, ProgressBar, піктограма "" на закладці Win32) використовують для наочної демонстрації стану виконання деякого процесу. Розглянемо три властивості індикатора CGauge:

Властивість	Опис властивості	Приклади значень
Kind	Тип індикатора	gkHorizontalBar – горизонтальний рядок, gkPie – кругова діаграма gkText – процентне відображення
Progress	Відображає стан індикатора	Ціле число між MaxValue і MinValue
ShoeText	Додатково відображає стан у процентах	True, False

9.3 Програма роботи

9.3.1. Запустити середовище програмування Dev-C++.

9.3.2. Написати програму для побудови графіка функції згідно свого варіанту (лабораторна 1 додаток 1).

9.4. Обладнання та програмне забезпечення

9.4.1.Персональний комп'ютер.

9.4.2. Програмне забезпечення: C++ Builder (RAD Studio).

9.5. Порядок виконання роботи і опрацювання результатів

9.5.1. Завантажте середовище візуального програмування C++ Builder.

9.5.2. Змініть заголовок (Caption) форми з "Form1" на "Табулювання функції" (без лапок) і збільшіть розміри форми у вертикальному напрямку.

9.5.3. Змініть піктограму у лівому верхньому куті форми, задавши конкретний файл з рисунком піктограми як значення властивості Icon (піктограма) форми.

9.5.4. Клацніть у рядку Icon на ""і, а потім на кнопці Load вікна Picture Editor, щоб отримати вікно Load picture. Відкрийте папку C: \ Program Files \ Common Files \ Borland Shared \ Images \ Icons, виберіть графічний файл з будь-якою піктограмою => Open => Ok.

9.5.5.Збережіть виконану на даний момент форму у своїй власній папці (File => Save All).

9.5.6.Розташуйте у формі поля редагування Editi, Edit2, Edit3 і відповідні їм текстові поля "Ліва межа", "Права межа", "Крок", а також текстове поле для вигляду заданої функції y = sin(x)+1 (див. рис. 9.1).

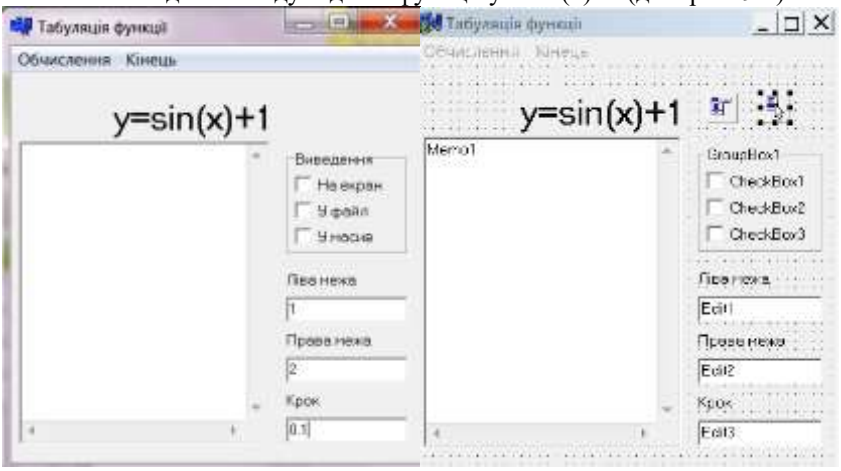


Рис. 9.1. Готова форма Рис. 9.2. Об’єкти на формі

Для того, щоб швидко вставити у форму декілька однотипних об'єктів, клацніть на піктограмі цього об'єкта, утримуючи натиснутою клавішу Shift. Тепер вставляння у форму всіх об'єктів цього типу відбуватиметься без повторного вибору піктограми. *Якщо випадково вставите зайвий об'єкт, то вилучіть його за допомогою клавіші Delete.* Щоб відмовитися від такого режиму, клацніть на зображенні стрілки на палітрі компонентів. Розмір, стиль і колір шрифтів виберіть на власний розсуд так, щоб форма виглядала якнайкраще. Save all.

9.5.7. Вирівняйте вставлені поля редагування до лівого краю першого об'єкта та відцентруйте текстове поле вигляду функції, скориставшись вікном вирівнювань Alignment.

Виокремте групу полів редагування та підписи до них і виконайте команду головного меню Edit (редагувати) => Align (вирівняти) => Horisontal - Left sides (горизонтально - ліві межі) => Vertical - Space equally (вертикально - рівномірно) => Ok. Виокремте текстове поле вигляду функції Edit => Align => Horisontal - Center in Window (горизонтально — до центру вікна) => Ok. Save All.

