Дрогобицький державний педагогічний університет

імені Івана Франка

Любов Лазурчак, Тетяна Вдовичин

ІНФОРМАТИКА. Програмування мовою C++

Методичні вказівки до виконання
лабораторних робіт

Дрогобич

2017

УДК 004 (07)
ББК 32.973.2 р
Л 19

Рекомендовано до друку вченою радою
Дрогобицького державного педагогічного університету імені Івана Франка
(протокол №6 від21.03.2017)

Р е ц е н з е н т и :

Галь Юрій Михайлович, кандидат фізико-математичних наук, доцент кафедри математики та методики викладання математики Дрогобицького державного педагогічного університету імені Івана Франка, доцент;

Сікора Оксана Володимирівна, кандидат технічних наук, доцент кафедри математики та методики викладання математики Дрогобицького державного педагогічного університету імені Івана Франка, доцент.

В і д п о в і д а л ь н и й з а в и п у с к :

Дорошенко Микола Васильович – кандидат фізико-математичних наук, доцент кафедри інформатики та обчислювальної математики Дрогобицького державного педагогічного університету імені Івана Франка.

Лазурчак Л.В., Вдовичин Т.Я.

Л 19 Інформатика. Програмування мовою С++ [методичні вказівки для підготовки фахівців першого (бакалаврського) рівня вищої освіти галузі знань 01 «Освіта» спеціальності 014.04 «Середня освіта (математика)», 014.08 «Середня освіта (фізика)» та галузі знань 11 «Математика та статистика» спеціальності 111 «Математика»] / **Любов Василівна Лазурчак, Тетяна Ярославівна Вдовичин.** – Дрогобич : Видавничий відділ Дрогобицького державного педагогічного університету імені Івана Франка, 2017. – 76 с.

Лабораторний практикум розроблено відповідно до програми навчальної дисципліни «Інформатика» для підготовки фахівців першого (бакалаврського) рівня вищої освіти галузі знань 01 «Освіта» спеціальності 014.04 «Середня освіта (математика)», 014.08 «Середня освіта (фізика)» та галузі знань 11 «Математика та статистика» спеціальності 111 «Математика».

У посібнику пропонуються теоретичні відомості з мови програмування С++ та методичні вказівки щодо виконання відповідних лабораторних робіт. Рекомендований студентам і викладачам при вивченні основ алгоритмізації та програмування, із застосування мови С++.

ЗМІСТ

ПЕРЕДМОВА

Навчальна дисципліна «Інформатика» є важливою складовою вищої освіти для підготовки фахівців галузі знань 01 «Освіта» предметна спеціалізація «Середня освіта (математика)» та «Середня освіта (фізика)».

У лабораторному практику продемонстровано використання мови програмування С++.

Основне завдання цього посібника — формування у майбутніх фахівців практичних навичок роботи з мовою програмування С++.

Структура лабораторних робіт передбачає: створення вихідної задачі, побудову алгоритму її розв’язку, аналіз отриманих результатів.

У кожній лабораторній роботі наведені: теоретичні відомості й методичні вказівки, необхідні для роз’язування запропонованих задач; детальний аналіз розв’язків типових задач з коротким поясненням основних положень; індивідуальні завдання. Контрольні запитання допоможуть виявити рівень засвоєння знань та вмінь студентів за підсумками виконання лабораторних робіт. Додаткові відомості з теорії можна отримати з книг, наведених у списку літератури. Така побудова посібника сприяє самостійному опрацюванню матеріалу дисципліни.

У навчальному посібнику розглянуті всі аспекти підготовки і розв'язання задач на комп'ютері, у тому числі способи формалізації та алгоритмізації задач, методи розробки програм, рекомендації до стилю, налагодження і тестуваннчя програм. Весь матеріал лабораторного практикуму розділений на 10 робіт, форма викладення яких методично глибоко продумана і забезпечує поступове та послідовне засвоєння основних можливостей мови С++. Матеріал поданий таким чином, що, починаючи вже з першої теми, можна приступити до практичного програмування з використанням комп'ютера. Усі теми однакові за своєю структурою і відображають процес навчання основ програмування.

Кожна робота складається з трьох розділів. Перший – «Теоретичні відомості», у якому досить ґрунтовно викладені основні аспекти мови з конкретної теми. Теми розташовані в такому порядку, щоб кожна наступна

тема не тільки надавала нові знання, але й закріплювала та розширювала знання з попередньої теми. Поруч із фрагментами програм, що ілюструють особливості застосування деяких типів або операторів, наводяться повні тексти програм з коментарями, що допомагають правильно виконати індивідуальне завдання. Виконуючи «Хід роботи», студент матиме змогу закріпити теоретичні знання і навчитися застосовувати їх на практиці. Ці завдання охоплюють матеріал певної теми і частково попередніх тем, що дає можливість закріпити, систематизувати й узагальнити набуті раніше знання, виробити погляд на процес програмування як на єдине ціле. Третій розділ – «Контрольні запитання» – уможливлює перевірити рівень засвоєних знань з тієї чи тієї тематики.

Оригінальна апробована методика опанування основами мови С++ зробить навчання доступним, зручним і приємним. Лабораторний практикум розроблений для студентів різних напрямів підготовки. Однак простота і послідовність викладення матеріалу, наявність прикладів з повними текстами програм роблять цей навчальний посібник загальнодоступним. Він з успіхом може бути використаний для самостійного вивчення мови С++. Особи, що засвоять в повному обсязі запропонований матеріал, будуть підготовлені до самостійного розв'язання задач, що виникають у практичній діяльності.

ВСТУП

Мову С розробив на початку 70-х рр. Деніс Рітчі, співробітник фірми Bell Telephon Laboratories (США). Початково вона створювалась як ефективний і зручний професійний інструментарій для програмування ОС Unix. Властивості С настільки захопили програмістів, що її почали застосовувати для створення програм у різних практичних сферах. Появилось багато версій мови. Тому в 1983 р. було сформовано комітет з розробки стандарту мови. У 1989 р. створюється стандарт, затверджений Американським національним інститутом стандартизації ANSI. У 1990 р. затверджується стандарт мови Міжнародною організацією стандартизації (ISO). Цей стандарт мови С повністю підтримують більшість сучасних компіляторів.

З 80-х рр. мовою С розробляють програми практично для всіх типів комп'ютерів, включаючи IBM-сумісні та комп'ютери Macintosh, а також для різних операційних середовищ: MS DOS, UNIX, Windows, Linux, OS/2. Створюються системи програмування мовою С та інтегровані середовища розробника (ICP). ICP призначені для швидкого та зручного запису й редагування тексту програм, їх компілювання і відлагодження, компонування великих програмних проектів. Найбільш популярне ICP – це Borland C/C++ фірми Borland Intern, яке уможливлює створювати програми мовами С та С++ для ОС MS DOS та Windows.

Мова С стала основою створення і розвитку багатьох мов ООП, зокрема С++, Java, C#. Реально С++ є розширенням мови С. Компілятори С++ підтримують всі синтаксичні конструкції та властивості мови С. Додатково С++ включає спеціальні засоби та бібліотеки, що реалізують принципи ООП.

Мова С – сучасна універсальна мова програмування, призначена для створення прикладних програмних продуктів і системних компонентів програмного забезпечення комп'ютерів. Основні риси мови:

- *потужність і гнучкість* (мовою С написано більшість програм ОС UNIX, створено ряд компіляторів з різних мов програмування, розроблено інтерфейси текстових і графічних редакторів та ін.);
- *ефективність* (програми швидкодіючі та ефективні);

- *структурованість* (мова С реалізує принципи структурного програмування: проектування зверху-вниз, модулність);
- *орієнтація на професіоналізм програміста;*
- *лаконізм;*
- *мобільність* (можливість переносу програм з однієї операційної платформи на іншу, тобто програми можуть бути виконані на комп'ютерах різних виробників і в різних операційних системах).

У 1998 р. були затверджені стандарти мови С++ (ANSI / ISO). ANSI – Американський інститут національних стандартів, ISO – міжнародна організація стандартів.

Мова С++ вторувала шлях для розроблення багатьох інших мов майбутнього. Наприклад, мови Java та C# – прямі «нащадки» мови С++. Мова С++ насправді є надбудовою мови С, тобто усі компілятори С++-програм можна використовувати для компілювання С-програм. Мову С++ можна назвати розширеною та поліпшеною версією мови С, у якій реалізовано технологію об'єктно-орієнтованого, узагальненого та процедурного програмування. Вона також містить ряд інших удосконалень мови С, наприклад, розширений набір бібліотечних функцій. Щоб до кінця зрозуміти й оцінити переваги мови програмування С++, необхідно зрозуміти все «як» і «чому» відносно мови С.

З погляду теорії програмування, у мову високого рівня закладено прагнення створити програмісту максимум зручностей у вигляді вбудованих засобів. Мова низького рівня не забезпечує програміста нічим, окрім доступу до реальних машинних команд. Мова середнього рівня надає програмісту певний (невеликий) набір інструментів, які дають змогу йому самому розробляти конструкції програм вищого рівня. Інакше кажучи, мова середнього рівня пропонує програмісту вбудовані можливості в поєднанні з її гнучкістю. Будучи мовою середнього рівня, мова С дає змогу маніпулювати бітами, байтами й адресами, тобто базовими елементами, з якими працює комп'ютер.

Таким чином, у мові С не передбачено спроби відокремити апаратні засоби комп'ютера від програмних. У мові С всі ці процедури виконуються за допомогою виклику бібліотечних функцій, а не за допомогою ключових слів, визначених у самій мові. Такий підхід підвищує функціональну гнучкість мови програмування С. Мова С дає змогу визначати підпрограми для виконання операцій високого рівня. Ці підпрограми називаються функціями, які є «будівельними» блоками мови С. Програміст може без особливих зусиль створити бібліотеку функцій, призначених для виконання різних задач, які має використовувати його програма. У цьому сенсі програміст може персоніфікувати мову С відповідно до своїх потреб.

Необхідно згадати ще про один аспект мови С, дуже важливий і для мови програмування С++: С – структурована мова, найхарактернішою особливістю якої є використання блоків. Блок – набір настанов, які логічно пов'язані між собою. Структурована мова підтримує концепцію функцій і локальних змінних.

Мова програмування С++ повністю містить всі особливості мови С, всі її засоби й атрибути, володіє всіма її перевагами. Для неї також залишається у силі принцип функціонування мови С, згідно з яким програміст, а не мова, особисто відповідальний за результати роботи своєї програми. Саме цей момент дає змогу зрозуміти, що процес розробки мови С++ не був спробою створити нову мову програмування. Це було мабуть удосконалення вже наявної (і при цьому дуже успішної) мови. Більшість нововведень, якими Б'ярн Страуструп збагатив мову С, було використано для підтримки технології ООП. По суті, мова С++ стала об'єктно-орієнтованою версією мови С. Проте в основу мови програмування С++ лягла не тільки мова С. Б'ярн Страуструп стверджує, що деякі об'єктно-орієнтовані засоби були інспіровані іншою об'єктно-орієнтованою мовою, а саме Simula 67. Таким чином, мова С++ є симбіозом двох могутніх технологій програмування. Хоча мова програмування С++ спочатку була спрямована на підтримку дуже великих програм, проте тільки цим її використання не обмежилося. Виявляється об'єктно-орієнтовані засоби мови програмування С++ можна ефективно

застосовувати практично до будь-якого завдання програмування. Мова програмування C++ часто використовують для створення компіляторів, редакторів, комп'ютерних ігор і програм мережевого обслуговування. Оскільки мова C++ володіє ефективністю мови C, то програмне забезпечення багатьох високоефективних систем побудоване з застосуванням мови C++. Окрім цього, мова програмування C++ – мова, яка найчастіше вибирається для Windows-програмування.

ЛАБОРАТОРНА РОБОТА № 1. Середовище програмування С++

Мета: *навчитися складати програми на С++ для обчислення арифметичних виразів.*

Теоретичні відомості

С++ є розширенням мови С. Окрім стандартних команд, сюди ввійшли засоби об'єктно-орієнтованого й узагальненого програмування. С++ – це перша мова в світі об'єктно-орієнтованого програмування, суть якого полягає в об'єднанні даних та алгоритмів їх опрацювання у єдине ціле.

Ім'я файла, який міститиме початковий код програми, формально може бути будь-яким. Але С++-програми зазвичай зберігаються у файлах з розширенням *.cpp.

Текст будь-якої програми складається з команд, описів змінних, сталих, приєднань бібліотек тощо. Такий текст часто називають *програмним кодом*. Програмний код необхідно перекласти на внутрішню мову комп'ютера, тобто створити машинний код. Цю операцію виконує *компілятор мови програмування*.

Особливості роботи в середовищі програмування С++

У різних операційних системах існує по декілька середовищ програмування мовою С++. Наприклад, для операційної системи Windows – це Turbo C++, Borland C++ 3.1, Borland C++ Builder тощо. Вони призначені для підготовки текстів програм мовою С++, їхньої компіляції (переведення програмного коду в машинний) та виконання. Принципи складання програм (окрім роботи з графікою у Windows) для них однакові. Розглянемо правила роботи на прикладі середовища Turbo C++ для DOS.

Для входу у середовище треба виконати команду tc.exe. У верхньому рядку екрана буде розміщене головне меню, а в нижньому – опис деяких функціональних клавіш.

Щоб активізувати (увійти в) головне меню, потрібно натиснути на клавішу F10. У розпорядженні користувача будуть такі пункти меню:

File – для роботи з файлами;

Edit – для редагування файла;

Search – для відшукання (заміни) заданого фрагмента тексту;

Run – для виконання програми;

Compile – для компіляції програми та створення exe-файла;

Debug – для налагодження програми;

Project – для створення проектів;

Options – для конфігурування середовища;

Window – для конфігурування вікон і роботи з ними;

Help – для надання допомоги.

Потрібний пункт вибирають стрілками переміщення курсора або мишкою і натискають на клавішу введення. Можна скористатися іншим способом, натиснувши й утримуючи клавішу Alt, натискають на клавішу з висвітленою літерою і відпускають обидві клавіші.

Розглянемо головні етапи, з яких складається сеанс роботи.

