МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДНІПРОДЗЕРЖИНСЬКИЙ ДЕРЖАВНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ

до практичних занять

з дисципліни: **«Обчислювальна техніка та програмування»**
(Частина 2)

для студентів, що навчаються за напрямами:

6.050701 – «Електротехніка та електротехнології»;
6.050702 – «Електромеханіка»

Затверджено редакційно-видавничою
секцією науково-методичної ради ДДТУ
«___» _____ *2014*р., протокол № __

Дніпродзержинськ
2014

Методичні вказівки до практичних занять з дисципліни «Обчислювальна техніка та програмування» (Частина 2) для студентів, що навчаються за напрямами: 6.050701 – «Електротехніка та електротехнології»; 6.050702 – «Електромеханіка» / Укладачі О.О.Жулковський, І.І.Жулковська.– Дніпродзержинськ: ДДТУ, 2014.– 96с.

Укладачі:
кандидат технічних наук, доцент **Жулковський О.О.,**
кандидат технічних наук, доцент **Жулковська І.І.**

Рецензент:
завідувач кафедри прикладної математики
доктор технічних наук, професор **Самохвалов С.Є.**

Відповідальний за випуск: кандидат технічних наук, доцент Жулковський О.О.

Затверджено на засіданні кафедри інформатики та комп'ютерних технологій
(протокол № *5* від *21 травня 2014* р.)

Коротка анотація видання. Містить короткі теоретичні відомості щодо елементів та основних прийомів програмування алгоритмічною мовою високого рівня С (С++); сприяє оволодінню основами структурної методології програмування, обчислювальними можливостями мови, керуванням порядком обчислень, організацією та обробкою основних типів даних, реалізацією файлового обміну даними тощо.

ЗМІСТ

ВСТУП

Дисципліна «Обчислювальна техніка та програмування» викладається для студентів, що навчаються за напрямами 6.050701 – «Електротехніка та електротехнології» і 6.050702 – «Електромеханіка», упродовж двох навчальних семестрів. Дане навчальне видання призначене для другого навчального семестру.

Робочою мовою програмування при виконанні практичних робіт обрано популярну, потужну і затребувану алгоритмічну мову високого рівня С, яка забезпечує формування найбільш ефективного машинного коду програми, що досягається прив'язкою мови до структури пам'яті та регістрової архітектури ЕОМ. Популярність мови С та похідних від неї мов програмування (С++, Objective-C, C# тощо) підтверджується також відомим світовим критерієм – індексом TIOBE (TIOBE programming community index).

Мова програмування С також є більш придатною для розв'язку практичних задач, пов'язаних з майбутньою професійної діяльністю у галузі електроніки.

Усі сучасні версії компілятора С відповідають ANSI-стандарту, запропонованому Американським національним інститутом стандартів (ANSI – American National Standard Institute). Не є виключенням і популярний компілятор Borland C++ для IBM PC фірми Borland Software Corporation, програмне середовище якого саме і застосовується для виконання наведених у даному виданні практичних робіт.

Практичні роботи передбачають ознайомлення з основами структурної методології програмування мовою С (С++), її обчислювальними можливостями, керуванням порядком обчислень, організацією та обробкою основних типів даних, реалізацією файлового обміну даними тощо, що дає можливість подальшого самостійного поглиблення своїх знань та навичок у програмуванні мовами високого рівня.

1. Загальні положення

Метою практичних занять з дисципліни «Обчислювальна техніка та програмування» є набуття практичних навичок алгоритмізації і програмування алгоритмічною мовою високого рівня під час розв'язку прикладних задач з використанням прикладного програмного забезпечення та сучасної обчислювальної техніки (персональних комп'ютерів – ПК).

Методичні вказівки складено у відповідності до робочої програми з дисципліни «Обчислювальна техніка та програмування» для студентів, що навчаються за напрямами 6.050701 – «Електротехніка та електротехнології» і 6.050702 – «Електромеханіка».

Практичні роботи виконується на практичних заняттях з дисципліни в обчислювальному залі з використанням ПК та в часи самостійної роботи.

Для студентів денної форми навчання виконання всіх практичних робіт даного навчального видання є обов'язковим. Студенти заочної форми навчання виконують лише роботи, передбачені робочою програмою навчальної дисципліни.

Перелік і кількість запропонованих до розв'язку задач визначається викладачем, який веде практичні заняття, відповідно до робочої програми дисципліни.

Усі практичні роботи складено за єдиним планом – на початку кожної роботи наведено її тему та мету, далі викладено короткі теоретичні відомості щодо тематики роботи з розглядом прикладів розв'язку типових задач, після чого містяться індивідуальні завдання та запитання для самоперевірки.

Наприкінці методичних вказівок наведено порядок виконання і захисту робіт та перелік посилань на використану укладачами навчального видання та рекомендовану для виконання роботи літературу.

Після виконання кожної роботи студенти складають звіт, котрий захищають. За результатами виконання та захисту роботи виставляються бали за спеціальною шкалою оцінювання, наведеною у робочій програмі. Бали, отримані за окремі роботи, формують загальну суму балів за практикум, яка враховується у підсумкову оцінку за модуль та семестр.

2. Основні вимоги з техніки безпеки

При виконанні практичних робіт у комп’ютерних залах студент повинен дотримуватися означених нижче правил техніки безпеки:

1. Не приступати до роботи, не прослухавши інструктаж з техніки безпеки.
2. Зареєструватись у журналі користувачів ПК у комп'ютерному залі (перед початком виконання першої роботи).
3. Не вмикати чи вимикати самостійно живлення у комп'ютерному залі або на своєму робочому місці.
4. При виконанні роботи за комп'ютером може знаходитись не більш, ніж два студенти одночасно.
5. Не змінювати налагоджень комп'ютера та його операційної системи.
6. Не встановлювати та не видаляти ніяких програм без дозволу викладача або відповідального інженера.
7. Закінчивши виконання практичної роботи студент повинен скопіювати результати на власний носій інформації, закрити всі програми, з якими працював, без збереження на ПК власних результатів.
8. Повідомити відповідального інженера та викладача про закінчення роботи.

3. Практична робота **№1.**
Засоби розробки С-програм

***Мета роботи*:** ознайомлення з процесом розробки програм у інтегрованому середовищі та за допомогою автономного компілятора Borland C++.

Теоретичні основи.

Для розробки С-програми найбільшу популярність набули інтегроване середовище розробки (далі IDE – Integrated Development Environment) та автономний компілятор фірми Borland Software Corporation.

Windows-орієнтоване середовище Borland C++ (файл **bcw.exe**) має текстовий редактор, компілятор, налагоджувач, компонувальник, менеджер проектів, довідкову систему тощо та може бути застосовано під час розробки різноманітних DOS- і Windows-додатків, у тому числі й 32-бітових.

Розділи головного меню IDE мають таке ж саме призначення, що й аналогічні розділи практично усіх Windows-орієнтованих середовищ, наприклад, того ж Borland Pascal або Delphi.

Так, для відкриття нового вікна редагування необхідно після запуску файла bcw.exe обрати команду головного меню IDE File|New|Text Edit. У цьому вікні вводиться текст програми, яка спочатку має ім’я noname00.cpp (рис. 3.1).