9.5.8. Вставте у форму панель групи об'єктів (об'єкт типу GroupBox). Для цього використайте компоненту GroupBox із закладки Standard.

Змініть значення властивості Caption (підпис) цього об'єкта на слово "Виведення" (без лапок). Розмір, стиль і колір шрифту виберіть на власний розсуд. Збільшіть панель групи.

9.5.9.Вставте у панель три прапорці (об'єкти типу CheckBox).

Для цього використайте компоненту CheckBox із закладки Standard палітри компонентів C++Builder. Вирівняйте прапорці, заздалегідь виокремивши їх (див. п. 9.5.7). Змініть значення властивості Caption (підпис) цих об'єктів на такі, як показано на рис. 9.1. Стиль і колір шрифтів виберіть на власний розсуд.

9.5.10.Встановіть прапорці "На екран" та "У масив" у положення "увімкнено".

Для цього виокремте ці об'єкти та змініть значення їхніх властивостей Checked (контроль вибору) на True.

9.5.11. Вставте у форму багаторядкове поле редагування (об'єкт Memo). Для цього використайте компоненту Memo із закладки Standard палітри компонентів C++Builder. Збільшіть розміри поля. Властивість ScrollBars (наявність смуг прокручування) цього об'єкта задайте як ssBoth (будуть обидві смуги — вертикальна і горизонтальна). File => Save All.

9.5.12. Задайте початкові значення для полів редагування лівої і правої меж аргумента функції та для кроку зміни цього аргумента, наприклад, такі, як на рис. 9.1. Для цього змініть властивість Text цих обєктів. Для набору символа десяткової крапки використайте символ, передбачений операційною системою комп'ютера.

9.5.13. Витріть слово Memo1 у багаторядковому полі редагування (див. пояснення нижче).

Для цього у вікні Object Inspector змініть значення властивості Lines (рядки) об'єкта Memol. Натиснувши на кнопку ""i, викличте вікно редагування цієї властивості (вікно String list editor - редактор багаторядкового поля). Витріть слово Memol та закінчіть роботу з цим вікном, клацнувши на кнопці Ok.

9.5.14. Вставте у форму головне і контекстове меню (об'єкти типу MainMenu і PopupMenu). Для цього використайте компоненти MainMenu і PopupMenu із закладки Standard. Розташуйте піктограми в довільному місці форми, на етапі виконання програми вони будуть невидимими.

9.5.15. Введіть назви команд головного меню форми (див. рис. 9.3). Для цього виберіть об'єкт MainMenui і двічі клацніть на значенні його властивості Items. Інший шлях - двічі клацніть на самому об'єкті. У вікні, яке відкриється (Form1->MainMenu1), вибирайте мишею рамку команди і записуйте назву команди, наприклад, "Обчислення", як значення властивості Caption у вікні Object Ispector. Закрийте вікно створення команд головного меню Form1 - >MainMenu1.

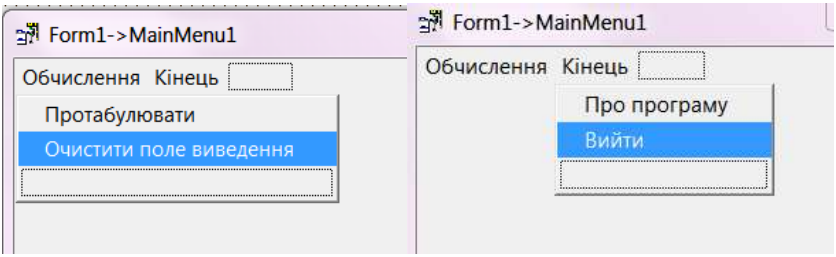


Рис. 9.3 Створення головного меню

9.5.16. Введіть назви команд контекстового меню форми. Для цього аналогічно змініть значення властивості Items об'єкта PopupMenui за допомогою вікна Forml.PopupMenul (виклик цього вікна див. у п. 9.5.15). У вікні Object Ispector введіть текст "Очистити поле виведення" без лапок як значення властивості Caption (рис.9.4). За бажанням можете придумати ще якусь команду. Закрийте вікно Forml.PopupMenul. Збережіть форму (Save All).

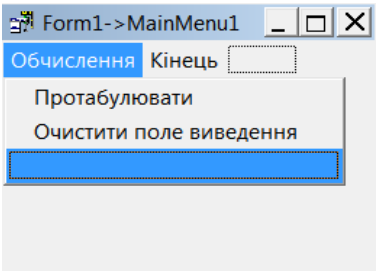


Рис. 9.4. Створення контекстного меню

9.5.17. Запрограмуйте команду "Очистити поле виведення" головного меню, скориставшись методом Clear об'єкта Memo.