1. Активізують головне меню (натискають на F10) і пункт File. Одержують додаткове (спадне) меню. У ньому вибирають команду New (записуватимемо так: F10 => File => New). Середовище переходить у режим створення нового файла з назвою NONAME00.CPP. Набирають текст програми.

2. Для виправлення очевидних помилок уведення користуються традиційними прийомами редагування тексту, зокрема, такими комбінаціями клавіш для роботи з блоками (фрагментами) тексту:

Shift+стрілки – виокремити чи зняти виокремлення блока тексту;

Ctrl+lnsert – копіювати блок у буфер обміну;

Shift+Del – перемістити блок із тексту у буфер;

Shift+lnsert – вставити текст із буфера у позначене курсором місце основного тексту;

Ctrl+Del – вилучити виокремлений блок із тексту;

Ctrl+Y – вилучити рядок, де є курсор;

Ctrl+Q, Y – вилучити текст від курсора до кінця рядка;

Ctrl+N – вставити рядок;

Ctrl+PgUp – перейти до початку тексту;

Ctrl+PgDn – перейти до кінця тексту.

Цю інформацію пам’ятати не обов'язково. Опис перших чотирьох можна побачити, активізувавши пункт головного меню Edit (Alt+E).

3. Якщо очевидних помилок немає, програму можна компілювати й виконати (F10=>Run або за допомогою комбінації клавіш Ctrl+F9).

4. Якщо система виявить синтаксичні помилки, то про це буде негайно повідомлено. Курсор буде в рядку, де допущено помилку, або безпосередньо вказуватиме на позицію з помилкою. На екрані появиться рядок червоного кольору з повідомленням про зміст помилки, що суттєво полегшує її виправлення. Середовище перебуватиме в режимі редагування і помилку можна буде виправити. Тепер вдруге компілюють і виконують програму (натискають на Ctrl+F9). Виправляють наступну помилку у разі потреби і т.д.

5. Якщо синтаксичних помилок не виявлено, але отримані результати програми неправильні, за допомогою команд Run => Step over (або F8) виконують покрокову компіляцію. Також можна простежити, як змінюються значення змінних під час компіляції. Для цього виконують команди Debug => Watches => Add watch... => вводять назву змінної => Ok (натискають на Ctrl+F7).

6. Якщо синтаксичних помилок немає, програма буде виконана. Результати можна побачити у вікні результатів, для чого натискають на Alt+F5 або використовують засоби пункту Debug. Натиснувши після перегляду результатів на будь-яку клавішу, переходять у режим редагування програми.

7. Щоб для заданого cpp-файла створити exe-файл, послідовно виконуємо F10 => Compile => Destination => Disk. Натискають на Alt+F9 і в поточний каталог на диску буде записано exe-файл, який можна виконувати поза середовищем.

8. Щоб зберегти текст програми у файлі з розширенням cpp, активізують F10 => File => Save As, якщо файлу дають нове ім'я, або F10 => File => Save (достатньо натиснути на клавішу F2) для зберігання файла зі старим іменем.

9. Для закінчення сеансу роботи і виходу зі середовища потрібно виконати F10 => File => Exit або натиснути на клавіші Alt+x.

10. Для роботи з програмою, що є на диску, виконують F10 => File =>

Open або натискають на клавішу F3. Одержимо діалогове вікно. За допомогою клавіші Tab переходимо в нижню частину вікна і вибираємо серед імен файлів потрібний, натискаємо на клавішу введення. Текст програми буде занесено у вікно редагування.

11. Вікон з програмами може бути декілька. Переходити від одної програми до іншої можна за допомогою клавіші F6. Щоб розкрити на весь екран чи згорнути вікно, користуються клавішею F5. Зручно розташувати вікна на екрані можна засобами пункту Window. Щоб закрити активне вікно, натискають на Alt+F3 або клацають мишею на значку прямокутника в рамці вікна.

12. Якщо потрібна додаткова інформація, використовують клавішу F1 і читають інформаційно-довідкові тексти про середовище та синтаксичні конструкції мови C++.

Компілювання програми

Спосіб компілювання програм *.cpp залежить від використовуваного компілятора і вибраних опцій. Квім того, багато компіляторів, наприклад, Visual C++ (Microsoft) і C++ Builder (Borland), надають два різні способи компілювання програм: за допомогою компілятора командного рядка й інтегрованого середовища розробки (Integrated Development Environment – IDE). Тому для компілювання C++-програм неможливо дати універсальні настанови, які підійдуть для всіх компіляторів. Це означає, що необхідно дотримуватися настанов, наведених у супровідній документації, що додається до компілятора. Але найпопулярнішими компіляторами є Visual C++ і C++ Builder. Наведемо настанови з компілювання програм, які відповідають цим компіляторам.

Найпростіше в обох випадках компілювати і виконувати програми з використанням компіляторів командного рядка.

Унаслідок роботи C++-компілятора утворюється об'єктний код, який може виконати комп'ютер після завершення роботи компілятора.

Базові елементи мови

Алфавіт мови, або його символи – це основні неподільні знаки, за допомогою яких формуються всі конструкції мови. Алфавіт мови С++ можна поділити на три групи символів:

1) великі і малі букви латинського алфавіту“A”, .. ,”Z”, “a”, .. ,”z”, цифри (0..9 – десяткова система числення, 0..9, A..F – шістнадцяткова система числення), символ підкреслення _ ;
2) знаки операцій, символи пунктуації, роздільники , а саме: + - * / % = <>& | ! ~ ^ ? , . ; : ‘ “ ( ) [ ] { } # \;
3) неграфічні символи – це пропуск, керуючі символи або ескейп-послідовності (послідовність символів, що починаються із оберненої похилої риски \ ). Призначені для відокремлення лексем і керування виведення даних і повідомлень.

Лексема – найменший змістовний елемент програми, що має самостійне значення і формується із символів алфавіту: ідентифікатори, ключові (зарезервовані) слова, знаки операцій, константи, роздільники.

Ідентифікатор – це ім’я програмного об’єкта, яке надає користувач. Формується із малих і великих латинських літер, цифр і символа підкреслення. Починається завжди із букви або із символу підкреслення. Пропуски не допускаються. Великі та малі букви розрізняються. Ідентифікатори мають мати інформативний зміст. Довжина ідентифікатора за стандартом не обмежена, але деякі компілятори її обмежують.

Ключові слова (зарезервовані або службові) – це набір визначених слів, що використовуються для установлення типів даних, класів пам’яті даних формування операторів тощо. Кожне ключове слово має своє призначення і застосовувати їх з іншою метою заборонено. Усі ключові слова розділені за призначенням:

- ключові слова типів даних: char, double, enum, int, float, long, short, signed, void,…
- ключові слова класифікаторів типів і класів пам’яті: auto, const, extern,…
- ключові слова операторів: break, case, do, if, else, for, goto, return,…

- інші: sizeof, typedef…

У ключових словах великі і малі літери розрізняються. Тому записувати їх слід за стандартом.

Знаки операцій – це один або більше символів, що визначають дії над операндами. Операції поділяються на унарні, бінарні та тернарні за кількістю операцій. Один і той самий знак може трактуватися по-різному, залежно від контексту. Всі знаки операцій є окремими лексемами, розривати їх не можна.

Константи – це об’єкти програм, значення яких змінювати не можна. Поділяються на: цілі, дійсні, символьні, рядкові.

Цілочисельні константи можуть записуватись в одному з трьох форматів:

- десятковий – послідовність десяткових цифр, що починається не з нуля, якщо це не число нуль. Наприклад: 5266, 200, -13, +455;
- вісімковий – нуль, за яким слідують вісімкові цифри (0, 1,…,7). Наприклад: 01, 020, 07155;
- шістнадцятковий – 0x або 0X, за яким йдуть шістнадцяткові цифри (0, 1, …,9 , A, B, C, D, E, F). Наприклад: 0x9342, 0XDA07.

Дійсні константи можуть записуватись у наступних форматах:

– з фіксованою крапкою, тобто **[цифри].[цифри]**. Наприклад: 7.123, 0.60, 384.4;

– з плаваючою крапкою (експоненціальна форма), тобто [цифри]$_{[.]}$[цифри].

Вираз задає правило обчислення деякого значення.

Оператор задає опис деякої дії. Для опису складних дій потрібна послідовність операторів. Їх можна об’єднувати в складений оператор, який сприймається як один оператор. Оператори бувають виконувані та невиконувані. Виконувані оператори задають дії над даними, невиконувані – служать для опису даних. Сукупність описів та операторів утворюють програму.

Директиви препроцесора

Препроцесор – це програма, яка опрацьовує директиви.

Директиви препроцесора – це команди компілятора відповідної мови програмування, які виконуються на початку компіляції програми. Директиви мови С++ починаються із символу #.

Директива #include означає, що до програми необхідно приєднати код із зазначеного після неї файла.

Файли, які приєднюють директивою #include, називаються файлами заголовків (header-файлами, бібліотеками, модулями). Усі стандартні команди та функції мови С++ визначені у файлах заголовків. Щоб приєднати модуль до програми користувача, директиву препроцесора необхідно зазначити на початку програми так:

```
#include <назва файлу.розширення>
```

Зазвичай усі стандартні бібліотеки розміщені у папці include середовища С++. У такому випадку назва файла є параметром директиви, її зазначають у кутових дужках <назва>.

Директива **#include <iostream.h>** під'єднює бібліотечний файл iostream.h, у якому описані функції, які дають змогу виконувати операції введення – виведення даних.

Суттєвою особливістю мови С++, порівняно з іншими мовами, є те, що програми складаються з функцій, які відіграють роль підпрограм в інших мовах. Головна функція, яка має бути у кожній програмі, – це функція вигляду

```
main(void)
    {
     тіло функції з командою return 0;
     }
```

де main() – **заголовок функції**. Ключове слово **void** означає, що функція не залежить від параметрів, його записувати не обов'язково.

Функцію main() можна записувати й так:

```
void main()
    {
     тіло функції;
     }
```

Така функція називається функцією main() типу **void**. Вона не повертає у програму жодних значень, тому команду return писати не треба.

Найпростіша програма мовою С++ має такий вигляд:

```
// коментарі
 #include <назва бібліотечного файла>
 void main()
          {
           <тіло функції>;
           }
```

Основні типи даних у С++

Будь-які дані, з якими працює програма, мають належати до певного типу. Тип даних визначає:

- внутрішнє представлення даних в пам'яті – обсяг оперативної пам'яті, який резервується для даного;
- множину значень;
- операції та функції, які можна застосувати до величин цього типу.

Визначити, який обсяг пам'яті компілятор надає даному того чи того типу, можна за допомогою команди **sizeof(<назва типу>).**

Усі типи мови С++ можна поділити на основні та складені. До основних належать:

1) int (цілий);
2) char (символьний);
3) wchar_t (розширений символьний);
4) bool (логічний);
5) float (дійсний);
6) double (дійсний подвійної точності).

Типи 1 – 4 називають цілими типами, типи 5,6 – типами із плаваючою крапкою. Існує 4 специфікатори типу, які уточнюють внутрішнє представлення та діапазон значень стандартних типів:

1) short (короткий);
2) long (довгий);
3) signed (знаковий);
4) unsigned (беззнаковий).

Типи даних

У таблиці 1 та 2 наведені назви основних числових типів, обсяги пам'яті, які резервують для екземплярів даних цих типів і діапазони допустимих значень даних.

Таблиця 1. Дані цілочисельних типів

Назва типу	Обсяг, байтів	Діапазон допустимих значень
int	2 або 4	-32768 … 32767 або -2147483648 … 2147483647
short int	2	-32768 … 32767
unsigned short int	2 або 4	0 … 65535 або 0 … 4294967295
long int	4	-2147483648 … 2147483647
unsigned long int	4	0 … 4294967295

Таблиця 2. Дійсні типи

Назва типу	Обсяг, байтів	Діапазон значень
float	4	$\pm 3.410^{-38} \ldots \pm 3.410^{38}; 0$
double	8	$\pm 1.710^{-308} \ldots \pm 1.710^{308}; 0$
long double	10	$\pm 1.1810^{-4932} \ldots \pm 1.1810^{4932}; 0$

Символьний тип (char) – це множина символів кодової таблиці. Символьна стала – це один символ (1 байт), узятий у лапки на зразок апострофа, або число у 8-й, 10-й чи 16-й системі числення, яке є кодом символу у таблиці ASCII.

Логічний тип (bool) характеризується двома значеннями даних: false (хибність) і true (істина).

Введення-виведення даних

У С++ немає вбудованих команд уведення – виведення даних. Для організації введення – виведення тут реалізована концепція потоків, яка визначена у спеціальних модулях (iostream.h – команда виведення і введення).

Під ***потоком*** розуміють процес уведення – виведення інформації у файл. Такі периферійні пристрої введення – виведення, як клавіатура, монітор, принтер, розглядаються як текстові файли. Під час виконання будь-якої програми автоматично підключаються стандартні потоки для введення даних з клавіатури (cin), виведення на екран (cout), виведення повідомлення про помилки (cerr) і допоміжний потік (clog).

Стандартні потоки використовують команди введення (>>) та виведення (<<) даних. *Команда уведення даних із клавіатури* дає змогу виконувати програму для різних вхідних даних, що робить її більш універсальною (масовою). Команда введення >> описана у бібліотеці iostream.h і має такий загальний вигляд:

```
cin >> <змінна>;
```

Для виведення на екран повідомлень і результатів обчислень використовують стандартний потік виведення cout і команду <<, які визначені у бібліотеці iostream.h:

```
cout << <вираз 1> << <вираз 2> << …<< <вираз N>;
```

Приклад 1

Скласти програму «Моя перша програма на мові С++».

```
#include<iostream.h>
void main()
{
cout<<"Moya persha programa na movi C++";_
}
```

Рис. 1. Реалізація прикладу 1

```
Moya persha programa na movi C++_
```

Рис. 2. Результати виконання прикладу 1

Хід роботи

1. Запустити середовище C++.
2. Антивізувати головне меню (натиснути F10). Вибрати пунк меню File, команда New.
3. Виконати програму в середовищі C++, розглянуту в теоретичній частині (приклад 1).
4. Набрати текст програми, яка виводитиме на екран Ваше прізвище, ім’я та по-батькові.
5. Поекспериментувати з пунктом меню Edit (Alt+E).
6. Здійснити компіляцію програми (Alt+F9).
7. При відсутності синтаксичних помилок, запустити програму на виконання (Ctrl+F9).
8. Переглянути вікно результатів (Alt+F5).
9. Повернутися з вікна перегляду результатів у вікно програми, натиснувши на будь-яку клавішу.
10. Записати програму у власному каталозі на диску D.
11. Набрати текст програми, яка виводитиме на екран Вашу адресу.