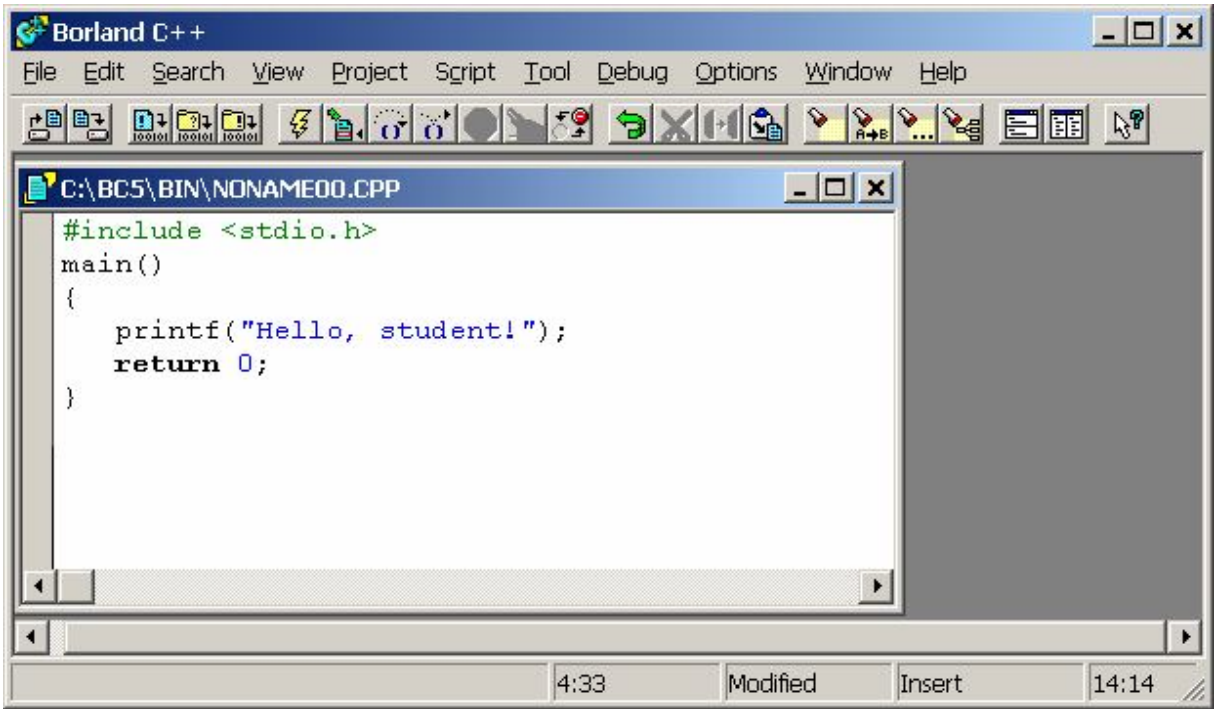


Рисунок 3.1 – Вікно IDE Borland C++

Щоб зберегти введену програму на диску необхідно скористатися командою головного меню IDE File|Save або File|Save as… або відповід-

ною кнопкою на оперативній панелі (SpeedBar).

За допомогою команди головного меню IDE File|Open… або відповідної кнопки на оперативній панелі (SpeedBar) можна завантажити у вікно редагування будь-який текстовий файл. Зазвичай це файли з розширеннями .c, .cpp або .h.

Якщо у вікні редагування клацнути один раз правою клавішею миші, відкриється оперативне меню (SpeedMenu), в якому необхідно обрати команду, що відкриває діалогове вікно TargetExpert (рис. 3.2).

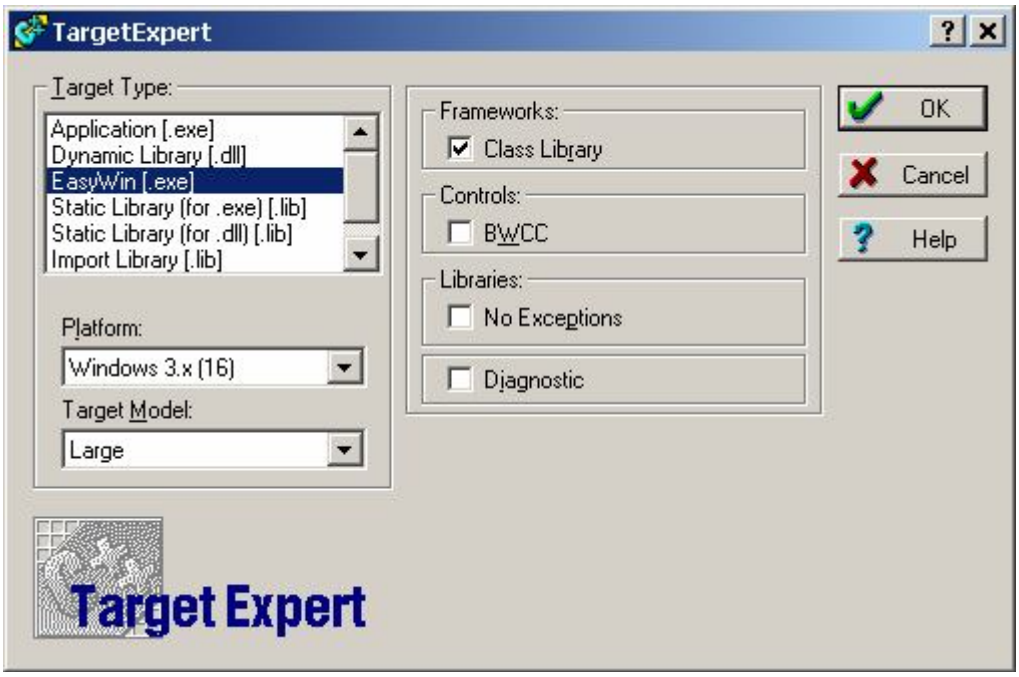


Рисунок 3.2 – Діалогове вікно TargetExpert IDE Borland C++

Наприклад, для компіляції лістингу програми для Windows необхідно у вікні TargetExpert встановити такі ж самі установки, як показано на рис. 3.2 та натиснути кнопку OK або клавішу Enter. Після завершення діалогу з вікном TargetExpert для компіляції та запуску програми необхідно натиснути комбінацію клавіш Ctrl+F9 або скористатися відповідною кнопкою на оперативній панелі (SpeedBar).

Щоб створити консольний додаток для Windows необхідно у вікні TargetExpert встановити Target Type у стан Application [.exe], а Platform – у Win32 та використати модель пам’яті Console (за замовчанням), а далі виконати ті ж самі дії, що й під час компіляції Windows-програми.

Під час компіляції програми IDE відкриває вікно повідомлень (Message) про помилки (Errors), попередження (Warnings), якщо такі є, тощо. Усі помилки та попередження виділяються у вікні редагування, а коментар щодо них міститься у вікні повідомлень. Коментар про помилку містить ім’я файлу (що дуже важливо для багатофайлових програм), номер рядка з

помилкою та коротке її пояснення. Текст хибної програми має бути відредагованим та заново скомпільованим, без чого неможливо створення виконавчого exe-файлу.

У разі успішної компіляції на екрані з'явиться вікно на кшталт зображеного на рис. 3.3.

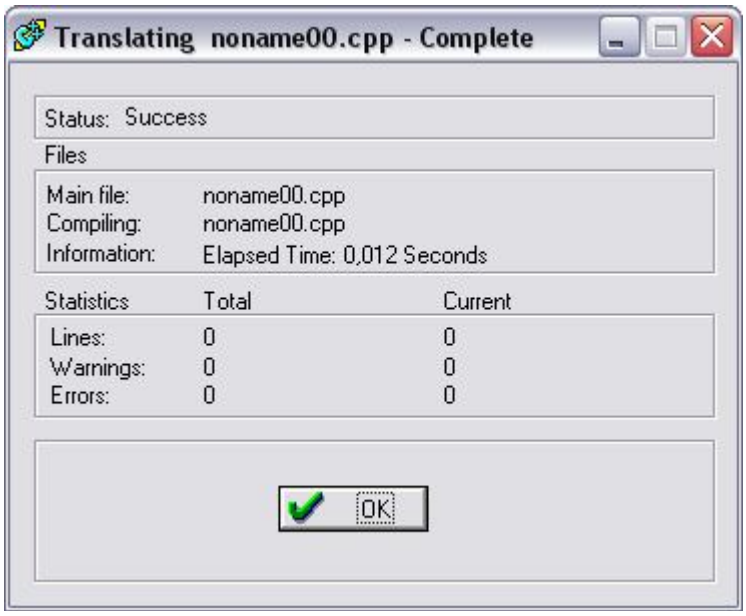


Рисунок 3.3 – Вікно статусу компіляції IDE Borland C++

Для отримання оперативної довідки про будь-які елементи мови програмування достатньо виділити у вікні редагування IDE Borland C++ необхідний елемент (для цього можна клацнути на елементі два рази лівою клавішею миші) та натиснути клавішу F1. Можна також встановити курсор на необхідний елемент програми та натиснути комбінацію клавіш Ctrl+F1.