Методи об'єкта — це набір функцій, які, аналогічно до властивостей, застосовують до об'єкта. Як і у випадку властивості, ім'я об'єкта від імені його метода відокремлюється крапкою. Виконайте команду меню "Очистити поле виведення", не запускаючи програму на виконання. З'явиться заготовка функції реакції на подію виклику цієї команди. У ній запишіть команду виклику методу Clear для очищення поля виведення об'єкта Memo1:

```
void __fastcall TForm1::N4Click(TObject *Sender)
{                // "Очистити поле виведення", у вас може
                 //  може бути інший його номер
Memo1->Clear(); //  не виправляйте
                 //  Викликаємо метод об'єкта Memo1, який
                 //  очищає багаторядкове поле редагування
                 //  Тепер клацніть на формі
}
```

9.5.18. Запрограмуйте команду "Вийти" головного меню, скориставшись стандартною функцією exit().

```
void __fastcall TForm1::N6Click(TObject *Sender)
{        // "Вийти"
exit(1); //  Закриваємо вікно програми
         //  Тепер клацніть на формі
}
```

Збережіть виконану на даний момент форму у своїй папці (File => Save All).

9.5.19. Запрограмуйте команду "Очистити поле виведення" контекстового меню.

Двічі клацніть на команді контекстового меню "Очистити поле виведення" у вікні Forml.PopupMenu1 (виклик цього вікна див. у п. 9.5.15). Текст функції очищення такий самий, як і у команди головного меню, а саме: Memo1 -> Clear(); (див. п. 9.5.17).

9.3.20. "Прив'яжіть" контекстове меню PopupMenu1 до форми Form1. *Клацніть на формі* і задайте властивість *форми* PopupMenu як PopupMenu1.

Значення властивості форми Menu автоматично встановлюється як MainMenu1 у момент створення головного меню (п. 9.5.14).

9.5.21. Запустіть створену програму та дослідіть її роботу.

Поекспериментуйте з багаторядковим полем редагування Memo1, вводячи та коректуючи у ньому будь-який текст. Зверніть увагу на те, що в цьому вікні можна виконувати такі ж дії з текстом, як і в текстовому редакторі: виокремлювати фрагмент тексту, копіювати, переносити чи вилучати цей фрагмент. Витріть текст за допомогою команди головного меню "Очистити поле виведення". Ще раз введіть текст і витріть його за допомогою контекстового меню (для викликуконтекстового меню форми потрібно клацнути правою клавішею миші на вільному місці форми). Закінчіть роботу програми, клацнувши на команді меню "Вийти".

Додайте бібліотеку #include <math.h> для можливості використання математичних функцій, в даному випадку sin(x).

9.5.23. Запрограмуйте команду "Протабулювати".

Властивість Lines об'єкта типу Мето є комплексною, тобто також є об'єктом зі своїми властивостями та методами. Результат роботи цієї програми - це таблиця, що складається з декількох рядків. Щоб долучити (додати) у поле Memo1 новий рядок до таблиці, треба змінити значення комплексної властивості Lines (рядки) за допомогою її методу Append (вставити) з одним аргументом - символьним рядком: Memol->lines->Append (рядок символів). Виконайте команду "Протабулювати" з головного меню, клацнувши на ній один раз. З'явиться заготовка функції, яку заповніть так:

```
void __fastcall TForm1::N3Click(TObject *Sender)
{
float x,y;
float a=StrToFloat(Edit1->Text); // Одержуємо числове значення лівої межі
float b=StrToFloat(Edit2->Text); // Одержуємо числове значення правої межі
float h=StrToFloat(Edit3->Text); // Одержуємо числове значення кроку
Memo1->Lines->Append("X\t\Y");   /* В об'єкт Memo1 вставляємо рядок з підписами
стовпців аргументу X і значення функції Y через табулятор \t. Починаємо
табулювати з лівої межі, поки аргумент x не перевищить праву межу з певним
допуском*/

for(x=a;x<b+h/2;x+=h)
{
y=sin(x)+1;
if (CheckBox1->Checked)
Memo1->Lines->Append(FloatToStrF(x,ffFixed,2,2)+'\t'+
                     FloatToStrF(y,ffFixed,2,2));
}
}
```

9.5.24 Виконайте програму і поекспериментуйте з різними значеннями лівої, правої межі та кроку аргументу. Закрийте вікно програми "Табулювання функції".

9.5.25.Збережіть створену програму у своїй папці.

9.5.26.Створіть exe-файл вашої програми.