Контрольні запитання

1. З чого складається алфавіт мови?
2. Що таке ключове слово?
3. Які є ключові слова?
4. Що таке препроцесор?
5. Що таке директива препроцесора?
6. Які є відомі директиви?
7. Що таке файл заголовків?
8. Яка загальна структура програми?
9. Які є типи даних?
10. Що таке специфікатор типу?

ЛАБОРАТОРНА РОБОТА № 2. Лінійні програми

Мета: *навчитися складати програми на С++ для обчислення арифметичних виразів.*

Теоретичні відомості

Змінні та сталі

Змінна чи стала – це пойменована ділянка оперативної пам'яті комп'ютера, де зберігається значення деякої величини.

Змінні та сталі (їх прийнято називати даними) мають такі властивості: назва (ім'я), значення, тип. Для роботи із даними слід зарезервувати певний обсяг оперативної пам'яті комп'ютера, де зберігатимуться їхні значення. Тому всі дані, які використовуються у програмі, потрібно заздалегідь описати (оголосити), оскільки компілятор розподіляє пам'ять згідно з описами.

Якщо значення деякої величини (даного) не змінюватиметься протягом усієї програми, то таке дане варто задати як сталу (константу, **const**). Це можна зробити так:

```
const <назва сталої 1> = <значення сталої 1>;
```

або так:

```
const <тип> <назва сталої 2> = <значення сталої 2>;
```

Сталу 2 називають ***типованою*** сталою. За замовчування *числова стала належить до* ***цілого*** *типу*. Під час виконання програми значення сталих змінювати не можна.

Дані, які під час виконання програми можуть набувати різних значень, називаються ***змінними***. Їх оголошують так:

```
<тип змінних 1> <список змінних 1>;
…
<тип змінних N> <список змінних N>;
```

Змінним можна задавати початкові значення відразу під час оголошення. Це називається ***ініціалізацією*** даних.

Програма – це послідовність команд, за допомогою яких записують алгоритм розв'язування конкретної задачі.

Команда присвоєння має такий загальний вигляд:

<назва змінної>= <вираз>

Арифметичні операції

Таблиця 3. Арифметичні операції

Пріоритет	Операції	Зміст операції
1	+, -	Присвоєння знака
2	*, /, %	Множення, ділення, остача від ділення
3	+, -	Додавання, віднімання
4	==, !=, <, <=, >, >=	Порівняння (відношення)

Операції порівняння:

- == означає дорівнює;
- != – не дорівнює;
- <= – менше або дорівнює;
- >= – більше або дорівнює.

Виконання кожної операції здійснюється з урахуванням їхніх пріоритетів (1 – найвищий). Для зміни звичайного порядку виконання операцій використовують круглі дужки.

Математичні функції

Усі стандартні математичні функції у C++ описані у бібліотеці math.h. Тому, якщо вони використовуються, на початку програми необхідно записати рядок під’єднання потрібного файла заголовків

#include <math.h>;

Основні математичні функції бібліотеки math.h наведені у таблиці 4. Аргументи функцій записують у круглих дужках.

Таблиця 4. Математичні функції

Назва функції	**Математичний запис**
abs(x)	$\|x\|$
cos(*x*)	cos(*x*)
sin(*x*)	sin(*x*)
tan(*x*)	*tg*(*x*)
log(*x*)	ln(*x*)
pow(*x*, *y*)	x^y
sqrt(*x*)	$\sqrt{x}$
exp(*x*)	e^x
*pow*10(*x*)	10^x
log10(*x*)	lg(*x*)
fabs(*x*)	$\|x\|$
acos(x)	arcos(x)
asin(x)	arcsin(x)
atan(x)	arctg(x)
ceil(x)	заокруглює число *x* до більш цілого
floor(x)	відкидає дробову частину числа *x*
fmod(x,y)	обчислює остачу від ділення числа *x* на число *y*

Усі наведені функції, крім abs(x) і *pow*10(*x*), мають тип аргумента і результат double. Для функцій abs(x) і *pow*10(*x*) типом аргумента і результату є int.

Усі інші математичні функції можна виразити через основні. Наприклад, $ctgx = 1/tgx$, $\log_b a = \ln(a)/\ln(b)$ тощо.

Послідовність виконання операцій у виразах така ж, як у математиці, й визначається правилом пріоритетів:

1) обчислюються значення всіх функцій, які входять у вираз;
2) виконуються операції присвоєння знака, множення, ділення та остачі від ділення;
3) виконуються операції додавання та віднімання.

Операції одного рівня виконуються послідовно зліва направо. Для зміни порядку виконання операцій використовують круглі дужки. Спочатку обчислюються вирази у дужках – передовсім у внутрішніх, потім – у зовнішніх. *Кількість відкритих і закритих дужок у виразі має бути однаковою.*

Усі елементи виразів (дроби, показник степеня, індекси) записують у горизонтальному рядку. У багатьох випадках їх беруть у дужки. Вирази можна записувати у декількох рядках. «Розривати» вирази можна, наприклад, після символу арифметичної операції. Власне символ дублювати не потрібно.

Приклад 2

Скласти програму, яка обчислює значення функції $y = \sqrt[5]{x^2 + 7.2} - |x - 5| + \sin\frac{\pi x}{3}$ для x=2.

```
#include<iostream.h>
#include<math.h>
#include<conio.h>
void main()
{
clrscr();
const float Pi=3.1415926;
float x,y;
cout<<"x=";
cin>>x;
y=pow(x*x+7.2,1/5)-abs(x-5)+sin(Pi*x/3);
cout<<"\n x="<<x<<"\t"<<"y="<<y;
getch();
}
```

Рис. 3. Реалізація прикладу 2

```
x=2     y=-1.133975
```

Рис. 4. Результати виконання прикладу 2

Хід роботи

1. Запустити середовище С++.
2. Виконати програму в середовищі С++, яка розглянута в теоретичні частині (приклад 2).
3. Скласти програму, яка обчислює значення функції $y = \log_5 |x - 12.5x^9| + \frac{2x - 4}{|x^8 - 12x^4 + 5.1x^3|}$.

4. Обчислити значення функції згідно зі своїм порядковим номером в академічному журналі:

№	**Функція** $f_n(x)$
1.	$9.2\cos 2x - \left\lvert \frac{\sin x}{1.1} \right\rvert$
2.	$12.4\sin\left\lvert \frac{x}{2.1} \right\rvert - 8.3\cos 1.2x$
3.	$\left\lvert \frac{\cos x}{2.7} \right\rvert + 9.1\sin(1.2x+1)$
4.	$\left\lvert \frac{\sin x}{3.12} + \cos x^2 \right\rvert - 8.3\sin 3x$
5.	$\frac{\cos\lvert 2x \rvert}{1.12} - \cos(3x-2) + 6.15$
6.	$\sin x \cos^2 x \sin(x+1.4)/0.85 + 7.14$
7.	$\lvert \sin(2x - 1.5 + 3\sin 4x \rvert + 2.38$
8.	$\cos x^2 \sin(2x-1) + 4.29$
9.	$\cos(x^{2.4}+1) - \lvert \sin 2x - 5.76 \rvert$
10.	$\sin x - \cos x^3 \sin(x^2 - 4.2) + 4.27$
11.	$\left\lvert \frac{\sin 12x \cos\lvert 2x \rvert}{3} \right\rvert + 4.21$
12.	$\frac{\cos x^3}{2.1} + \frac{\cos x^2}{1.1} - 8.3\sin(3x+1)$

Контрольні запитання

1. Що таке стала?
2. Що таке змінна?
3. Як оголосити змінну?
4. Як оголосити сталу?
5. Пріоритет виконання арифметичних операцій.
6. Операції порівняння.
7. Яку дію виконує команда clrscr()?
8. Яку дію виконує команда getch()?
9. Як описати стандартні математичні функції?
10. Вигляд математичних функцій у середовищі C++.

ЛАБОРАТОРНА РОБОТА № 3. Реалізація найпростіших математичних обчислень у середовищі програмування С++

Мета: *навчитися розв'язувати прості математичні задачі засобами С++.*

Теоретичні відомості

Керуючі послідовності

Керуючі послідовності – це комбінації спеціальних символів, які використовуються для введення та форматного виведення даних. Керуюча послідовність складається із символу «\» і спеціально означеного символу. Вони призначені для форматованого виведення результатів обчислення на екран, наприклад, для переходу на новий рядок, подання звукового сигналу, а також для виведення на екран деяких спеціальних символів: апострофа, лапок тощо. Основні керуючі послідовності наведено у таблиці 5.

Таблиця 5. Керуючі послідовності

Символи керуючих послідовностей	***Коментар***
\a, \7	Подати звуковий сигнал
\b	Повернути курсор на один символ назад (знищити попередній символ)
\f	Перейти на нову сторінку
\n	Перейти на новий рядок
\r	Повернути курсор на початок рядка
\t	Перевести курсор до наступної позиції табуляції
\v	Вертикальна табуляція
\\	Вивести символ похилої риски
\’	Вивести символ апострофа
\”	Вивести символ лапок
\?	Вивести знак запитання

Під час виконання програми функція clrscr() зітре з екрана усі результати та повідомлення попередніх програм (якщо такі будуть).

У бібліотечному файлі conio.h визначені функції clrscr() (очищення екрана) та getch() (затримка результів виконання програми).

Операції інкременту (++) та декременту (--)

Операції інкременту і декременту існують у двох формах – префіксній та постфіксній. Якщо символи ++ (--) записані перед змінною – то це інкремент (декремент) у префіксній формі, а якщо після змінної – у постфіксній. Операція інкременту має такий вигляд:

++ <змінна> або <змінна>++

Дія операції. Значення змінної збільшується на одиницю. Команди a++, ++a рівносильні команді a=a+1. Форма інкременту (декременту) впливає на порядок виконання операцій у виразах.

Аналогічно операція декременту має такий вигляд:

-- <змінна> або <змінна>--

Значення змінної зменшується на одиницю. Команди –a та a—діють як і команда a=a-1.

Приклад 3

Нехай задано катети прямокутного трикутника a=3, b=4. Знайти периметр і площу трикутника.

```
#include <iostream.h>
#include <math.h>
#include <conio.h>
void main()
{
clrscr();
int a=3,b=4,c,p,s;
c=sqrt(a*a+b*b);
p=a+b+c;
s=a*b/2;
cout<<"\n p="<<p;
cout<<"\n s="<<s;
cout<<"\n Vykonav  Petrenko I.";
getch();
}
```

Рис. 5. Реалізація прикладу 3

```
p=12
s=6
Vykonav  Petrenko I.
```

Рис. 6. Результати виконання прикладу 3

Приклад 4 (про трикутник, заданий координатами вершин)

Дано координати трьох вершин трикутника A(1;1), B(2;2) та C(-1;2). Обчислити медіану m_b і радіус описаного кола *r*.

```
#include <iostream.h>
#include <math.h>
#include <conio.h>
void main()
{
clrscr();
float x1,x2,x3,y1,y2,y3;
cout<<"Vvid koordynat tochky A \n";cin>>x1>>y1;
cout<<"Vvid koordynat tochky B \n";cin>>x2>>y2;
cout<<"Vvid koordynat tochky C \n";cin>>x3>>y3;
float a,b,c,x,y,mb,p,s,r;
a=sqrt(pow(x3-x2,2)+pow(y3-y2,2));
b=sqrt(pow(x1-x3,2)+pow(y1-y3,2));
c=sqrt(pow(x1-x2,2)+pow(y1-y2,2));
x=(x1+x3)/2;
y=(y1+y3)/2;
mb=sqrt(pow(x-x2,2)+pow(y-y2,2));
p=(a+b+c)/2;
s=sqrt(p*(p-a)*(p-b)*(p-c));
r=a*b*c/(4*s);
cout<<"\n mb="<<mb;cout<<"\n r="<<r;getch();}
```

Рис. 7. Реалізація прикладу 4

```
Vvid koordynat tochky A
2 1
Vvid koordynat tochky B
1 4
Vvid koordynat tochky C
5 5

 mb=2.692582
 r=2.507385
```

Рис. 8. Результати виконання прикладу 4

Довідка. Для розв'язування типових задач про трикутник наведемо формули обчислення деяких величин:

- відстань d між точками (x_1, y_1), (x_2, y_2): $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$;
- координати середини відрізка: $x = (x_1 + x_2)/2$, $y = (y_1 + y_2)/2$;
- півпериметр трикутника: $p = (a + b + c)/2$;
- площа трикутника: $s = \sqrt{p(p-a)(p-b)(p-c)}$;
- висота трикутника: $h_a = 2s/a$, $h_b = 2s/b$, $h_c = 2s/c$;
- бісектриса трикутника:

$$\omega_\alpha = \frac{2}{b+c}\sqrt{bcp(p-a)},\ \omega_\beta = \frac{2}{a+c}\sqrt{acp(p-b)},\ \omega_\gamma = \frac{2}{a+b}\sqrt{abp(p-c)};$$

- радіус описаного кола: $R = \frac{abc}{4s}$;
- радіус вписаного кола: $r = \frac{s}{p}$,

де a, b, c – сторони,

а α, β, γ – відповідно кути трикутника.

Хід роботи

1. Запустити середовище C++.
2. Виконати програми в середовищі C++, які розглянуті в теоретичній частині (приклад 3 та 4).
3. Скласти й виконати програму, задавши вхідні дані самостійно, згідно зі своїм порядковим номером в академічному журналі:

№	Задача
1.	Квіткова клумба має форму круга. Обчислити її периметр і площу за заданим радіусом.
2.	Обчислити периметр і площу прямокутного трикутника за заданим катетом і гострим кутом.
3.	Обчислити довжину кола і площу круга за заданим діаметром.