Автономний компілятор Borland C++ (файл **bcc.exe**) запускається із командного рядка DOS і застосовується для компіляції програм, створених за допомогою будь-якого текстового редактора. Компілятор bcc здатний виконати компіляцію і компонування окремих модулів у виконавчий файл однією командою. Так, наприклад, аби скомпілювати та скомпонувати програму name.c необхідно у командному рядку записати:

bcc name.c

У разі успішної компіляції та компонування на диску буде створено файл name.exe.

При цьому для С-програм необхідно обов'язково зазначати як ім'я, так і розширення (.c) текстового файла, бо без зазначення розширення текстовий файл буде компілюватися начебто програма C++, тобто як файл з розширенням .cpp.

Окрім можливості компіляції С- та С++-програм, а також їх автоматичного компонування автономний компілятор Borland C++ здатний компонувати об'єктні модулі (файли з розширенням .obj) та асемблювати програми на асемблері (файли з розширенням .asm), а також змішувати файли з різними розширеннями в одній команді. Наприклад, командний рядок:

bcc main.c sub1 sub2.obj

компілює main.c у main.obj, а sub1.cpp (розширення береться за замовчанням) у sub1.obj і автоматично викликає програму **tlink.exe** аби скомпонувати обидва отримані obj-файли з третім – sub2.obj для створення результуючого файла main.exe.

Комплект поставки Borland C++ містить також 32-бітові автономний компілятор (файл bcc32.exe) і компонувальник (файл tlink32.exe) для створення 32-бітових Windows-додатків.

Завдання:

Наступна програма-гра генерує випадкове натуральне число від 0 до 9 та пропонує користувачеві його вгадати, вводячи з клавіатури. У разі вгаданого числа програма завершується та повідомляє про набрані користувачем бали. Чим скоріше генероване число буде вгадане, тим більше балів отримає гравець.

З використанням інтегрованого середовища Borland C++ створити виконавчий файл для наведеної програми-гри:

```
/* 1.1 */
#include <stdio.h>
#include <stdlib.h>
main()
{
 const del=10;
 int x,y,prize=100;
 printf(″Define number, which computer generate\n″);
 randomize();
 y=random(10);
 do
 {
   printf(″Enter number=″);
   scanf("%i",&x);
   if (x>y)
   {
     printf(″Wrong, enter smaller\n″);
     prize-=del;
```

```
    }
    if (x<y)
    {
      printf(″Wrong, enter bigger\n″);
      prize-=del;
    }
  }
  while (x!=y);
  printf(″You have guessed right!\n″);
  printf(″You have won the prize=%i″,prize);
  return 0;
}
```

Розв'язок:

1. Знайти на диску файл інтегрованого середовища Borland C++ (bcw.exe) та запустити його на виконання (зазвичай цей файл знаходиться за маршрутом C:\BCx\BIN\bcw.exe, де x – версія IDE Borland C++, або може бути активований за допомогою свого ярлика на робочому столі чи через меню команд кнопки Пуск ОС Windows).

2. У середовищі Borland C++, що з'явиться на екрані, відкрити вікно noname00.cpp за допомогою меню IDE File|New|Text Edit. У цьому вікні вводиться текст програми, яка спочатку має ім'я noname00.cpp

3. У вікні noname00.cpp натиснути праву клавішу миші та в оперативному меню (SpeedMenu), що відкриється, обрати меню TargetExpert, де встановити Target Type у стан EasyWin [.exe], Platform – у Windows 3.x (16), а Target Model – у Large для компіляції Windows-програми (або інші настроювання – у разі створення консольного додатку Windows тощо).

4. У вікні noname00.cpp з використанням редактора IDE набрати текст наведеної у прикладі завдання С-програми.

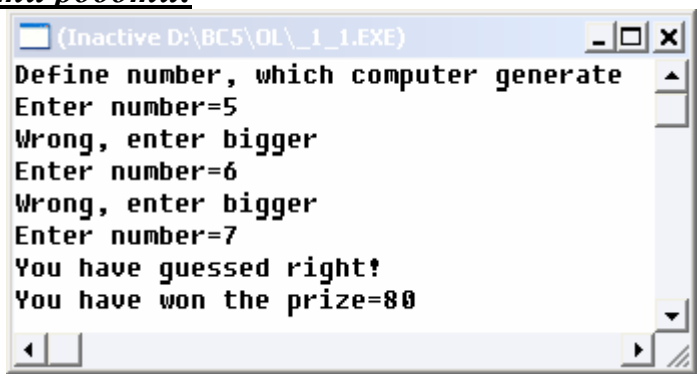
5. За допомогою меню IDE File|Save або File|Save as… зберегти набрану програму на диску під власним іменем, наприклад _1_1.cpp

6. Скомпілювати програму за допомогою меню IDE Project|Compile (натиснувши на клавіатурі Alt+F9) або за допомогою відповідної кнопки на оперативній панелі інструментів. Виправити припущені помилки та знову перейти до виконання п.5.

Уразі успішної компіляції з'явиться вікно статусу компіляції з повідомленням Status: Success, та автоматично на диску буде створено файл _1_1.exe. Місцезнаходження цього файлу визначається меню IDE Options|Project…|Output Directories.

7. Запустити програму на виконання за допомогою відповідної кнопки на оперативній панелі інструментів або натиснувши на клавіатурі Ctrl+F9. Програму можна також виконати у традиційний спосіб у Windows, вийшовши із середовища Borland C++.

Результати роботи:



Контрольні запитання:

1. Якими засобами можна розробляти програми C/C++?
2. За допомогою якого файлу активується інтегроване середовище розробки програм (IDE) Borland C++?
3. Яка структура та призначення IDE Borland C++?
4. Яка структура та призначення головного меню IDE Borland C++?
5. Які основні етапи розробки програми у IDE Borland C++?
6. Що таке компіляція програми? Що означає термін «компонування програми»?
7. Яким чином виконуються компіляція та запуск програми у IDE?
8. Що означає термін «автономна компіляція програми»? За допомогою якого файлу активується автономний компілятор Borland C++ і як їм користуватись?
9. Яке розширення мають файли програм C/C++?
10. Які файли з’являються на диску після компіляції програм C/C++?

4. Практична робота **№2.**
Створення простих програм

***Мета роботи*:** набуття навичок створення простих програм у середовищі Borland C++; ознайомлення з алфавітом мови програмування C/C++, використанням коментарів, змінних та констант, а також функцій введення-виведення.

***Теоретичні основи*.**

Найкоротша коректна С-програма має наступний вигляд:

```
main()
{
   return 0;
 }
```

Ця програма містить лише одну головну функцію **main()**.

Відкрита та закрита фігурні дужки обмежують блок функції main(). Зазвичай у цьому блоці розташовуються оператори програми, у тому числі виклики інших функцій. У даному випадку головна функція містить лише один оператор – оператор повернення return, який завершує виконання програми та повертає деяке ціле значення. Ненульові значення оператора return свідчать про помилки у програмі.

Більш складні С-програми можуть містити різні стандартні оголошення або виклики стандартних (бібліотечних) функцій. У такому разі до програми необхідно долучити спеціальні заголовні файли. Наприклад, програма на ANSI C, що містить стандартні оголошення, пов’язані із введенням або виведенням інформації, має починатися директиви включення:

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

Для організації введення-виведення засобами Borland C++ замість наведеного рядка програми записують:

```
#include <iostream.h>
```

Ім’я директиви вказує компілятору на читання тексту з заданого файлу (у даному випадку з файлів stdio.h або iostream.h).