9.5.27. Закрийте C++ Builder, запустіть створену програму і виконайте обчислення для різних початкових даних.

9.5.28. Продемонструйте створену форму викладачеві. Завершіть роботу.

9.5.29. Додайте до контекстового меню команду "Вийти" і запрограмуйте її.

9.5.30. Створіть *ще одне* контекстове меню з командами "Зняти всі прапорці", "Встановити всі прапорці", "Встановити інверсивно" і запрограмуйте його. Таке меню має з'являтися після клацання правою

клавішею миші на довільному місці панелі з прапорцями. Не забудьте "прив'язати" це меню (PopupMenu2) до панелі з прапорцями (див. пункт 9.5.20).

9.5.31. Модифікуйте реалізацію програми, передбачивши можливість табулювання функції і її похідної. Вибір варіанта табулювання (з похідною чи без неї) здійсніть за допомогою додаткового прапорця.

Виконайте такі дії:

- вставте у форму об'єкт типу CheckBox (прапорець), надайте його властивості Caption значення "Похідна", виберіть для підпису один із кирилізованих шрифтів 12-го розміру, вирівняйте вставлений об'єкт;
- змініть програмний код команди "Протабулювати".

9.5.32.Визначте кількість елементів масиву більших, ніж 0,5, і менших, ніж 1.

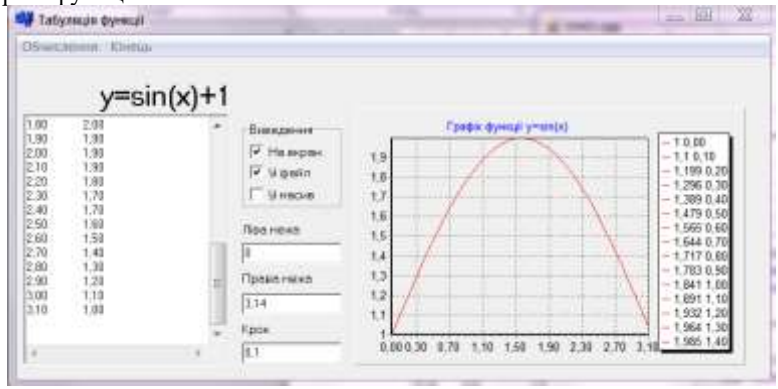
9.5.33. Передбачте у створеній програмі додаткову можливість для визначення максимального та мінімального значень функції.

Підказка. У тілі функції N4Click, що описує програмний код кнопки "Протабулювати", скористайтеся командами:

```
max = sin(a) + 1;   // На початку функції:
. . .
if (max < y) max = y;          // У циклі табулювання:
. . .
```

9.5.34. Змініть функції команди "Протабулювати" так, щоб для увімкненого прапорця "У масив" виведення результатів виконувалось в одновимірний масив.

9.5.35. Вставте у форму об'єкт Chart із закладки Aditional для побудови графіка функції:



Двічі клацніть на ньому і на закладці Series, клацнувши на кнопці Add, у вікні TeeChart Gallery, виберіть піктограму потрібного графіка. Набору даних, що відповідатиме цьому типу графіка, присвоюється стандартне ім'я Seriesl. Поставте чи заберіть прапорець 3D за власним бажанням => Ok. Поекспериментуйте з різними закладками вікна Editing Chartl. Створіть додатковий пункт меню або вставте кнопку "Нарисувати графік" для отримання графіка і запрограмуйте (наприклад, кнопку) так:

```
void __fastcall TForm1::N7Click(TObject *Sender)
{
float x,y;
float a=StrToFloat(Edit1->Text); // Одержуємо числове значення лівої межі
float b=StrToFloat(Edit2->Text); // Одержуємо числове значення правої межі
float h=StrToFloat(Edit3->Text); // Одержуємо числове значення кроку
for(x=a;x<b;x+=h)
{
Series1->Add(sin(x)+1,FloatToStrF(x,ffFixed,2,2),clRed);}
}
```

Поекспериментуйте з різними закладками вікна Editing Chartl, щоразу запускаючи програму на виконання Заберіть легенду (Legend), маркери (Points) і підписи до них (Marks). Придумайте та вставте заголовок (Title) та підпис (Foot) до графіка.

9.5.36. Сумістіть побудову графіка з табулюванням функції.

9.6. Контрольні запитання

9.6.1. Для чого призначена Геометрична фігура Shape?

9.6.2. Які властивості має Індикатор стану CGauge?

9.6.3. Яке значення властивості Kind відповідає круговій діаграмі стану CGauge?

9.6.4. Які значення може приймати аргумент Button?