№	Задача
4.	Ділянка лісу має форму рівнобічної трапеції. Обчислити її периметр і площу за заданими сторонами.
5.	Ресторан закуповує щодня масло m_1 кг по 8,50 грн за кілограм, сметану m_2 кг по 2,40 грн, вершки m_3 кг по 4,10 грн. Визначити суми, потрібні для купівлі окремих продуктів, і загальну суму.
6.	Скільки секунд мають доба, тиждень, рік?
7.	Обчислити кінетичну $E = m\vartheta^2/2$ та потенціальну $P = mgh$ енергії тіла заданої маси m, яке рухається на висоті h зі швидкістю ϑ.
8.	Обчислити площу поверхні $S = 4\pi r^2$ та об'єм $V = 4\pi r^3/3$ сфери за заданим радіусом r.
9.	Швидкість світла 299792 км/с. Яку відстань долає світло за годину, добу?
10.	Радіус Місяця 1740 км. Обчислити площу поверхні $S = 4\pi r^2$ та об'єм планети $V = 4\pi r^3/3$.
11.	Обчислити об'єм та площу бічної поверхні куба, якщо відоме ребро.
12.	Обчислити площу бічної поверхні $S = 2\pi rh$ та об'єм $V = \pi r^2 h$ діжки за заданою висотою h та радіусом основи r.

4. Скласти і виконати програму, задавши вхідні дані самостійно, згідно зі своїм порядковим номером в академічному журналі:

Задача про трикутник.

Дано координати трьох вершин трикутника A(0;0), B(i;i-1) та C(-i;i+1), де i – номер варіанта.

№	Обчислити:
1.	висоту h_a та бісектрису ω_c
2.	медіану m_a та бісектрису ω_b
3.	бісектрису ω_a та радіус вписаного кола r
4.	висоту h_a та медіану m_b
5.	медіану m_b та бісектрису ω_c
6.	бісектрису ω_a та радіус описаного кола R
7.	висоту h_b та бісектрису ω_a

№	Обчислити:
8.	висоту h_b та медіану m_c
9.	висоту h_a та радіус вписаного кола r
10.	медіану m_c та бісектрису ω_a
11.	висоту h_b та бісектрису ω_c
12.	медіану m_c та радіус вписаного кола r

Контрольні запитання

1. Як подати звуковий сигнал, використовуючи керуючі послідовності?
2. Як повернути курсор на один символ назад, використовуючи керуючі послідовності?
3. Як перейти на нову сторінку, використовуючи керуючі послідовності?
4. Як перейти на новий рядок, використовуючи керуючі послідовності?
5. Як повернути курсор на початок рядка, використовуючи керуючі послідовності?
6. Як перевести курсор до наступної позиції табуляції, використовуючи керуючі послідовності?
7. Як здійснити вертикальну табуляцію, використовуючи керуючі послідовності?
8. Як вивести символ похилої риски, використовуючи керуючі послідовності?
9. Як вивести символ апострофа, використовуючи керуючі послідовності?
10. Як вивести символи лапок, використовуючи керуючі послідовності?

ЛАБОРАТОРНА РОБОТА № 4. Програмування розгалужених алгоритмів: оператор IF

Мета: *вивчити принципи роботи команди розгалуження if та вміти її застосовувати при розв'язуванні задач.*

Теоретичні відомості

Логічний вираз – це засіб, що дає змогу записувати умову у задачах відшукання даних, що задовольняють деякий критерій. Логічний вираз може набувати значення true (істинність) або false (хибність). Логічні вирази бувають ***прості і складені***. Простий – це два арифметичні вирази, з'єднані символом відношення, а складений – це прості логічні вирази, з'єднані ***логічнимим операціями:***

1) ! – не,
2) && – і,
3) ‖ – або.

Логічні вирази обчислюються з урахуванням пріоритету логічних операцій (1 – найвищий). Однакові логічні операції виконуються послідовно зліва направо. Для зміни порядку виконання логічних операцій, як і для звичайних арифметичних, використовують круглі дужки.

Команда розгалуження **if** має дві форми: повну та коротку.

Повна така:

if (<логічний вираз>) <команда 1>; else <команда 2>;

Дія команди. Обчислюється значення логічного виразу. Якщо це значення істинне, то виконується команда 1, у протилежному випадку – команда 2. Команда 1 та команда 2 можуть бути порожніми, простими або складеними.

Коротка команда розгалуження **if** має вигляд:

if (<логічний вираз>) <команда 1>;

Дія команди. Обчислюється значення логічного виразу. Якщо воно істинне, то виконується команда 1, інакше виконується команда, яка записана після команди **if .**

Приклад 5

Обчислити й вивести на екран значення складеної функції y у деякій заданій користувачем точці x, якщо

$$y = \begin{cases} tg|x|, x < 0, \\ x^3, 0 \le x < 5, \\ \log_5 x, x \ge 5. \end{cases}$$

```
# include <iostream.h>
#include <math.h>
#include <conio.h>
void main()
{
clrscr();float x,y;
cout<<"\n Vvedit x=";cin>>x;
if (x<0) y=tan(fabs(x)); else
                              if (x>=0 && x<5) y=pow(x,3);
                              else y=log(x)/log(5);
cout<<"\n y="<<y;getch();  }
```

Рис. 9. Реалізація прикладу 5

Таблиця 6. Результати виконання програми, реалізованої оператором **if**

$x < 0$	$0 \le x < 5$	$x \ge 5$
Vvedit x=-3 y=-0.142547_	Vvedit x=1 y=1	Vvedit x=6 y=1.113283_

Хід роботи

1. Запустити середовище C++.
2. Виконати програму в середовищі C++, яка розглянута в теоретичній частині (приклад 5).
3. Скласти дві програми, використовуючи:
   1) повну команду розгалуження **if**;
   2) коротку команду розгалуження **if**.

Вхідні дані (x, a, b, c, d) увести з клавіатури довільні. Результати обчислень вивести на екран. Скласти і виконати програму, задавши вхідні дані самостійно, згідно зі своїм порядковим номером в академічному журналі.

Задача.

Увести довільне значення x та обчислити значення функції

$$y = \begin{cases} f_{i+2}(\varphi), |x| < 10, \\ f_{i+3}(\omega), |x| \geq 10, \end{cases}$$

де $\varphi = tg(x + a) - \log_i |b + 7|$,

$\omega = c\sqrt[5]{x^2 + de^{1.3}}$,

i – номер варіанта.

3. Скласти й виконати програму, використовуючи команду розгалуження if, задавши вхідні дані самостійно, згідно зі своїм порядковим номером в академічному журналі:

№	Варіанти завдань
1.	$y = 1 + 9x + \begin{cases} \ln\lvert\sin x\rvert + x^7 & , x \leq 0 \\ \operatorname{ctg}\dfrac{\lvert x+1\rvert}{2} & , 0 < x \leq 3 \\ 3x - x^5 & , x > 3 \end{cases}$
2.	$y = \dfrac{1}{x} + 4 - \begin{cases} 0{,}65x + 8 & , x < 1 \\ \operatorname{arctg}\dfrac{x+6{,}1}{2} + e^x & , 1 \leq x < 5 \\ \sqrt{1+\sqrt{x}} & , x \geq 5 \end{cases}$
3.	$y = \dfrac{2}{x} + \lvert x\rvert + \begin{cases} 1 + 4x^2 & , x < 0 \\ \left(e^x + \lvert x\rvert\right)^2 & , 0 \leq x \leq 2 \\ 5\sin\left(x^2 + 1\right) & , x > 2 \end{cases}$
4.	$y = 1 + x + \begin{cases} e^{\ln(2+2x)+2x} & , x \leq 4 \\ \operatorname{ctg}\dfrac{1+x}{9} + 8x & , 4 < x \leq 7 \\ 1 - 7x + x^2 - 2x^3 & , x > 7 \end{cases}$

5.	$y=\frac{1}{\lvert x+2\rvert}+1-\begin{cases}7x^2+x-8 & ,\ x<1\\ \operatorname{ctg}\frac{x+4}{\sqrt{11}}+3 & ,\ 1\le x\le 4\\ \sqrt{1+\lvert\cos^3 x\rvert} & ,\ x>4\end{cases}$
6.	$y=5e^{3x}-\begin{cases}3+\sin\lvert x\rvert & ,\ x\le -1\\ 2e^{\frac{x}{4}-1} & ,\ -1<x\le 3\\ 7-\sqrt{2}x^3 & ,\ x>3\end{cases}$
7.	$y=x^2\sin\frac{x}{2}+\begin{cases}\operatorname{arctg}e^x & ,\ x\le -5\\ 1+\frac{x^3}{4} & ,\ -5<x\le 0\\ \ln\lvert x\rvert-\frac{x}{5} & ,\ x>0\end{cases}$
8.	$y=2+6x+\begin{cases}\ln\cos x+x^5 & ,\ x\le 0\\ \operatorname{ctg}\frac{1+\ln x}{3} & ,\ 0<x\le 3\\ 12x-x^8 & ,\ x>3\end{cases}$
9.	$y=2\lvert x\rvert^3-\begin{cases}5\cos 18x & ,\ x\le -0{,}1\\ \operatorname{arctg}\frac{x+2}{5} & ,\ -0{,}1<x<1{,}2\\ \operatorname{ctg}x+18 & ,\ x\ge 1{,}2\end{cases}$
10.	$y=4{,}95x^2+\begin{cases}4+x^{-2} & ,\ x<-3{,}5\\ \operatorname{tg}\frac{3{,}5+x}{5} & ,\ -3{,}5\le x<1\\ \sin 3x-\cos x & ,\ x\ge 1\end{cases}$
11.	$y=2\lvert 5-x\rvert-\begin{cases}e^{\lvert 2+x\rvert} & ,\ x\le -1\\ \sin^2\frac{1}{\lvert 2+x\rvert} & ,\ -1<x<1\\ \frac{\cos^2 x}{1+\lvert\sin x\rvert} & ,\ x\ge 1\end{cases}$
12.	$y=\frac{2+x}{x^2}+1+\begin{cases}x^3-2x^4 & ,\ x<0\\ \left(\lvert x\rvert+e^x\right)^3 & ,\ 0\le x\le 2\\ 4\cos\left(x^2-2\right) & ,\ x>2\end{cases}$

Контрольні запитання

1. Який вигляд має складена програма?
2. Що таке порожня програма?
3. Яку функцію виконує кома як команда?
4. Що таке логічний вираз?
5. Які бувають логічні вирази?
6. Які є логічні операції?
7. Що таке команда розгалуження?
8. Які форми має команда розгалуження?
9. Як виглядає повна форма розгалуження?
10. Як виглядає коротка форма розгалуження?

ЛАБОРАТОРНА РОБОТА № 5. Програмування розгалужених алгоритмів: оператор SWITCH

Мета: *вивчити принципи роботи настанови багатовибірного розгалуження switch та вміти її застосовувати при розв'язуванні задач.*

Теоретичні відомості

Настанова багатовибірного розгалуження

Настанова **switch** – настанова багатовибірного розгалуження, яка дає змогу вибрати одну з множини альтернатив. Хоча різнонаправлене тестування можна реалізувати за допомогою послідовності вкладених **if**-настанов, однак у багатьох ситуаціях настанова **switch** виявляється набагато ефективнішим і очевидним рішенням.

Настанова багатовибірного розгалуження працює таким чином. Значення виразу послідовно порівнюється з константами із заданого переліку. Внаслідок виявлення збігу для однієї з умов порівняння здійснюється послідовність настанов, пов'язана з цією умовою.

Елемент ***вираз*** настанови **switch** під час обчислення має давати цілочисельне або символьне значення. Вирази, що мають, наприклад, тип з плаваючою крапкою, тут не дозволені. Дуже часто як керівний **switch**-вираз використовується одна змінна.

Настанова ***break*** завершує виконання коду програми, що визначається настановою ***switch.***

Загальний формат запису настанови **switch** є таким:

```
switch (вираз) {
  case  константа1:послідовність настанов break;
  case константа2:послідовність настанов  break;
  case константа3:послідовність настанов  break;
   ...
default:
послідовність настанов
}
```

Послідовності настанов **default**-гілки виконуються у тому випадку, якщо жодна із заданих **case**-констант не буде збігатися з результатом обчислення **switch**-виразу. Гілка **default** є необов'язковою. Якщо вона відсутня, то при не співпа-данні результату розрахунку виразу ні з однією з **case**-констант, ніякої дії виконано не буде. Якщо такий збіг все-таки виявиться, то виконуватимуться настанови, відповідні цій **case**-гілці доти, доки не трапиться настанова **break.**

Приклад 6

Записати програму, що виконує одну з операцій «+», «-», «*», «/» над заданими числами a, b.

```
#include <iostream.h>
#include <conio.h>
void main()
{ clrscr();
int f=1;
float a,b,rez;
char op;
cout<<"\n a=";cin>>a;
cout<<"\n b=";cin>>b;
cout<<"\n op=";cin>>op;
switch (op){
case '+':rez=a+b;break;
case '-':rez=a-b;break;
case '*':rez=a*b;break;
case '/':rez=a/b;break;
default: cout<<"\n  Error operation";f=0;
}
if (f==1) cout<<"\n rez="<<rez;
getch();
}
```

Рис. 10. Реалізація прикладу 6, використовуючи оператор ***switch***

Таблиця 7. Результати виконання програми, реалізованої оператором switch

Результати виконання програми				
Операція «+»	**Операція «-»**	**Операція «*»**	**Операція «/»**	**Помилкова операція**
a=45 b=55 op=+ rez=100_	a=100 b=50 op=- rez=50_	a=60 b=3 op=* rez=180	a=250 b=5 op=/ rez=50	a=25 b=5 op=t Error operation

Хід роботи

1. Запустити середовище С++.
2. Виконати програму в середовищі С++, яка розглянута в теоретичній частині (приклад 6).
3. Скласти й виконати програму, використовуючи **switch**.

Задача.

Нехай населені пункти позначені номерами від 1 до 8. Вартість одного квитка до конкретного пункту k визначається так:

$$cina = \begin{cases} 22, k = 1, \\ 25, k = 2,3,4 \\ 30, k = 5,6, \\ 35, k = 7,8. \end{cases}$$

Скільки коштуватимуть m квитків до населеного пункту, номер k якого вводять з клавіатури?