Коментарі ANSI С-програми обмежуються складеними символами: /* та */. Такі коментарі можуть використовуватись у будь-якому місці програми, а також займати декілька її рядків. Коментарі Borland C++ починаються з пари символів // і все, що знаходиться далі, аж до кінця поточного рядка, є коментарем.

Приклади коментарів:

```
/* Звичайний коментар  ANSI C …*/
// Коментар Borland C++ …
```

Ідентифікатори використовуються для присвоєння імен змінним, константам, функціям програми тощо. Ідентифікатори мають починатися з літери і можуть містити виключно літери, цифри та символ підкреслення у будь-якому сполученні. Бажано не використовувати символ підкреслення з самого початку ідентифікатора, бо такі ідентифікатори зарезервовано для системного рівня Borland C++. Довжина ідентифікаторів не обмежена, проте компілятор за замовчанням відрізняє лише перші 32 символи.

Мова C/C++ є чутливою до регістру літер ідентифікаторів, тому, наприклад, наступні ідентифікатори: myVar, myvar або MYVAR – цілком різні ідентифікатори.

Не припустимо у якості ідентифікаторів використання ключових слів мови програмування:

asm	**default**	**for**	**short**	**union**
auto	**do**	**goto**	**signed**	**unsigned**
break	**double**	**if**	**sizeof**	**void**
case	**else**	**int**	**static**	**volatile**
char	**enum**	**long**	**struct**	**while**
const	**extern**	**register**	**switch**	
continue	**float**	**return**	**typedef**	

Borland C++ розширює наведений список власними зарезервованими словами.

Ключові слова C/C++ записуються виключно малими літерами.

Оголошення змінних у програмі здійснюється за наступними схемами:

ім’я_типу ідентифікатор;
ім’я_типу ідентифікатор_1, …, ідентифікатор_n;

наприклад:

```
int name; /* оголошення цілої змінної на ім’я name */
float a,b,c; /* оголошення декількох дійсних змінних */
```

Змінні, що оголошені у функції main(), є локальними і недоступними з інших функцій. Змінні, що оголошені поза функцією main() та інших функцій, є глобальними і доступними з будь-якого місця програми.

Локальні змінні перед використанням необхідно ініціалізувати за допомогою оператора присвоєння, тобто необхідно задавати їх стартові зна-

чення, наприклад, наступним чином:

```
int name;
name=255;
```

або

```
int name=255;
```

Неініціалізовані глобальні змінні автоматично дорівнюють нулю.

Для оголошення у програмі констант використовується ключове слово const, наприклад:

```
const i=0;
const int p=1,s=0;
```

Подальше змінення значень констант у програмі не припустимо.

Стандартні функції **printf()** і **scanf()**, прототипи яких містяться у заголовному файлі stdio.h, здійснюють у ANSI C форматне виведення і введення даних відповідно.

Функція виведення printf() має наступні команди (специфікатори) формату, які починаються з символу **%**:

%c	– символ;
%d, %i	– ціле десяткове число;
%e, %E	– число у форматі x.xx e+xx або x.xx E+xx відповідно;
%f, %g, %G	– десяткове дійсне число;
%o	– вісімкове число;
%s	– рядок символів;
%u	– ціле десяткове число без знаку;
%x, %X	– шістнадцяткове число (наприклад, 7f або 7F відповідно);
%%	– символ %;
%p, %n	– вказівник.

Окрім того, до окремих команд формату можуть бути застосовані модифікатори **l** і **h**, наприклад: %ld – друк long int, %hu – друк unsigned short, %lf – друк long double тощо.

Між символом % і форматом команди може знаходитись ціле число, що зазначає найменше поле, що виділяється для друку. Для зазначення кількості десяткових знаків при виведенні дійсного числа після цілого числа ставлять крапку та число знаків.

Вирівнювання друку відбувається за лівим краєм поля. Для вирівнювання за правим краєм необхідно за символом % поставити символ «мінус».

Приклади:

```
printf(″Result=%d\n″,x);
```

```
printf(″%–5.2f″,y);
```

де ″\n″ – керуючий символ переведення рядка, тобто після виведення на екран значення змінної (у даному разі x) положення курсору переміститься на наступний рядок (керуючі символи розглянуто у практичній роботі №7).

Введення інформації з клавіатури супроводжується відображенням її на екрані. У разі, коли функція виведення scanf() має декілька параметрів, дані потрібно вводити послідовно через пробіл, символ табуляції або нового рядка. Після введення необхідно натиснути клавішу Enter.

Функція введення scanf() автоматично перетворює введені дані у зазначений формат:

%c	– читання символу;
%d, %i	– читання десяткового цілого числа;
%e, %f, %g, %E, %G	– читання числа типу float;
%h	– читання числа типу short int;
%o	– читання вісімкового числа;
%s	– читання рядка символів;
%x	– читання шістнадцяткового числа;
%p, %n	– читання вказівника.

У функції scanf() також припустимо застосування модифікатору формату l, наприклад %lE – читання дійсного числа типу long double.

Усі змінні функції scanf() мають бути зазначені з префіксом **&**.

Приклади:

```
scanf(″%d″,&x);
scanf(″%i%f%i″,&a,&b,&c);
```

У Borland C++ для виведення даних використовується ідентифікатор потоку виведення **cout** і оператор **<<**, а для введення даних – ідентифікатор потоку введення **cin** і оператор **>>**, наприклад:

```
cin>>x;
cout <<″Result=″<<x;
```

<u>Приклад завдання 2.1:</u>

З використанням середовища Borland C++ створити виконавчий файл для наступної програми:

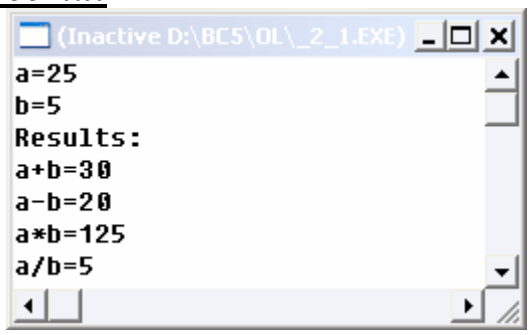
```
/* 2.1 */
#include <stdio.h>
main()
{
```

```
  const a=25;
  int b;
  printf("a=%i\n",a);
  printf("b=");
  scanf("%i",&b);
  printf("Results:\n");
  printf("a+b=%i\n",a+b);
  printf("a–b=%i\n",a–b);
  printf("a*b=%i\n",a*b);
  printf("a/b=%i",a/b);
  return 0;
}
```

Розв'язок:

Необхідно виконати послідовність дій, що наведено у прикладі розв'язку завдання до практичної роботи №1.

Результати роботи:



Завдання:

З використанням інтегрованого середовища Borland C++ створити виконавчий файл для наступної програми (варіант див. у табл. 4.1):

Таблиця 4.1 – Завдання до практичної роботи №2

1.	/* 2.1 */
	#include <stdio.h>
	main()
	{
	float time,way;
	printf("Calculation of speed\n");
	printf("Input time:\n");
	printf("time=");
	scanf("%f",&time);
	printf("Input way:\n");
	printf("way=");
		2.	/* 2.2 */
			#include <stdio.h>
			main()
			{
			float v1,v2;
			printf("Calculation of ");
			printf("total volume\n");
			printf("Input volume:\n");
			printf("volume_1=");
			scanf("%f",&v1);
			printf("volume_2=");

```
scanf("%f",&way);
printf("speed:\n");
printf("v=%5.2f",way/time);
return 0;
}
```

```
scanf("%f",&v2);
printf("Total volume=");
printf("%5.2f",v1+v2);
return 0;
}
```

3.

```
/* 2.3 */
#include <stdio.h>
main()
{
const tk=273;
long int tc;
printf("Celsius temperature: ");
scanf("%ld",&tc);
printf("Translate (+273)\n");
printf("Kelvin temperature: ");
printf("%ld",tc+tk);
return 0;
}
```