4. Скласти й виконати програму, використовуючи **switch**, задавши вхідні дані самостійно, згідно зі своїм порядковим номером в академічному журналі:

№	Завдання
1.	Скласти програму, яка б виводила на екран розклад роботи на тиждень.
2.	Скласти програму, яка б виводила на екран розклад роботи на день.
3.	За даним порядковим номером місяця вивести кількість днів.
4.	За даним порядковим номером місяця вивести пору року.
5.	За порядковим номером дня тижня вивести його назву.
6.	За парним номером місяця року, вивести його назву.
7.	За непарним номером місяця року, вивести його назву.
8.	За заданим порядковим номером місяця, вивести назву місяця.
9.	Дано ціле число у діапазоні [1,12], що означає місяць. Вивести на екран повідомлення про квартал. Наприклад; 3– перший квартал…
10.	Для заданих значень радіусів 10, 20, 40, 80 підрахувати площу круга та вивести на екран.
11.	Написати програму – абетку, яка на введену букву (малу або велику) виводить на екран відповідне повідомлення. Наприклад, уводимо ‘а’ або ‘А’ – антилопа.
12.	Дано ціле число у діапазоні [0,9]. Вивести на екран число прописом. Наприклад: 7 – сім…

Контрольні запитання

1. Що таке настанова **switch**?
2. Як працює **switch**?
3. Який загальний вигляд настанови **switch**?
4. Якого значення може набувати елемент виразу настанови **switch**?
5. Що означає **break** у настанові **switch**?
6. Як працює **default**-гілки у настанові **switch**?
7. Коли можна використовувати команду розгалуження **if**, а коли **switch**?
8. Чим відрізняється настанова **switch** від **if**?
9. У яких випадках слід використовувати оператор **switch**?
10. Чи є випадки, коли слід опустити оператор **break**?

ЛАБОРАТОРНА РОБОТА № 6. Циклічні програми

Мета: *вивчити дію операторів повторенння та їхнє практичне застосування*

Теоретичні відомості

Цикл (повторення) – це процес виконання певного набору команд деяку кількість разів. У мові С++ є три типи команди циклу – **for, while** та **do-while**.

Цикл **for** повторює вказану настанову задану кількість разів. Настанова **for** у мові програмування С++ діє практично так само, як настанова **for**, визначена в таких мовах програмування, як Pascal і Visual Basic. Її простий формат є таким:

for(*ініціалізація*; *логічний вираз*; *модифікації*) *настанова*;

У цьому записі елемент *ініціалізація* є настановою присвоєння, яка встановлює *керівній змінній циклу* початкове значення. Ця змінна діє як лічильник, який керує роботою циклу. Елемент *логічний вираз* є виразом, у якому тестується значення керівної змінної циклу. Результат цього тестування визначає, виконається цикл **for** ще раз чи ні. Елемент *модифікація* – вираз, який визначає, як змінюється значення керівної змінної циклу після кожної ітерації. Цикл **for** виконуватиметься доти, доки обчислення елемента *логічний вираз* дає істинний результат. Як тільки умова стане хибною, виконання програми продовжиться з настанови, що є наступною за циклом **for**.

Команда циклу з лічильником for:

for (<вираз 1>) ; <логічний вираз 2>; <вираз 3>) <команда 1>;

Вираз 1 призначений для підготовки циклу і виконується один раз. Переважно тут задають початкові значення змінних циклу (підготовляють цикл). У виразі 2 записують умову виходу із циклу. У виразі 3 — команди зміни параметрів циклу. Якщо за допомогою одного із виразів необхідно виконати декілька дій, то використовують команду «кома». Вирази 1 і 3 або один із них у команді **for** можуть бути відсутні**.** У цьому випадку опускати символ **«;»** не можна. Наприклад, **for (; i<10;) i++;.**

Дія команди:

1) обчислюються значення виразів 1 і 2;
2) якщо значення виразу 2 істинне, то виконується команда 1, якщо хибне – виконавець програми переходить до наступної після **for** команди;
3) обчислюються значення виразів 3 та 2 і перевіряється пункт 2).

Команда циклу з передумовою (while) має вигляд**:**

```
while (<вираз>) <команда 1>;
```

Дія команди:

1) обчислюються значення виразу. Якщо воно істинне, то переходимо до пункту 2), якщо хибне – до пункту 3);
2) виконується команда 1 і відбувається перехід до пункту 1);
3) відбувається перехід до наступної після **while** команди.

Виразом може бути довільний логічний вираз, стала або змінна цілого типу. Якщо треба перевірити декілька умов, то застосовують команду «кома». Команда 1 може бути порожньою, простою або складною.

На відміну від циклів **for** і **while**, у яких умова перевіряється під час входу, цикл **do-while** перевіряє умову при виході з циклу. Це означає, що цикл **do-while** завжди здійснюється хоча би один раз. Цикл **do-while** здійснюється доти, доки залишається істинним елемент <вираз>, який є умовним виразом. Ітераційна настанова **do-while** — єдиний цикл, який завжди робить ітерацію хоча б один раз.

Команда циклу з післяумовою (do-while) має вигляд**:**

```
do <команда 1>;
        while (<вираз>);
```

Дія команди:

1) виконується команда 1 і обчислюється значення виразу;
2) якщо значення виразу істинне, то див. пункт 1), якщо значення виразу хибне – відбувається перехід до наступної після **do-while** команди. Обчислюються значення виразу. Якщо воно істинне, то переходимо до пункту 2), якщо хибне – до пункту 3);
3) виконується команда 1 і відбувається перехід до пункту 1);

4) відбувається перехід до наступної після **while** команди.

Виразом може бути довільний логічний вираз, стала або змінна цілого типу. Якщо треба перевірити декілька умов, то застосовують команду «кома». Команда 1 може бути порожньою, простою або складною.

Приклад 7

Обчислити $\sum_{i=1}^{15} \frac{\sin 10i + \cos\frac{10}{i}}{\sqrt{i}}$.

Таблиця 8. Результати виконання циклічної програми

Результати виконання програми з використанням:	**Програмна реалізація**

Циклу з лічильником for:	`#include <iostream.h>`
	`#include <math.h>`
	`#include <conio.h>`
	`#define n 10`
	`void main()`
	`{`
	`clrscr();`
	`float s=0;`
	`for(int i=1;i<=n;i++)`
	`s+=(sin(10*i)+cos(10/i)/sqrt(i));`
	`cout<<"\n s="<<s;`
	`getch();`
	`}`

Циклу з передумовою:	`#include <iostream.h>`
	`#include <math.h>`
	`#include <conio.h>`
	`#define n 10`
	`void main()`
	`{`
	`clrscr();`
	`float s=0;int i=1;_`
	`while (i<=n)`
	`{s+=(sin(10*i)+cos(10/i)/sqrt(i));`
	`i++;}`
	`cout<<"\n s="<<s;`
	`getch();`
	`}`

Результати виконання програми з використанням:	**Програмна реалізація**

Циклу з після умовою:	`#include <iostream.h>`
	`#include <math.h>`
	`#include <conio.h>`
	`#define n 10`
	`void main()`
	`{ clrscr();`
	`float s=0;int i=1;`
	`do`
	`{`
	`s+=(sin(10*i)+cos(10/i)/sqrt(i));`
	`i++;`
	`}`
	`while (i<=n);`
	`cout<<"\n s="<<s; getch();}`

Результати виконання програми:	`s=-0.911013`

Хід роботи

1. Запустити середовище С++.
2. Виконати програму в середовищі С++, яка розглянута в теоретичній частині (приклад 7).
3. Скласти й виконати програму, використавши команди циклу з лічильником **for**, циклу з передумовою (**while**) та після умовою (**do-while**). Завдання вибрати згідно зі своїм порядковим номером в академічному журналі:

№	**Завдання**
1.	$\sum_{i=1}^{N} \frac{\sin i}{1+\cos i}$
2.	$\sum_{i=1}^{N} \frac{\cos \sin i}{1+\sin^2 i}$
3.	$\sum_{i=k}^{15} \frac{\cos i}{1+\sin^2 i}$

№	Завдання
4.	$\sum_{i=k}^{N} \frac{\sin\cos i}{1+\cos^2 i}$
5.	$\sum_{i=k}^{N} \frac{\sin i \cos i}{1+\sin^2 i}$
6.	$\sum_{i=N}^{20} \frac{\cos i + \sin i}{1+\cos i \sin i}$
7.	$\sum_{i=3}^{N} \frac{\cos\frac{i}{2} + \sin 2i}{1+\sin\cos i}$
8.	$\sum_{n=1}^{18} \frac{\sin\frac{1}{n^2}}{n+\sqrt[n]{\frac{1}{n^2}}}$
9.	$\sum_{i=1}^{15} \frac{\sin 10i + \cos\frac{10}{i}}{\sqrt{i}}$
10.	$\sum_{k=1}^{10} \left(\frac{\sin k^2}{k}\right)^k$
11.	$\sum_{k=1}^{N} \frac{1}{(2k+1)^2}$
12.	$\sum_{j=2}^{N} \frac{j(N-j)}{j^2+(N-j)^2}$

Контрольні запитання

1. Що таке цикл?
2. Які є команди циклу в С++?
3. Як організувати цикл за допомогою команди **for**?
4. Як працює команда циклу з передумовою?
5. Як організувати цикл, використовуючи команду з післяумовою?
6. Яка відмінність між циклом з переумовою та циклом з післяумовою?
7. Що таке ініціалізації у циклі з лічильником **for**?
8. Що таке модифікації у циклі з лічильником **for**?
9. Чи можна записувати декілька операторів в частині модифікації?
10. Чи можна записувати декілька операторів в частині ініціалізації?

ЛАБОРАТОРНА РОБОТА № 7. Ітераційні циклічні програми

Мета: *вивчити реалізацію ітераційних циклічних програм у середовищі C++*

Теоретичні відомості

Для виконання цього завдання необхідно використати теоретичні знання попередньої лабораторної роботи, що стосуються циклів з передумовою або циклів з післяумовою. Продемонструємо організацію ітераційних циклів на конкретному прикладі.

Приклад 8

Нехай $x-$ деяке число, яке необхідно ввести з клавіатури під час виконання програми, $e = 0.001-$ точність обчислень.

Обчислити суму елементів знакозмінного ряду $\sum_{n=1}^{\infty} a_n$,

де $a_n = (-1)^n \frac{(2x)^n}{n!}$.

Визначити кількість доданків. Вивести на екран результати обчислень.

Визначимо значення доданка при $n = 1$.

Отримаємо - $d = -2 * x.$

Обчислимо значення коефіцієнта, на який необхідно помножити попередній елемент, щоб отримати наступний - $m = \frac{a_{n+1}}{a_n} = -\frac{2 * x}{(n+1)}$.

Сумування будемо проводити до тих пір, доки не виконається така умова - $|d| < \varepsilon$.

Таблиця 9. Результати виконання ітераційної циклічної програми

Результати виконання програми з використанням:	**Програмна реалізація**

Циклу з передумовою (while)	`#include<iostream.h>`
	`#include<math.h>`
	`#include<conio.h>`
	`void main()`
	`{`
	`clrscr();`
	`const float e=0.0001;`
	`float x,d,m,s=0;`
	`int n=1;`
	`cout<<"\n x=";cin>>x;`
	`d=-2*x;`
	`while (fabs(d)>e)`
	`{s+=d;`
	`n++;`
	`m=-2*x/(n+1);`
	`d*=m;`
	`}`
	`cout<<"\n s="<<s;`
	`cout<<"\n Kil dod="<<n;`
	`}`

Цикл з післяумовою (do-while):	`#include<iostream.h>`
	`#include<math.h>`
	`#include<conio.h>`
	`void main()`
	`{`
	`clrscr();`
	`const float e=0.0001;`
	`float x,d,m,s=0;`
	`int n=1;`
	`cout<<"\n x=";cin>>x;`
	`d=-2*x;`
	`do`
	`{s+=d;`
	`n++;`
	`m=-2*x/(n+1);`
	`d*=m; }`
	`while (fabs(d)>e);`
	`cout<<"\n s="<<s;`
	`cout<<"\n Kil dod="<<n;`
	`}`

Результати виконання програми:	`x=5.67`
	
	`s=-1.823745`
	`Kil dod=35`

Хід роботи

1. Запустити середовище С++.
2. Виконати програму, розглянуту в теоретичній частині, в середовищі С++ (приклад 8), використавши команду циклу з післяумовою та з передумовою.
3. Виконати програму $\sum_{n=1}^{\infty}\frac{(-1)^{n+1}nx^n}{(n!)^2}$; 10^{-4},

   використавши команду циклу з післяумовою та з передумовою.
4. Скласти й виконати програму, використавши команду циклу з передумовою та з післяумовою, згідно зі своїм порядковим номером в академічному журналі:

№	Завдання
1.	$\sum_{n=1}^{\infty}\frac{\sin nx}{e^{n\cos x}n!}$; 10^{-4}
2.	$\sum_{n=1}^{\infty}\frac{(-2)^n\sqrt{n}}{x^{n+1}(n+1)!}$; 10^{-4}
3.	$\sum_{n=1}^{\infty}\frac{(-2)^n}{x^n n!}$; 10^{-3}
4.	$\sum_{n=1}^{\infty}\frac{x^{4n+1}}{(4n+1)!}$; 10^{-3}
5.	$\sum_{n=1}^{\infty}(-1)^n\frac{x^{4n+1}}{n!\cos(4n+1)}(-1)^n$; 10^{-3}
6.	$\sum_{n=1}^{\infty}(-1)^{n+1}\frac{x^{n+1}}{\cos n(n+1)!}$; 10^{-3}
7.	$\sum_{n=1}^{\infty}(-1)^{2n-1}\frac{x^{2n}\cos nx}{(2n+1)!}$; 10^{-3}
8.	$\sum_{n=1}^{\infty}\frac{x^{4n+1}}{\sin(4n+1)(4n+1)!}$; 10^{-3}
9.	$\sum_{n=1}^{\infty}\frac{x^{2n+1}}{\sqrt[n]{2}(2n)!}$; 10^{-3}
10.	$\sum_{n=1}^{\infty}\frac{2n-1}{n!2^n}$; 10^{-4}
11.	$\sum_{n=1}^{\infty}(-1)^n\frac{2^n}{n!}\cos\sqrt{\mathrm{nx}}$; 10^{-3}
12.	$\sum_{n=1}^{\infty}(-1)^{2n}\frac{x^{2n}\sqrt{\sin nx}}{(2n+2)!}$; 10^{-3}

Контрольні запитання

1. Що таке ітераційний цикл?
2. Як організувати ітераційний цикл?
3. Що таке цикл з передумовою?
4. Що таке цикл з післяумовою?
5. Що таке точність обчислень?
6. З чого слід розпочати реалізацію ітераційного циклу?
7. Як обчислити значення коефіцієнта?
8. Яка умови виходу для ітераційного циклу?
9. Як відбувається сумування знакозмінного ряду в ітераційному циклі?
10. Як на мові С++ реалізувати ітераційний цикл?

ЛАБОРАТОРНА РОБОТА № 8. Одновимірні масиви

Мета: *вивчити способи організації одновимірних масивів та їхнє застосування для розв'язування типових задач*

Теоретичні відомості

Масив — упорядкований набір фіксованої кількості однотипних елементів, що зберігаються у послідовно розташованих комірках оперативної пам'яті, мають порядковий номер і спільне ім'я, яке надає користувач.

Масив – це сукупність однотипних змінних з однаковим іменем. Кожен елемент масиву має власний індекс, за цим індексом і відбувається звернення до елементів. Нумерація індексів починається з 0. Кількість індексів визначає розмірність масиву. Розрізняють одно- та багатовимірні масиви. Наприклад, двовимірний масив даних – це таблиця, що складається з декількох рядків і стовпців. У математиці поняттю масив відповідають вектор і матриця.

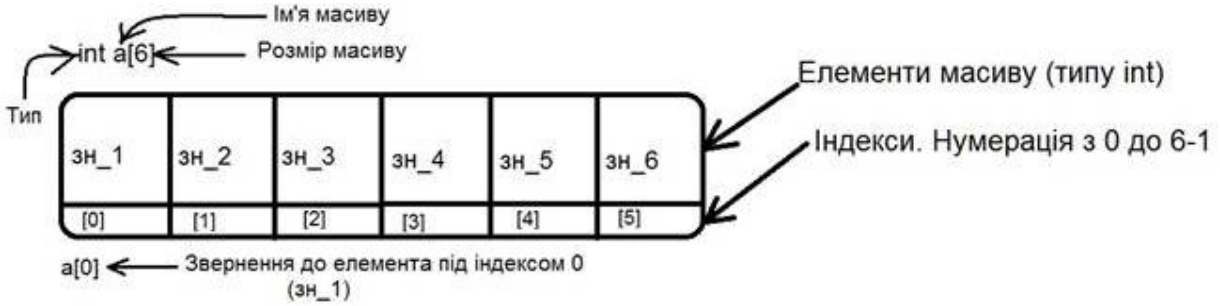


Рис. 11. Розміщення елементів одновимірного масиву в пам'яті комп'ютера

Елементи масиву у пам'яті розташовуються у порядку їх зростання, елемент за елементом. Крім цього кожний елемент має власний розмір, який дорівнює типу масиву. Якщо масив типу **int**, то кожен елемент буде займати 4 байти. А розмір всього масиву буде дорівнювати 4 * кількість елементів.

Створення масиву

Загальний вигляд конструкції опису одновимірного масиву такий:

<тип> <ім'я масиву> < [розмір]>

Розмір – це кількість елементів масиву. Розмір необхідно знати і задавати заздалегіть, оскільки компілятор має зарезервувати для нього необхідний обсяг пам'яті. Розміром може бути лише стала величина (не змінна).

Ім’я масиву у програмі змінювати не можна – це стала величина, яка містить адресу першого елемента. Отже, назва масиву – це вказівник на його перший елемент.

Тип – це будь–який із відомих типів (**int**, **bool**, **char**, **double**). Наприклад, команда **int st[5]** оголошує масив з іменем **st**, який складається із п’яти цілих чисел; команда **float rist[10]** оголошує масив **rist,** який містить десять чисел дійсного типу; **char sym[6]** оголошує масив із шести символів.

Звернутись до елементів масиву можна двома способами: за допомогою імені масиву або використовуючи вказівники.

Нумерація елементів масиву завжди починається з *нуля.* Щоб звернутись до деякого елемента, необхідно зазначити ім’я масиву, а у квадратних дужках – його номер. Наприклад, змінна stud[2] є третім елементом масиву stud, а stud[4] – п’ятим, оскільки масив stud має елементи stud[0], stud[1], stud[2], stud[3] та stud[4].

Назва масиву stud є вказівником на його перший елемент. Змінна *stud містить значення першого елемента масиву (елемента stud[0]). Оскільки всі елементи масиву розміщені у послідовних комірках оперативної пам’яті комп’ютера, то вказівник (stud + 1) вказуватиме на другий елемент масиву (зміщення відносно вказівника stud на одну одиницю пам’яті), а вказівник (stud + 4) - на п’ятий (зміщення на чотири одиниці).

Проініціалізувати масив (надати значення елементам масиву) можна одним із способів:

- використовуючи принцип замовчування;
- безпосередньо під час його оголошення;
- застосовуючи команду присвоєння;
- під час уведення даних із клавіатури.

За замовчуванням усім елементам масиву надається значення 0. Масив можна ініціалізувати повністю або частково відразу під час його оголошення, записуючи значення змінних через кому у фігурних дужках. Наприклад,

int Stud[] = {2, 10, 5, 7, 3);

float rist[10] = (163.4, 154.6, 170, 172.8};

Перші чотири елементи масиву rist були проініціалізовані, а решта – ні. Якщо масив повністю ініціалізують під час оголошення, то його розмір зазначати не обов'язково. У цьому випадку компілятор сам визначає, скільки пам'яті необхідно зарезервувати. У наведеному прикладі масив Stud складатиметься з п'яти цілих чисел.

Надати значення іншим елементам масиву rist або змінити значення вже проініціалізованих елементів можна командою присвоєння, наприклад, так: rist[3] = 175.4, rist[9] = 184.1 або так: *(rist+2) = 164.5, *(rist +7) = 148.0 тощо. Елементи масиву також можна вводити з клавіатури під час виконання програми, як це робимо для змінних простих стандартних типів.

Масиви – сталі (значення яких змінювати у програмі не можна) оголошують так: **const int** flag = {1, 2}. Сталі масиви треба ініціалізувати під час оголошення, інакше елементам масиву автоматично будуть присвоєні нульові значення.

Для опрацювання елементів масиву найчастіше використовують команду циклу **for**, хоча можна застосувати і **while** або **do-while**.

Наприклад, створити масив з перших ста цілих чисел і обчислити суму всіх його значень можна одним із способів:

```
//1-й спосіб
int n[100], S = 0;
for (k = 0; k < 100;)
{    *(n+k) = ++k;
        S += *(n+k);     }
}
// 2-й спосіб
i nt n[ 100], S = 0;
   for (k = 0; k < 100; k++) {
   n[k] = k++; S += n[k];
```

Задачі відшукання в масиві конкретних даних розв'язуть методом перебору всіх елементів масиву за допомогою циклу й умовної команди, де зазначають умову пошуку потрібних даних.

Динамічне оголошення масивів

Під час компіляції програмного коду для статично оголошених масивів надається пам'ять. Для ефективного використання пам'яті призначене динамічне оголошення масивів, а саме:

<тип вказівника> *<назва> = new <тип змінної>[<кількість>];

Після виконання цієї команди буде виділена неперервна ділянки пам'яті обсягом **sizeof**(тип змінної) * <кількість> , і назва масиву вказуватиме на початок цієї ділянки.

З динамічною змінною можна виконувати операції, визначені для даних відповідного базового типу.

Після опрацювання масиву вивільнити пам'ять можна за допомогою команди **delete[]** <назва вказівника на масив даних>**.** Під час вивільнення пам'яті розмір масиву зазначати не потрібно.

Можна використати динамічний розподіл пам'яті, наприклад:

```
int *n = new int [100];      // Виділяємо пам'ять для ста цілих чисел
for (int S = O, k = 0; k < 100;)
      {// Опрацьовуємо масив
      *(n+k) = ++k;
      S += *(n+k);      }
delete[]n; // Вивільняємо пам'ять
```

За допомогою динамічних змінних можна розв'язати задачу почергового опрацювання одною програмою деякої кількості великих масивів (якщо всі масиви не можливо одночасно ввести у пам'ять). Задачу розв'язують так. Створюють масив, наприклад, *mas1 = **new** <тип>[<кількість>] і опрацьовують динамічні змінні*mas1, *(mas1+1), ... Вивільняють пам'ять **delete[]** mas1. Створюють й опрацьовують елементи другого масиву *mas2 = **new** <тип> [<кількість>] і т.д.

Для того, щоб отримати ціле випадкове число з діапазону від 0 до n, можна скористатися функцією random(n), яка описана у модулі stdlib.h. Для того, щоб під час виконання програми кожен раз отримувати різні значення, перед використанням функції random() треба записати функцію randomize().

При роботі з масивами найчастіше трапляються такі умови обробки елементів масиву:

Таблиця 10. Умови обробки елементів масиву

Умови	Реалізація умови на С++
Парні елементи масиву	$A[i]\%2==0$
Непарні елементи масиву	$A[i]\%2!=0$
Елементи масиву кратні k	$A[i]\%k==0$
Елементи масиву не кратні k	$A[i]\%k!=0$
Елементи масиву, що стоять на парних місцях	$i\%2==0$
Елементи масиву, що стоять на непарних місцях	$i\%2!=0$
Додатні елементи масиву	$A[i]>0$
Від'ємні елементи масиву	$A[i]<0$
Елементи масиву, що є в інтервалі (n_1,n_2)	$(A[i]>n_1 \&\& A[i]<n_2)$

Приклад 9

Утворити масив з 10-ти елементів за допомогою генератора випадкових чисел (числа з діапазону від 0 до 20). Обчислити кількість елементів масиву , що попадають в інтервал $(1,7)$.

Таблиця 11. Результати виконання програми (приклад 9)

Програмна реалізація	Результати виконання програми:
1 спосіб – звернення до елементів масиву за іменем масиву та порядковим номером елемента в масиві:	

```
#include <iostream.h>
#include <conio.h>
#include <math.h>
#include <stdlib.h>
void main()
{  clrscr(); const int n=10;
randomize();
int y[n],kil=0,k;
for (k=0; k<=n;k++)
y[k]=random(20);
for (k=0; k<=n;k++)
if (y[k]>1 && y[k]<7)kil++;
for (k=0; k<=n;k++)
cout<<"\n"<<y[k];
cout<<"\n kil="<<kil;
}
```

```
17
0
5
13
16
3
12
0
18
15
1
 kil=2
```

Програмна реалізація	Результати виконання програми:
2 спосіб – використання динамічного розподілу пам’яті:	

```
#include <iostream.h>
#include <conio.h>
#include <math.h>
#include <stdlib.h>
void main()
{  clrscr(); const n=10;
randomize();
int *y=new int[n],kil=0;
for (int k=0; k<n;k++)
{  *(y+k)=random(20);
  cout<<"\n y["<<k<<"]="<<*(y+k);  }
for (k=0; k<n;k++)
if (*(y+k)>1 && *(y+k)<7)kil++;
cout<<"\n \n kil="<<kil;
delete[]y;_
}
```

```
y[0]=18
y[1]=6
y[2]=14
y[3]=4
y[4]=8
y[5]=4
y[6]=18
y[7]=11
y[8]=17
y[9]=19

kil=3
```

Хід роботи

1. Запустити середовище С++.
2. Виконати програму в середовищі С++, розглянуту в теоретичній частині (приклад 9).
3. Утворити масив з 20-ти елементів за допомогою генератора випадкових чисел. Числа вибрати з діапазону від 0 до 40. Виконати завдання згідно з порядковим номером у журналі групи. Масив опрацювати двома способами:
   - через ім’я масиву та порядковий номер елемента масиву (спосіб 1);
   - з використанням динамічного розподілу пам’яті (спосіб 2).

№	Завдання
1.	У одновимірному масиві знайти середнє арифметичне парних елементів, що розміщені на непарних місцях.
2.	У одновимірному масиві знайти добуток елементів, що діляться на «3» і не діляться на «7».
3.	У одновимірному масиві знайти мінімальний елемент і поміняти його місцями з першим від’ємним елементом масиву.

№	Завдання
4.	У одновимірному масиві знайти максимальний елемент і поміняти його місцями з першим додатнім елементом масиву.
5.	У одновимірному масиві знайти максимальний елемент і поміняти його місцями з другим від'ємним елементом масиву.
6.	У одновимірному масиві знайти середнє арифметичне максимального та мінімального елементів масиву.
7.	У одновимірному масиві знайти максимальний та мінімальний елементи масиву та поміняти їх місцями.
8.	У одновимірному масиві знайти максимальний парний елемент масиву.
9.	У одновимірному масиві знайти суму додатніх елементів масиву індекс яких кратний 3.
10.	У одновимірному масиві знайти суму додатніх і середнє арифметичне від'ємних елементів масиву.
11.	У одновимірному масиві знайти суму додатніх елементів масиву у яких індекс кратний 2.
12.	У одновимірному масиві знайти середнє геометричне парних елементів масиву.

Контрольні запитання

1. Що таке масив?
2. Як є різновиди масивів?
3. Як створити масив?
4. Який загальний вигляд конструкції опису одновимірного масиву?
5. Як можна звернутися до елементів масиву?
6. Які типи даних можна використовувати в масивах?
7. Як розпочинається нумерація елементів масиву?
8. Що означає «проініціалізувати масив»?
9. Як оголошувати масиви – сталі?
10. Що таке динамічне оголошення масиву?

ЛАБОРАТОРНА РОБОТА № 9. Багатовимірні масиви

Мета: *вивчити способи організації багатовимірних масивів і їхнє застосування для розв'язування типових задач*

Теоретичні відомості

Якщо елемент масиву має не один, а декілька індексів, то такі масиви називаються ***багатовимірними.***

Загальний вигляд конструкції опису багатовимірного *(N* – вимірного) масиву такий:

<тип> <ім'я масиву>[<p_1>][<p_2>]... [<p_N>],

де p_1, p_2,..., p_N задають розміри для кожного виміру.