4.

```
/* 2.4 */
#include <stdio.h>
main()
{
float s;
printf("Calculation of volume\n");
printf("side=");
scanf("%f",&s);
printf("Volume:\n");
/* Results */
printf("V=%5.2f",s*s*s);
return 0;
}
```

5.

```
/* 2.5 */
#include <stdio.h>
main()
{
float t1,t2;
printf("Calculation of ");
printf("total temperature\n");
printf("Input temperatures:\n");
printf("t1=");
scanf("%f",&t1);
printf("t2=");
scanf("%f",&t2);
printf("Total temperature=");
printf("%5.2f",(t1+t2)/2);
return 0;
}
```

6.

```
/* 2.6 */
#include <stdio.h>
main()
{
const sp=32;
int a;
printf("Hello!\n");
printf("Enter whole number:");
scanf("%d",&a);
printf("----------\n");
printf("%i\n",a);
printf("%i%c%i\n",a,sp,a*a);
printf("%i%c%i%c%i",a,sp,a*a,sp,
a*a*a);
return 0;
}
```

7.

```
/* 2.7 */
#include <stdio.h>
main()
{
float op1,op2;
printf("Calculator\n");
printf("Enter operand1:");
scanf("%f",&op1);
printf("Enter operand2:");
```

8.

```
/* 2.8 */
#include <stdio.h>
main()
{
const
k=3;
float a,b,c;
printf("Input variables:\n");
printf("a=");
```

	scanf("%f",&op2);
	printf("product=");
	printf("%5.2f\n",op1*op2);
	printf("sum=");
	printf("%5.2f\n",op1+op2);
	printf("difference=");
	printf("%5.2f\n",op1–op2);
	return 0;
	}
			scanf("%f",&a);
			printf("b=");
			scanf("%f",&b);
			printf("c=");
			scanf("%f",&c);
			printf("Arithmetical mean a,b,c=");
			printf("%5.2f",(a+b+c)/k);
			return 0;
			}

9.	/* 2.9 */
	#include <stdio.h>
	main()
	{
	float c1,c2;
	printf("Calculation of areal\n");
	printf("Input cathetus_1:");
	scanf("%f",&c1);
	printf("Input cathetus_2:");
	scanf("%f",&c2);
	/* Result */
	printf("S=%5.2f",0.5*c1*c2);
	return 0;
	}
		10.	/* 2.10 */
			#include <stdio.h>
			main()
			{
			const float pi=3.14;
			float r;
			printf("Calculation of volume\n");
			printf("Input radius:\n");
			printf("r=");
			scanf("%f",&r);
			printf("Volume:\n");
			printf("V=%5.2f",4*pi*r*r/3);
			return 0;
			}

11.	/* 2.11 */
	#include <stdio.h>
	main()
	{
	float side1,side2;
	printf("Calculation of perimeter");
	printf("\nInput side_1:");
	scanf("%f",&side1);
	printf("Input side_2:");
	scanf("%f",&side2);
	printf("Perimeter:\n");
	printf("P=%5.2f",2*(side1+side2));
	return 0;
	}
		12.	/* 2.12 */
			#include <stdio.h>
			main()
			{
			const float
			tf=32,
			k=1.8;
			float tc;
			printf("Celsius temperature:");
			scanf("%f",&tc);
			printf("Fahrenheit temperature:");
			printf("%5.2f",tf+k*tc);
			return 0;
			}

13.	/* 2.13 */
	#include <stdio.h>
	main()
	{
	const float pi=3.14;
	float r;
		14.	/* 2.14 */
			#include <stdio.h>
			main()
			{
			const float pi=3.14;
			float r;

	printf("Input radius:\n");
	printf("R=");
	scanf("%f",&r);
	printf("Result: ");
	printf("L=2*Pi*R");
	printf("=%5.2f",2*pi*r);
	return 0;
	}
			printf("Input radius:\n");
			printf("R=");
			scanf("%f",&r);
			printf("Result: ");
			printf("S=Pi*R^2");
			printf("=%5.2lf",r*r*pi);
			return 0;
			}

15.	/* 2.15 */
	#include <stdio.h>
	main()
	{
	float a,b,c;
	printf("Calculation of");
	printf("algebraical expression:\n");
	printf("2*(a+b)/(a*b*c)\n");
	printf("Input variables:\n");
	printf("a="); scanf("%f",&a);
	printf("b="); scanf("%f",&b);
	printf("c="); scanf("%f",&c);
	printf("Result=");
	printf("%5.2f",2.0*(a+b)/(a*b*c));
	return 0;
	}
		16.	/* 2.16 */
			#include <stdio.h>
			main()
			{
			float a,t,spd;
			printf("Calculation of way\n");
			printf("time=");
			scanf("%f",&t);
			printf("acceleration=");
			scanf("%f",&a);
			printf("speed=");
			scanf("%f",&spd);
			printf("Way:");
			printf("s=%5.2f",spd*t+0.5*a*t*t);
			return 0;
			}

17.	/* 2.17 */
	#include <stdio.h>
	main()
	{
	const
	float k=1.609344;
	long int miles;
	printf("Hello!\n");
	printf("Translate miles in km\n");
	printf("How many miles? :");
	scanf("%li",&miles);
	printf("Kilometers=");
	printf("%5.2f",miles*k);
	return 0;
	}
		18.	/* 2.18 */
			#include <stdio.h>
			main()
			{
			float sd1,sd2,sd3;
			printf("Input side1:");
			scanf("%f",&sd1);
			printf("Input side2:");
			scanf("%f",&sd2);
			printf("Input side3:");
			scanf("%f",&sd3);
			printf("Perimeter:\n");
			printf("P=%5.2f",sd1+sd2+sd3);
			return 0;
			}

19.	/* 2.19 */
	#include <stdio.h>
	main()
	{
		20.	/* 2.20 */
			#include <stdio.h>
			main()
			{

```
float t1,t2,s1,s2;
printf("Calculation of way\n");
printf("Input time:\n");
printf("time1=");scanf("%f",&t1);
printf("time2=");scanf("%f",&t2);
printf("Input speed:\n");
printf("speed1=");scanf("%f",&s1);
printf("speed2=");scanf("%f",&s2);
printf("way:\n");
printf("l=%5.2f",s1*t1+s2*t2);
return 0;
}
```

```
float w1,w2,s1,s2;
printf("Calculation of time\n");
printf("Input way:\n");
printf("way1=");scanf("%f",&w1);
printf("way2=");scanf("%f",&w2);
printf("Input speed:\n");
printf("speed1=");scanf("%f",&s1);
printf("speed2=");scanf("%f",&s2);
printf("time:\n");
printf("l=%5.2f",w1/s1+w2/s2);
return 0;
}
```

21.

```
/* 2.21 */
#include <stdio.h>
main()
{
float t1,t2,c,m;
printf("Calculation Q\n");
printf("Input temperatures:\n");
printf("t1=");scanf("%f",&t1);
printf("t2=");scanf("%f",&t2);
printf("c=");scanf("%f",&c);
printf("m=");scanf("%f",&m);
printf("Q=cm(t2-t1)\n");
printf("Q=%5.2f",c*m*(t2-t1));
return 0;
}
```

22.

```
/* 2.22 */
#include <stdio.h>
main()
{
float a,b,c,d;
printf("Calculation of:\n ");
printf("x=a*b+c*d\n");
printf("a=");scanf("%f",&a);
printf("b=");scanf("%f",&b);
printf("c=");scanf("%f",&c);
printf("d=");scanf("%f",&d);
printf("Result:\n");
printf("x=%5.2f",a*b+c*d);
return 0;
}
```

23.

```
/* 2.23 */
#include <stdio.h>
main()
{
const sp=32;
unsigned int a;
printf("Enter digit:");
scanf("%i",&a);
printf("*************\n");
printf("%d%d%d%d%d\n",a,a,a,a,a);
printf("%c%d%c%d%c\n",sp,a,sp,a,sp);
printf("%c%c%d%c%c\n",sp,sp,a,sp,sp);
printf("%c%d%c%d%c\n",sp,a,sp,a,sp);
printf("%d%d%d%d%d",a,a,a,a,a);
return 0;
}
```

24.

```
/* 2.24 */
#include <stdio.h>
main()
{
const k=1, s=32;
printf("%c%c%c%c%d%c%c%c%c\n",s,s,s,s,k,s,s,s,s);
printf("%c%c%c%d%c%d%c%c%c\n",s,s,s,k,s,k,s,s,s);
printf("%c%c%d%c%d%c%d%c%c\n",s,s,k,s,k+k,s,k,s,s);
printf("%c%d%c%d%c%d%c%d%c\n",s,k,s,3*k,s,3*k,s,k,s);
printf("%d%c%d%c%d%c%d%c%d",k,s,4*k,s,6*k,s,4*k,s,k);
return 0;
}
```

25.

```
/* 2.25 */
#include <stdio.h>
main()
{
const c=32;
unsigned int k;
printf("Enter whole number k:");
scanf("%i",&k);
printf("----------------------\n");
printf("%c%c%c%c%i%c%c%c%c\n",c,c,c,c,k,c,c,c,c);
printf("%c%c%c%i%c%i%c%c%c\n",c,c,c,k,c,k,c,c,c);
printf("%c%c%i%c%i%c%i%c%c\n",c,c,k,c,k+k,c,k,c,c);
printf("%c%i%c%i%c%i%c%i%c\n",c,k,c,3*k,c,3*k,c,k,c);
printf("%i%c%i%c%i%c%i%c%i\n",k,c,4*k,c,6*k,c,4*k,c,k);
return 0;
}
```

<u>Контрольні запитання:</u>

1. Як виглядає найкоротша програма C/C++?
2. Яка головна функція програми C/C++ та її призначення?
3. Як завершується виконання програми C/C++?
4. Для чого застосовуються заголовні файли (директиви включення)?
5. Як оформлюються коментарі програм C та C++?
6. Які призначення та правила конструювання ідентифікаторів?
7. Що таке ключові слова мови програмування?
8. Як здійснюється у програмі оголошення змінних та констант?
9. Чим відрізняються змінні від констант програми?
10. Що означає локальність та глобальність змінних?
11. Яким чином здійснюється введення та виведення у C-програмі? Як працюють ідентифікатори потоків введення та виведення C++?

5. Практична робота **№3.**
Виконання простих обчислень

***Мета роботи*:** вивчення числових типів даних, основних операторів та порядку дій при їх обчисленні; набуття навичок використання стандартних математичних функцій C/C++.

***Теоретичні основи*.**

Числові типи C/C++ поділяються на цілочислові та дійсні.

Точні діапазони значень і розміри пам'яті для різних числових типів даних визначаються певним компілятором та операційною системою ЕОМ.

Діапазони значень цілих чисел Borland C++:

Тип даних	Розмір, байт	Діапазон значень
signed char	1	–128…127
unsigned char	1	0…255
signed short int	2	–32768…32767
unsigned short int	2	0…65535
signed int	2	–32768…32767
unsigned int	2	0…65535
signed long int	4	–2147483648…2147483647
unsigned long int	4	0…4294967295

Слово signed не є обов'язковим. Також не є обов'язковим слово int в типах short int, long int і unsigned int.

Мова С використовує цілі типи також і для зображення логічних тверджень, тобто нуль означає хибність (false), а ненульове значення – істину (true).

У C/C++ припустимо використання суфіксів **U** (або **u** – unsigned), **L** (або **l** – long) і **H** (або **h** – short) для змінення типу цілочислового літералу, наприклад, 1234L має тип long, 1234U – unsigned int, а 1234UL – unsigned long.

Застосування префіксів **0** (нуль) та **0X** (або 0x) дозволяє записувати цілі числа у вісімковому та шістнадцятковому форматах відповідно, наприклад: 0724, 0x9FAC тощо.

Числа з плаваючою комою (дійсні числа) у C/C++ записуються у формі з десятковою крапкою або за допомогою «наукової нотації», наприклад: 375.5, або 3.755E2, або 37.55e+01, або 0.3755e3, або 37550E–2 тощо.

Діапазони значень дійсних чисел Borland C++:

Тип даних	Розмір, байт	Діапазон значень
float	4	$\pm 3.4 \cdot 10^{-38} \ldots \pm 3.4 \cdot 10^{38}$
double	8	$\pm 1.7 \cdot 10^{-308} \ldots \pm 1.7 \cdot 10^{308}$
long double	10	$\pm 3.4 \cdot 10^{-4932} \ldots \pm 1.1 \cdot 10^{4932}$

Використання суфіксів **F** (або **f** – float) і **L** (або **l** – long double) дозволяє змінювати тип дійсного літералу, наприклад, число 8.75f має тип float, а 8.75L – long double.

У програмах C/C++ використовуються наступні оператори: арифметичні, відношень, логічні, оператор заперечення, інкременту і декременту, порозрядні, присвоєння.

Арифметичні оператори C/C++:

Оператор	Дія	Приклад
+	додавання	c=a+b;
–	віднімання	c=a–b;
*	множення	c=a*b;
/	ділення	c=a/b;
%	остача від ділення	c=a%b;

Оператори відношень C/C++:

Оператор	Опис	Приклад
<	менше	(a<b)
<=	менше або дорівнює	(a<=b)
>	більше	(a>b)
>=	більше або дорівнює	(a>=b)
==	дорівнює	(a==b)
!=	не дорівнює	(a!=b)

Логічні оператори **&&** і || об’єднують вирази відношень у відповідності до правил для логічного «І» (AND) і «АБО» (OR) відповідно та використовуються у складених логічних виразах, наприклад:

```
((a<b)&&(b<c))
```

або

```
((a<b)||(c<d))
```

Унарний оператор заперечення **!** використовується для інвертування результату будь-якого логічного виразу. Наприклад, наступні вирази є еквівалентними:

```
(!(a<b)) і (a>=b)
```

та

```
(!(a==b)) і (a!=b)
```

Оператор інкременту (++) додає до операнду одиницю, а оператор декременту (--) віднімає її, наприклад:

```
k++; /* k=k+1 */
```

```
k--; /* k=k–1 */
x=k++; /* x=k; k=k+1; */
x=--k; /* k=k–1; x=k; */
```

Порозрядні оператори C/C++:

Оператор	Опис	Приклад
&	порозрядне «І»	c=a&b;
|	порозрядне «АБО»	c=a|b;
^	порозрядне виключне «АБО»	c=a^b;
<<	зсув бітів уліво	c=a<<b;
>>	зсув бітів управо	c=a>>b;
~	порозрядне заперечення	c=~a;

Традиційний оператор присвоєння (=) застосовується для присвоєння результату будь-якого виразу змінній, а також для одночасної ініціалізації декількох змінних, наприклад:

```
z=1;
x=y+n;
c=a=b=d+50;
```

У C/C++ застосовуються також скорочені оператори присвоєння:

+=, -=, *=, /=, %=, &=, |=, ^=, <<=, >>=,

наприклад:

```
count+=x; /* count=count+x; */
count*=x; /* count=count*x; */
```

У складних виразах операції виконуються у відповідності до пріоритету та асоціативності (порядку обчислень):

Рівень пріоритету	Оператори	Асоціативність
1.	**()**, **.**, **[]**, **–>**	зліва направо
2.	*, **&**, **!**, ~, ++, **--**, +, –, **(тип)**, **sizeof**	справа наліво
3.	*, /, **%**	зліва направо
4.	+, –	зліва направо
5.	<<, >>	зліва направо
6.	<, <=, >, >=	зліва направо
7.	==, **!=**	зліва направо
8.	**&**	зліва направо
9.	^	зліва направо

10.	|	зліва направо
11.	**&&**	зліва направо
12.	||	зліва направо
13.	**?:**	справа наліво
14.	=, *****=, /=, –=, +=, **%**=, <<=, >>=, **&**=, ^=, |=	справа наліво
15.	**,**	зліва направо