Кількість індексів визначає вимірність масиву: двовимірні масиви мають два індекси, тривимірні – три і т.д. Усі багатовимірні масиви можна розглядати й опрацьовувати як одновимірні. Наприклад, тривимірний масив s[5][20][30] можна інтерпретувати як п'ять масивів розміром 20×30, а інші — як 20 одновимірних масивів, які містять по 30 елементів.

Детальніше охарактеризуємо двовимірні масиви.

Елементи двовимірного масиву визначаються іменем масиву та двома індексами: перший індекс означає номер рядка, інший – номер стовпця, на перетині яких розміщений елемент. Нумерація індексів масиву завжди починається з нуля.

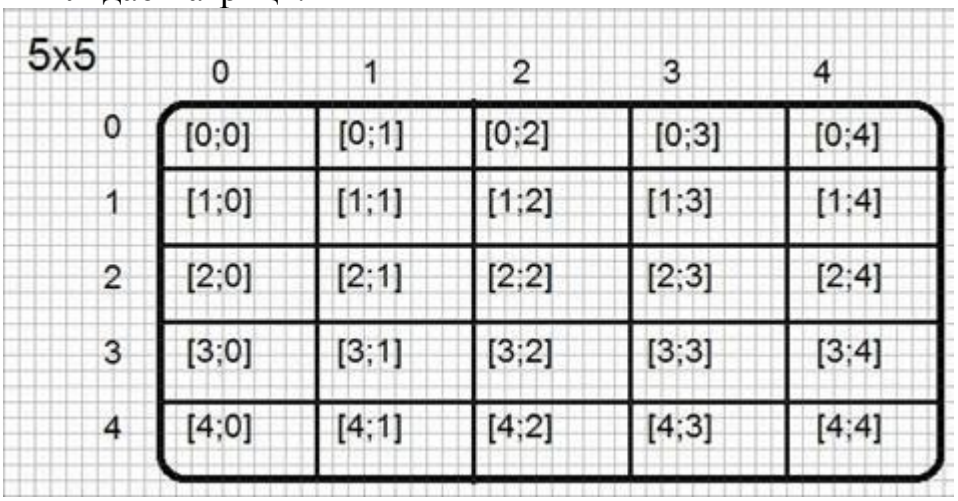
Двовимірні масиви компілятор розглядає як послідовність одновимірних. Тому до елементів двовимірного масиву, як і для одновимірних, можна також звертатись через вказівники. У такому випадку це вказівник на вказівник одномірного масиву*:*

***(*(<назва вказівника>+<зміщення за рядками>)+< зміщення за стовпцями>)**

Наприклад, елемент *(*(doba+2)+15) розміщений на перетині 3-го рядка та 16-го стовпця.

Такі масиви можна уявляти і як просту таблицю, наприклад, таблиця MS Excel, кожна її клітинка також має два параметри.

Ось так виглядає матриця:



5x5	0	1	2	3	4
0	[0;0]	[0;1]	[0;2]	[0;3]	[0;4]
1	[1;0]	[1;1]	[1;2]	[1;3]	[1;4]
2	[2;0]	[2;1]	[2;2]	[2;3]	[2;4]
3	[3;0]	[3;1]	[3;2]	[3;3]	[3;4]
4	[4;0]	[4;1]	[4;2]	[4;3]	[4;4]

Рис. 12. Нумерація індексів двовимірного масиву

Але у пам’яті комп’ютера вона буде виглядати так:

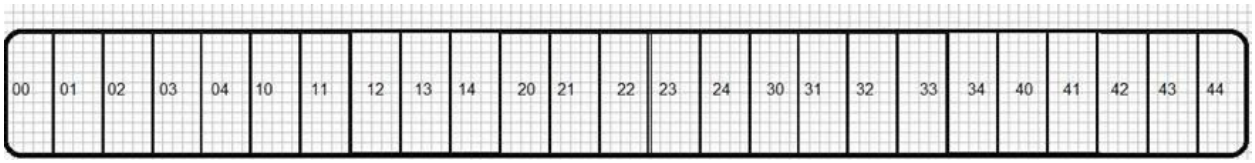


Рис. 13. Розміщення елементів двовимірного масиву у пам’яті ПК

Під час оголошення двовимірні масиви можна частково або повністю ініціалізувати.

Оголосимо й проініціалізуємо двовимірний масив цілих чисел

int ball[2][3] = {4, 5, 3, 3, 5, 2}.

У такому випадку елементам надаються значення так:

ball[0][0] = 4, ball[0][1] = **5,** ball[0][2] = 3, ball[1][0] =3, ball[1][1] = **5,** ball[1][2] = 2.

Двовимірні масиви автоматично ініціалізуються «за рядками», тобто спочатку модифікується зовнішній (правіший) індекс. Надавати значення елементам масиву можна і так: **int ball[2][3] = {{**4**,** 5, 3**}, {**3, 5, 2**}};**

або так:

```
int ball[2][3] = {4, 5, 3,
                  3, 5, 2};
```

Приклад 10

Скласти програму для занесення в двовимірний масив ***p*** таблиці множення двох чисел і виведення масиву на екран. Доступ до елементів масиву здійснювати за іменем масиву, індексом рядка та стовпця, тобто за допомогою конструкції *p*[*i*][*j*].

```
#include <iostream.h>
#include <math.h>
#include <conio.h>
void main()
{
clrscr();
const int n=9;int p[n][n];
for (int i=0;i<n;i++)
{ for (int j=0;j<n;j++)
  {p[i][j]=(i+1)*(j+1);
   cout<<p[i][j]<<"\t";  }
cout<<"\n";}
getch();}
```

Рис. 14. Реалізація прикладу 10

```
1  2  3  4  5  6  7  8  9
2  4  6  8  10 12 14 16 18
3  6  9  12 15 18 21 24 27
4  8  12 16 20 24 28 32 36
5  10 15 20 25 30 35 40 45
6  12 18 24 30 36 42 48 54
7  14 21 28 35 42 49 56 63
8  16 24 32 40 48 56 64 72
9  18 27 36 45 54 63 72 81
```

Рис. 15. Результати виконання прикладу 10

Існує ще один спосіб створення масиву – ввід його елементів з клавіатури, зокрема:

```
{
const int n = 5;
int mas[n][n];
for (int i=0; i<n;i++)
{    for (int j=0; j<n; j++)
     cin>> mas[i][j];  }
}
```

Щоб оголосити масив **int** mas[24][60]. Він містить елементи цілого типу і складається з 24 рядків і 60 стовпців. Елемент doba[23][59] розміщений на перетині останнього 24-го рядка і 60 стовпця (нумерація індексів масиву починається з нуля). Через вказівники звернення до того самого елемента виглядає наступним чином – *(*(doba+23)+59).

Приклад 11

Знайти максимальний та мінімальний елементи двовимірного масиву *a* та поміняти їх місцями. Доступ до елементів масиву здійснювати через вказівники. Вивести на екран вихідний масив у вигляді таблиці.

```
#include <iostream.h>
#include <conio.h>
#include <math.h>
void main()
{  clrscr(); const int n=4;
int i,j,r1,s1,r2,s2,z,max,min;
 int y[n][n]={{4,13,-6,9},{67,-9,11,123},{3,7,8,99},{4,8,55,2}};
max=**y;min=**y;
for (i=0; i<n;i++)   {
for (j=0; j<n;j++)
cout<<*(*(y+i)+j)<<"\t";cout<<"\n";  }
for  (i=0; i<n;i++)
for  (j=0; j<n;j++)
{ if  (*(*(y+i)+j)>max) {max=*(*(y+i)+j);r1=i;s1=j;}
if  (*(*(y+i)+j)<min) {min=*(*(y+i)+j);r2=i;s2=j;}}
cout<<"\n max="<<max<<"\n min="<<min;
z=*(*(y+r1)+s1); *(*(y+r1)+s1)=*(*(y+r2)+s2);  *(*(y+r2)+s2)=z;
cout<<"\n";
for (i=0; i<n;i++) {
for (j=0; j<n;j++)
cout<<*(*(y+i)+j)<<"\t"; cout<<"\n";  } }
```

Рис. 16. Програмна реалізація прикладу 11

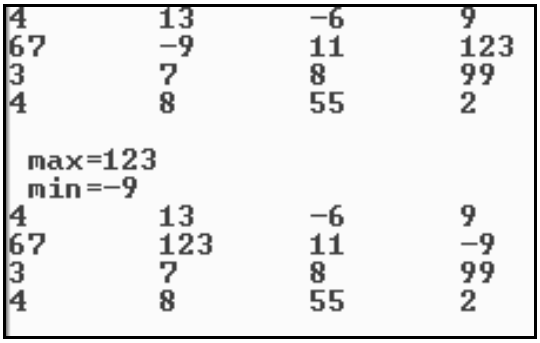


Рис. 17. Результати виконання прикладу 11

Хід роботи

1. Запустити середовище С++.
2. Виконати програми в середовищі С++, розглянуті в теоретичній частині (приклади 10, 11).
3. Вибрати подані завдання згідно з порядковим номером у журналі групи. Доступ до елементів масиву здійснити двома способами:
   - за іменем масиву, індексом рядка та стовпця, тобто за допомогою конструкції $a[i][j]$;
   - через вказівники.

№	Завдання
1.	Написати програму, яка шукає кількість елементів матриці $A(k \times k)$, більших від заданого числа С.
2.	Написати програму, яка шукає мінімальний елемент матриці $A(k \times k)$.
3.	Написати програму, яка шукає суму максимального та мінімального елементів матриці $A(k \times k)$.
4.	Написати програму, яка шукає кількість елементів матриці $A(k \times k)$, значення яких є в діапазоні $[a,b]$. Числа a,b увести з клавіатури.
5.	Написати програму, яка шукає суму елементів допоміжної діагоналі матриці $A(k \times k)$.

№	Завдання
6.	Написати програму, яка шукає суму елементів головної діагоналі матриці $A(k \times k)$.
7.	Написати програму, яка міняє місцями максимальний та мінімальний елементи матриці $A(k \times k)$.
8.	Написати програму, яка шукає середнє арифметичне елементів матриці $A(k \times k)$, що діляться на 3 і не діляться на 5.
9.	Написати програму, яка шукає середнє арифметичне елементів головної діагоналі матриці $A(k \times k)$.
10.	Написати програму, яка шукає середнє геометричне елементів матриці $A(k \times k)$, розміщених у парних рядках.
11.	Написати програму, яка шукає суму додатніх елементів матриці $A(k \times k)$, розміщених у парних стовпцях.
12.	Написати програму, яка шукає максимальний елемент головної діагоналі матриці $A(k \times k)$ та мінімальний елемент допоміжної діагоналі.

Контрольні запитання

1. Що таке багатовимірний масив?
2. Який загальний вигляд конструкції опису багатовимірного масиву?
3. Що визначає вимірність масиву?
4. Як організувати двовимірні масиви?
5. Навести приклади двовимірних масивів.
6. Які елементи визначають двовимірний масив?
7. З якого числа розпочинається нумерація масиву?
8. Як розглядає компілятор двовимірні масиви?
9. Як ініціалізувати двовимірні масиви?
10. Чи можна до двомірного масиву звертатись через вказівники?

ЛАБОРАТОРНА РОБОТА № 10. Функції користувача

Мета: *освоїти способи організації функцій користувача, уміти практично застосовувати.*

Теоретичні відомості

Мова програмування С++ дає можливість реалізувати концепцію структурного аналізу алгоритмів. *Структурний аналіз* полягає у попередньому опрацюванні громіздкого алгоритму та поділі його на окремі простіші частини. У С++ ці частини реалізуються за допомогою *функцій.* Окремі функції об'єднують у спільну програму. У відкомпільованому вигляді така програма утворює модуль. Головна функція, що обов'язково входить до кожної програми, – main(). Розрізняють стандартні функції та функції користувачів. Стандартні функції мови описані у бібліотеках. До таких функцій, зокрема, належать математичні функції $\sin(x)$, $\cos(x)$ з бібліотеки math.h, функція очистки екрана clrscr() з бІбліотеки conio.h та багато інших.

Функція користувача – це поіменована група команд, яка оголошена у файлі заголовків (або в основній програмі) та описана у модулі (в основній програмі). До функції можна звертатись (викликати) з будь-якого місця програми необхідну кількість разів.

Оголошення функцій користувача

Кожну функцію користувача перед першим викликом передусім необхідно оголосити. За стандартом ISO/ANSI прототипи функцій оголошують у спеціальних файлах заголовків. У програму ці файли приєднують за допомогою директиви **#include.**

Функцію користувача оголошують так:

<тип функції> <назва функції> (<список формальних параметрів>);

де тип функції — це тип даного, який функція повертає в основну програму. Тип функції можна не зазначати. За замовчуванням функція повертає у програму дане цілого типу **int.** Функцію, яка не повертає у програму жодного результату, оголошують з типом **void.** Для функції, яка не залежить від жодних параметрів, у круглих дужках записують службове

слово **void.**

Назву функції надає користувач за правилом створення ідентифікаторів.

У списку формальних параметрів через кому записують змінні, зазначаючи їхні типи. Тип необхідно зазначати для кожної змінної окремо. Імена змінних можна опускати. Якщо функція не набуває жодних значень, то список формальних параметрів може бути відсутній. Круглі дужки опускати не можна.

Розглянемо такі сигнатури функцій:

1) **float** Suma(**int** kil, **float** cina) – оголошена функція Suma типу **float,** яка залежить від двох змінних: перша змінна цілого типу **int,** друга – типу **float;**
2) **void dil(float, float) –** оголошена функція dil залежить від двох змінних дійсного типу **float** і не повертає у програму жодного значення;
3) kod(**int** k1, **int** k2) **–** оголошена ціла функція kod залежить від двох змінних цілого типу;
4) **double** loto(**void**) – оголошена дійсна функція loto типу **double,** яка не залежить від жодних параметрів.

Під час оголошення можна відразу ініціалізувати формальні параметри функції, тобто надавати їм певних значень. Такі значення називаються *значеннями за замовчуванням.* Значення таких параметрів у програмі можна змінювати. Враховуючи це, розглянемо ще один спосіб оголошення функцій.