Borland C++ містить достатньо обширну бібліотеку стандартних функцій. Найбільш використовувані наступні математичні функції:

Виклик функції	Призначення	#include
abs(*x*), **labs(*x*)**	– абсолютне значення цілого *x*	math.h
acos(*x*), **acosl(*x*)**	– арккосинус *x*, радіан	math.h
asin(*x*), **asinl(*x*)**	– арксинус *x*, радіан	math.h
atan(*x*), **atanl(*x*)**	– арктангенс *x*, радіан	math.h
atan2(y,*x*),**atan2l(y,*x*)**	– арктангенс y/*x*, радіан	math.h
ceil(*x*), **ceill(*x*)**	– округлення *x* до більшого цілого	math.h
cos(*x*), **cosl(*x*)**	– косинус *x*	math.h
div(*x*,*y*), **ldiv(*x*,*y*)**	– частка і остача від ділення цілих *x* і *y*	stdlib.h
exp(*x*), **expl(*x*)**	– експонента e^x	math.h
fabs(*x*), **fabsl(*x*)**	– абсолютне значення дійсного *x*	math.h
floor(*x*), **floorl(*x*)**	– округлення *x* до меншого цілого	math.h
fmod(*x*,*y*), **fmodl(*x*,*y*)**	– остача від ділення дійсних *x* і *y*	math.h
log(*x*)	– натуральний логарифм *x*	math.h
log10(*x*)	– десятковий логарифм *x*	math.h
modf(*x*,&*y*), **modfl(*x*,&*y*)**	– розбивка дійсного *x* на цілу і дробову частини	math.h
poly(…)	– значення поліному	math.h
pow(*x*,*y*),**powl(*x*,*y*)**	– значення *x* у степені *y*	math.h
pow10(*x*),**pow10l(*x*)**	– значення 10 у степені *x*	math.h
rand()	– генероване випадкове ціле від 0	stdlib.h
random(*n*)	– генероване випадкове ціле від 0 до *n*-1	stdlib.h
randomize()	– ініціалізація генератора випадкових чисел	stdlib.h, time.h
sin(*x*), sinl(*x*)	– синус *x*	math.h
sinh(*x*), **sinhl(*x*)**	– гіперболічний синус *x*	math.h
sqrt(*x*), sqrtl(*x*)	– квадратний корінь *x*	math.h
srand(*n*)	– ініціалізація генератора передбачуваних випадкових чисел	stdlib.h
tan(*x*), tanl(*x*)	– тангенс *x*	math.h
tanh(*x*), tanhl(*x*)	– гіперболічний тангенс *x*	math.h

Деякі функції, із-за їх відсутності серед бібліотечних, необхідно конструювати самостійно, за допомогою інших наявних функцій. Так,

наприклад, логарифм за основою $\log_a x$ можна замінити виразом C/C++: $\log(x)/\log(a)$, а котангенс $\mathrm{ctg}(x)$ – виразом: $1/\tan(x)$ тощо.

Також у заголовному файлі math.h визначено багато корисних символічних констант математичного призначення, наприклад:

M_E=2.71828182845904523536 /* e */
M_LN10=2.30258509299404568402 /* $\ln(10)$ */
M_PI=3.14159265358979323846 /* π */
M_SQRT2=1.41421356237309504880 /* $\sqrt{2}$ */

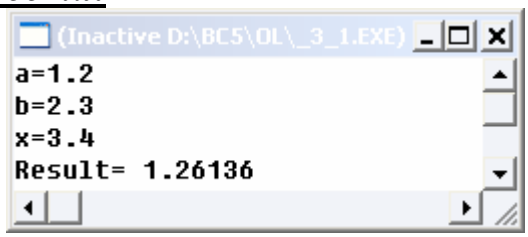
та інші, які використовуються у математичних виразах замість їх літеральних значень.

Приклад завдання 3.1:
Дано дійсні числа a, b та x.
Обчислити: $w = \lg|a| + \mathrm{arctg}(x^2)\dfrac{\pi a}{\sqrt{|a+x|}}\left(\sqrt[3]{b} - \cos\dfrac{a}{b}\right)$.

Розв'язок:

```
/* 3.1 */
#include <stdio.h>
#include <math.h>
main()
{
  float a,b,x,w;
  printf("a=");
  scanf("%f",&a);
  printf("b=");
  scanf("%f",&b);
  printf("x=");
  scanf("%f",&x);
  w=log10(fabs(a))+atan(x*x)*M_PI*a*(pow(b,1.0/3)–
  cos(a/b))/sqrt(fabs(a+x));
  printf("Result=%8.5f",w);
  return 0;
}
```

Результати роботи:



Завдання:

1. Дано дійсні числа y та генероване випадкове дійсне число x (x=0…1). Обчислити $a=(1+y^5)\dfrac{x+y/(\sqrt{|x|}+4)}{e^{-x-2}+1/(x^2+4)}$. Результат округлити.

2. Дано дійсне число y, натуральне число n та генероване випадкове дійсне число x (x=0…n). Обчислити $a=\dfrac{1+\sin^2(x+y)}{2+\left|x-2x/(1+x^2y^3)\right|}+\sqrt[3]{x}$. Вивести дробову частину результату.

3. Дано дійсні числа y, z, натуральне число n та генероване випадкове натуральне число x (x=0…n). Обчислити $a=\ln\left|\left(y-\sqrt{|x|}\right)\cdot\left(x-\dfrac{y}{z^5+x^2/4}\right)\right|$. Вивести цілу частину результату.

4. Дано дійсні числа x та y. Знайти цілу частину $a=x^3+\dfrac{\sin y^x+x-y/2}{2x-y+1}$. Вивести значення молодшого біта цілої частини числа a.

5. Дано дійсні числа x та y. Округлити $a=x+\dfrac{\sqrt[5]{x}}{1+\sin^2(x+y)}$. Вивести значення старшого біта округленого числа a.

6. Дано дійсні числа x, y і z Обчислити $a=\dfrac{\sqrt{|x-1|}-\sqrt[3]{|z|}}{1+x^2/2+y^2/4}+z^4$ і $b=\dfrac{3+e^{y-1}}{1+x^5|y-\mathrm{tg}z|}+x^2$ та знайти залишок від цілочислового ділення округлених a і b.

7. Дано дійсні числа x, y та натуральне число n. Обчислити $a=\dfrac{2\cos(x-\pi/6)}{1/2+\sin^2 y}+10xy$. Виконати цілочислове ділення округленого a на n.

8. Дано ціле число y. Генерувати випадкове ціле число n (n=0…x), де $x=2y^4-3y^3+4y^2-5y+6$.

9. Дано дійсні числа x, y, натуральне число n та генероване випадкове ціле число k (k=0…n). Обчислити $a=\dfrac{2\sin(x-\pi/6)}{1/3+\cos^5 y}+xy^2$. Виконати цілочислове ділення округленого a на k+1.

10. Дано дійсні числа x і y. Обчислити $a=x^5+x^4-3x^3+4x^2-5x+6$ і $b=9y^3-7y^2+5y-3$ та знайти результат цілочислового ділення округлених a і b.

11. Дано натуральне число n та генеровані випадкові цілі числа x і y

(x=0…n; y=0…n). Обчислити $a=\sin\left|\left(y-\sqrt{|x|}\right)\cdot\left(x-\dfrac{y}{y^2+x^7/4}\right)\right|$.

12. Дано дійсне число x (x=0…1) та генероване випадкове дійсне число y (y=0…1). Обчислити $a=\dfrac{2y^3-y^2+3y-1}{\left|x^2-2x+1\right|+\sin y}$. Результат округлити.

13. Дано дійсне число y, натуральне число n та генероване випадкове дійсне число x (x=0…n). Обчислити $a=\dfrac{\cos y^x+x-y/3}{2x^2-y+1}$. Вивести дробову частину результату.