1) **float** Suma(**int** kil, **float** cina = 2.5);
2) **void** dil(**float** v = 1.2, **float** n = 3);
3) kod(**int** k1, int k2 = 5).

Опис функцій користувача

Опис функції складається із заголовка *без крапки з комою* і тіла функції, записаного у фігурних дужках, а саме:

<тип функції> <назва функції>(<список формальних параметрів>)
{ <тіло функції>;
return (<назва змінної 1>); }

У тілі функції записують команди, які задають дію функції. Результат виконання функції повертається в основну програму (у точку виклику) за допомогою змінної 1 командою return. Тип змінної 1 має збігатися з типом функції. У тілі функцій типу **void** команду return не зазначають. У команді return круглі дужки можна не писати.

Зауваження 1. Функцію можна описувати і на початку програми. У такому випадку декларувати її не потрібно.

Наприклад, функцію Suma можна описати так:

```
float Suma(int kil, float cina)     // Заголовок функції
    { float s = kil * cina;   return (s);}
```

Якщо потрібно проініціалізувати значення змінних, що входять у функцію, то це можна зробити у сигнатурі функції. Тоді в описі у заголовка функції значення за замовчуванням не задають.

Сигнатуру функції, яка обчислює периметр *k* – кутної правильної фігури зі стороною r, можна оголосити так:

```
float perymetr (int к = 4, float r = 2.5);
```

а описати функцію так:

```
float perymetr (int k, float r)
    { float p;  p = k * r;
      return (p);}
```

Виклик функцій користувача

До функції користувача звертаються з розділу команд основної програми (функції main()) або з іншої функції. Виклик функцій можна виконати двояко: або командою виклику, або з виразів так:

<назва функції>(<список фактичних параметрів>)

Список фактичних параметрів може містити сталі, змінні, посилання, вказівники, вирази. Списки формальних і фактичних параметрів мають бути узгодженими за типами та кількістю елементів. Якщо у списку формальних параметрів є проініціалізовані змінні, то у списку фактичних параметрів ці змінні можуть бути відсутні, їм будуть надані значення за замовчуванням.

Розглянемо сигнатуру попереднього прикладу.

float perymetr (**int** k = 4, **float** r = 2.5);

До цієї функції можна звернутись одним із способів:

1) perymetr (7, 2.8);
2) perymetr (8);
3) perymetr ().

У першому випадку змінній *k* буде присвоєне значення 7, а змінній *r* – 2,8. У другому випадку *k* = 8, г = 2,5. Для третього випадку *k* = 4, *r* = 2,5.

Приклад 12

Дано дійсні s, t. Обчислити $v = \frac{h(a/b,1,b)+h(b,1,ab)}{h(1.7,ab,9)}$,

де $h(x,y,z) = \frac{x+y+z}{xy}$.

```
#include <iostream.h>
#include <math.h>
#include <conio.h>
float h(float x,float y,float z)
{
float v;
v=(x+y+z)/(x*y);
return(v);
}
void main()
{
clrscr();
float a,b,rez;
cout<<"\n a=";cin>>a;
cout<<"\n b=";cin>>b;
rez=(h(a/b,1,b)+h(b,1,a*b))/h(1.7,a*b,9);
cout<<"\n rez="<<rez;
}
```

Рис. 18. Програмна реалізація прикладу 12

```
a=3.4
b=6.75
rez=24.271826
```

Рис. 19. Результати виконання прикладу 12

Хід роботи

1. Запустити середовище С++.
2. Виконати програму в середовищі С++, розглянута в теоретичній частині (приклад 12).
3. Скласти програми обчислення заданих виразів з використанням функцій користувача. Виконати нижче подані завдання згідно з порядковим номером у журналі групи.

№	Завдання
1.	Дано дійсні p, q. Обчислити $\frac{k(1+pq,q^2)+k^2(p,p^2)}{1+k(pq+q^2,p)}$, де $k(x,y)=\frac{x}{1+\sin^2 y}+\frac{y}{1+x^2}$
2.	Дано дійсні p, q. Обчислити $k^2\left(p+\sqrt{q},q-\sqrt{p}\right)-k(1,p+q)$, де $k(x,y)=\frac{x}{\lvert x^3+y^3\rvert}+\frac{y}{\lvert x+y\rvert}$
3.	Дано дійсні p, q. Обчислити $\frac{k(1+p^2,1-q^2)-k^2(1,pq)}{1+k(pq,1)}$, де $k(x,y)=\frac{\sin x}{x^2+y^2}+\frac{\cos y}{1+\lvert xy\rvert}$
4.	Дано дійсні p, q. Обчислити $\frac{k^2(1+p,q^2)-k(qp,1)}{1+k(p^2,q)}$, де $k(x,y)=\frac{\sin x}{y^2}+\frac{\cos y}{x^2}$

5.	Дано дійсні s, t. Обчислити $\dfrac{h^4(s,t)+h(1,s^2+t^2)}{1+h^2(st,1)}$, де $h(a,b)=\dfrac{a}{b^2+1}+\dfrac{1}{a^2+b^2}$
6.	Дано дійсні s, t. Обчислити $\dfrac{h(s^2,t^2)+h^2(s+t,1)}{1+h^2(st,2)}$, де $h(x,y)=\dfrac{xy}{1+x^2y^2}$
7.	Дано дійсні s, t. Обчислити $f(a,b,c)=\dfrac{\sin^2 abc}{a^2+b^2+c^2}$, де $\dfrac{f(1,t^2,s)+f(t,s^2,1)}{1+f^2(1,ts,1)}$,
8.	Дано дійсні s, t. Обчислити $\dfrac{g(s^2,t+1)+g(t^2,s+1)}{1+g^2(s+t,st)}$, де $g(a,b)=\dfrac{\sin ab}{a^2+b^2}$
9.	Дано дійсні s, t. Обчислити $h(1,s+t)+h(s,s-t)-h(t,\sqrt{s^2+t^2})$, де $h(x,y)=\dfrac{x^2+y^2}{1+x^2+x^2y^2}$
10.	Дано дійсні s, t. Обчислити $\dfrac{f(t,s,2)+f(1,s+t,t-s)}{1+f^2(1,1,t^2+s^2)}$, де $f(a,b,c)=\dfrac{a+b+c}{\sin^2 ab+c^2}$

11.	Дано дійсні s, t. Обчислити $\frac{g(1,s)+(1+g^2(t,1))^2}{1+g^3(s+t,1)}$, де $g(a,b)=a^2+ab+b^2$
12.	Дано дійсні s, t. Обчислити $\frac{f(1,t+s,s)+f(t,st,1)}{1+f^2(s,1,t)}$, де $f(a,b,c)=a\sin b+b\sin a+c^2$

Контрольні запитання

1. Що таке структурний аналіз алгоритмів?
2. Як у С++ реалізуються функції?
3. Що таке функції користувача?
4. Як оголошується функція користувача у С++?
5. Що таке сигнатура функції?
6. Дане якого типу функція повертає у програму за замовчуванням?
7. Що таке значення за замовчуванням?
8. Як викликати функцію користувача?
9. Як узгоджуються списки формальних і фактичних параметрів?
10. З якого місяця програми можна викликати функцію користувача?

ЗРАЗОК ОФОРМЛЕННЯ ЗВІТУ ПРО ВИКОНАНУ РОБОТУ

Звіт про виконану лабораторну роботу № _____

Ітераційні циклічні програми

студента групи _____

Петренка Петра

Завдання 1.

Запустити середовище С++.

Виконання завдання 1

Запустили середовище програмування С++.

Завдання 2

Обчислити суму елементів знакозмінного ряду $\sum_{n=1}^{\infty} a_n$, де $a_n = (-1)^n \frac{(2x)^n}{n!}$. Визначити кількість доданків. Вивести на екран результати обчислень. Виконати програму в середовищі С++, використавши команду циклу з післяумовою та з передумовою.

Виконання завдання 2

Задано $x-$ деяке число, яке необхідно ввести з клавіатури під час виконання програми.

$e = 0.001-$ точність обчислень.

Значення доданка при $n = 1$:

$d = -2 * x.$

Значення коефіцієнта, на який необхідно помножити попередній елемент, щоб отримати – $m = \frac{a_{n+1}}{a_n} = -\frac{2 * x}{(n+1)}$.

Сумування будемо проводити до тих пір, доки не виконається умова – $|d| < \varepsilon$.

Фрагмент програми, використовується команда циклу з передумовою (while) :

```
#include<iostream.h>
#include<math.h>
#include<conio.h>
void main()
{
clrscr();
const float e=0.0001;
float x,d,m,s=0;
int n=1;
cout<<"\n x=";cin>>x;
d=-2*x;
while (fabs(d)>e)
{s+=d;
n++;
m=-2*x/(n+1);
d*=m;
}
cout<<"\n s="<<s;
cout<<"\n Kil dod="<<n;
}
```

Результати виконання програми:

```
x=5.67

s=-1.823745
Kil dod=35
```

Фрагмент програми,

використовується команду циклу з післяумовою (do-while):

```
#include<iostream.h>
#include<math.h>
#include<conio.h>
void main()
{
clrscr();
const float e=0.0001;
float x,d,m,s=0;
int n=1;
cout<<"\n x=";cin>>x;
d=-2*x;
do
{s+=d;
n++;
m=-2*x/(n+1);
d*=m; }
while (fabs(d)>e);
cout<<"\n s="<<s;
cout<<"\n Kil dod="<<n;
}
```

Результати виконання програми:

```
x=5.67

s=-1.823745
Kil dod=35
```

Завдання 3

Виконати програму $\sum_{n=1}^{\infty}\frac{(-1)^{n+1}nx^n}{(n!)^2}$; 10^{-4},

використавши команду циклу з післяумовою та з передумовою.

Виконання завдання 3

Задано $x-$ деяке число, яке необхідно ввести з клавіатури під час виконання програми.

$e=0.001-$ точність обчислень.

Значення доданка при $n=1$:

$d=x.$

Значення коефіцієнта, на який необхідно помножити попередній елемент, щоб отримати $m=\frac{a_{n+1}}{a_n}=-\frac{x}{(n+1)n}.$

Сумування будемо проводити до тих пір, доки не виконається умова – $|d|<\varepsilon$.

Фрагмент програми, використовується команда циклу з передумовою (while) :

```
#include<iostream.h>
#include<math.h>
#include<conio.h>
void main()
{
clrscr();
const float e=0.0001;
float x,d,m,s=0;
int n=1;
cout<<"\n x=";cin>>x;
d=x;
while (fabs(d)>e)
{s+=d;
n++;
m=-x/(n*(n+1));
d*=m; }
cout<<"\n s="<<s;
cout<<"\n Kil dod="<<n;
}
```

Результати виконання програми:

```
x=5.76

s=2.24874
Kil dod=9_
```

Фрагмент програми,

використовується команда циклу з післяумовою (do-while):

```
#include<iostream.h>
#include<math.h>
#include<conio.h>
void main()
{
clrscr();
const float e=0.0001;
float x,d,m,s=0;
int n=1;
cout<<"\n x=";cin>>x;
d=x;
do
{s+=d;
n++;
m=-x/(n*(n+1));
d*=m; }
while (fabs(d)>e);
cout<<"\n s="<<s;
cout<<"\n Kil dod="<<n;
}
```

Результати виконання програми:

```
x=5.76

s=2.24874
Kil dod=9_
```

Висновок: ми отримали навички роботи з ітераційними циклічними програмами в середовищі програмування С++.

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Лєхан С.А.Інформатика. Мова програмування С++: навчальний посібник. – Шепетівка : – «Аспект», – 2007р. – 310 с.
2. Павловская Т.А. С/С++ Программирование на языке высокого уровня. – СПб.: Питер, 2008. – 461 с.: ил.
3. С/С++. Структурное программирование: Практикум / Т.А. Павловская, Ю.А. Щупак – СПб.: Питер, 2007. – 239 с.: ил.
4. Бьерн Страуструп. Язык программирования С++. Второе дополненное издание / Страуструп Бьерн. – М., БИНОМ, 2015. – 1136 с.
5. Вайсфельд Мєтт. Обьектно-ориентированный подход: Java, .Net, С++ / Метт Вайсфельд. – М. : КУДИЦА-ОБРАЗ, 2005.
6. Гилберт С. Самоучитель Visual С++ в примерах / С. Гилберт, Б. Маккарти. – К. : ТИД ДС, 2003.
7. Глинський Я.М., Анохін В.Є., Ряжська В.А. С++ і С++ Builder: Навч. посібн. 5-те вид.– Львів: СПД Глинський, 2011. – 192 с.
8. Глушков С.В. Язык программирования С++ / С.В. Глушаков, А.В. Коваль, С.В. Смирнов. – Харьков : Фолио, 2003.
9. Дудзяний І.М. Програмування мовою С++. Частина 1 : Парадигма процедурного програмування: навчальний посібник / І.М. Дудзяний. – Львів : ЛНУ імені Івана Франка, 2013. – 468 с.
10. Наконечний С.І. Математичне програмування : навч. посібник / С.І. Наконечний, С.С. Савіна. – К. : КНЕУ, 2003.
11. Нікольський Ю.В. Дискретна математика : підручник / Ю.В. Нікольський, В.В. Пасічник, Ю.М. Щербина. – К. : BHV, 2007.
12. Шилдт Г. С++: базовый курс. 3-е издание / Г. Шилдт. – М.: Вильямс, 2010.

Навчальне видання

Любов Лазурчак, Тетяна Вдовичин

ІНФОРМАТИКА: С++

Головний редактор
Ірина Невмержицька

Редактор
Іванна Біблій

Технічний редактор

Коректор
Оксана Бульбах

Здано до набору 14. 07. 2014 р. Підписано до друку 22. 09. 2014 р.
Формат 60х90/16. Папір офсетний. Гарнітура. Times. Наклад 300 прим.
Ум. друк. арк. 8,37. Зам. № 275

Видавничий відділ Дрогобицького державного педагогічного університету імені Івана Франка (Свідоцтво про внесення суб'єкта видавничої справи до державного реєстру видавців, виготівників і розповсюджувачів видавничої продукції ДК № 2155 від 12. 04. 2005 р.) 82100 Дрогобич, вул. І.Франка, 24, к.42, тел. 2 – 23 – 78.