14. Дано дійсні числа y, z, натуральне число n та генероване випадкове натуральне число x (x=0…n). Обчислити $a=\dfrac{5x^5\ln y+3}{4y^2e^{\sin z}-x/(y+1)}$. Вивести цілу частину результату.

15. Дано дійсне число x (x<10) та генероване випадкове дійсне число y (y<10). Знайти цілу частину $a=x^5+\dfrac{\mathrm{tg}y^x+x-y/2}{2x^2-y^3+1}$. Вивести значення молодшого біта цілої частини числа a.

16. Дано дійсні числа x (x<10) та y. Округлити $a=x^7+\dfrac{\mathrm{arctg}\sqrt[5]{x}}{\ln y+\sin^2(x+y)}$. Вивести значення старшого біта округленого числа a.

17. Дано дійсні числа x, y і z Обчислити $a=5x\dfrac{\sqrt{|x-1|}-\sqrt[3]{|z|}}{1+x^2/2+y^2/4}$ і $b=3y\dfrac{2+e^{2y-3}}{1+x^5\left|y-\mathrm{tg}z\right|}$ та знайти залишок від цілочислового ділення округлених a і b.

18. Дано ціле число y. Генерувати випадкове ціле число n (n=0…x), де $x=y^4+3y^3+9y^2+y+7$.

19. Дано натуральне число n та генеровані випадкові цілі числа x (x=0…n) і y (y=0…n). Обчислити $a=\cos\left|\left(y^3+\sqrt{|x|}\right)\cdot\left(x-\dfrac{y}{\sin y^2+x^5/5}\right)\right|+\log\left(x^{0,25}y\right)$.

20. Дано дійсні числа x і y. Обчислити значення $a=\left|\dfrac{\sin^3\left|3x^3+2y^2-6\right|}{\sqrt{(3x^3+2y^2-6)^2+\pi}}\right|$, а також цілу та дробову його частини.

21. Дано дійсні числа x і y. Знайти z – результат цілочислового ділення цілих частин x і y та використати його для обчислення поліному $z^5+z^4-5{,}4z^3+6{,}5z^2-6{,}7z+8{,}7$.

22. Дано дійсні числа x і y. Знайти z – залишок від цілочислового ділення цілих частин x і y та використати його для обчислення поліному $z^4 + 1{,}09z^3 + 7{,}03z^2 + 2{,}01z - 5{,}05$.

23. Дано два натуральних числа x і y. Генерувати випадкове ціле число z (z=0…xy) та використати його для обчислення $a = e^{z^2} \dfrac{\sqrt[3]{x + 10y}}{z - xy}$.

24. Дано натуральне число x та дійсне число y. Обчислити $a = x + \left| \dfrac{\ln^3 \left| 2x^3 + 3y^2 - 7 \right|}{\sqrt[3]{xy}\sqrt{(3x^3 + 2y^2 - 6)^2 + \pi}} \right|$. Результат округлити і помножити на генероване випадкове ціле число n (n=0…x).

25. Дано натуральне число z. Генерувати випадкове ціле число (0…y), де $y = z^4 + 10z^3 + 7z^2 + 2z - 5$. Результат поділити на z та округлити.

Контрольні запитання:

1. Які основні числові типи даних C/C++?
2. Які основні характеристики цілочислових типів C/C++?
3. Які основні характеристики дійсних типів C/C++?
4. Яким чином можна змінювати тип числового літерального значення?
5. Які основні оператори C/C++ та як вони використовуються?
6. Який порядок виконання операцій у складних виразах?
7. Які основні функції C/C++ математичного призначення.
8. Коли і яким чином застосовуються символічні константи?

6. Практична робота **№4.**
Реалізація алгоритмів розгалуження

***Мета роботи*:** набуття навичок роботи з алгоритмічними конструкціями розгалуження у середовищі Borland C++, у тому числі з використанням складених операторів.

***Теоретичні основи*.**

Алгоритмічна конструкція, що дозволяє обирати ту чи іншу послідовність дій залежно від певних умов, називається розгалуженням. Існують такі різновиди конструкції розгалуження: альтернатива і поліваріантний вибір.

В алгоритмічній мові C/C++ алгоритми розгалуження реалізуються умовними операторами, що функціонують за наступними схемами (рис. 6.1).

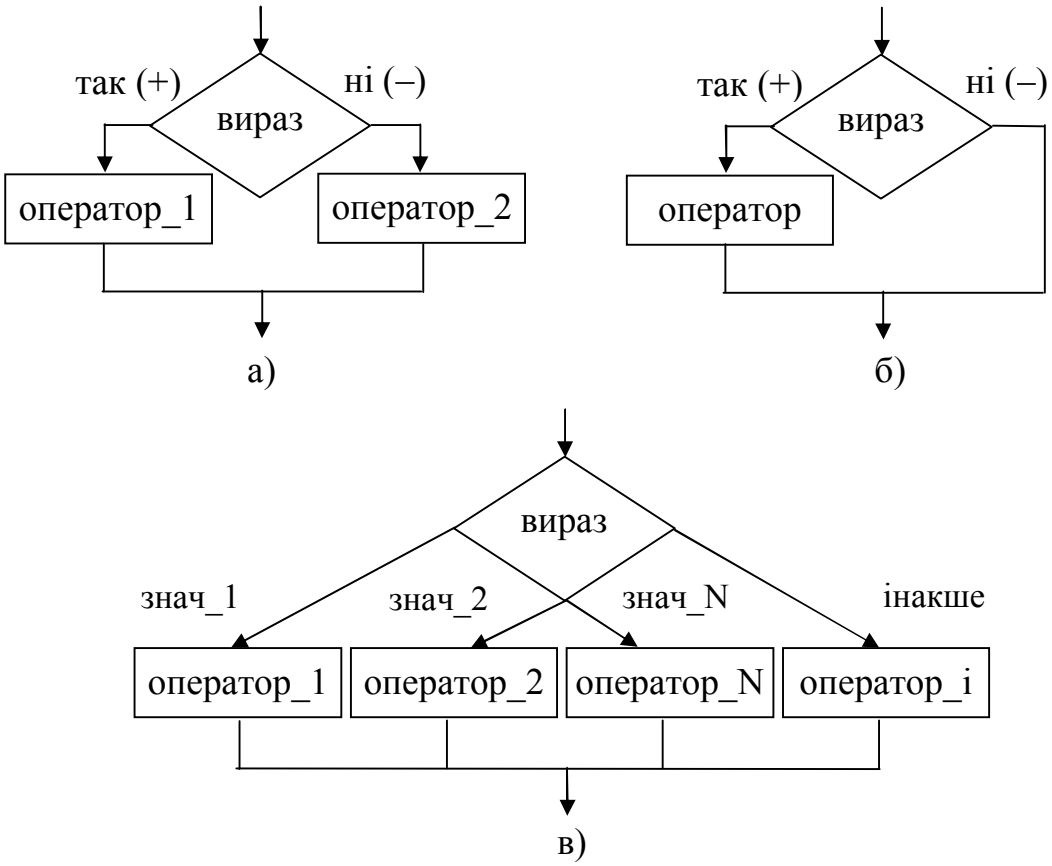


Рисунок 6.1 – Схеми алгоритмів розгалуження

Альтернативне розгалуження (рис. 6.1, а) реалізується умовним оператором за таким синтаксисом:

if (вираз) оператор_1; else оператор_2;

де if (якщо) та else (інакше) – зарезервовані слова; оператор_1 і оператор_2

– довільні оператори, що виконуються в залежності від результату обчислення виразу. Якщо цей вираз є істинним (має ненульове значення), то виконується оператор_1 і керування передається наступному за умовним оператору, а оператор_2 пропускається; якщо вираз є хибним (має нульове значення), то оператор_1 пропускається, проте виконується оператор_2.

В той же час і оператор_1 і оператор_2 можуть бути як простими так і складеними. Складені оператори обмежуються фігурними дужками, наприклад:

```
if(a>b) b++;
else
{
  printf(″%d\n″,a);
  a++;
}
```

Скорочена форма умовного оператора (рис. 6.1, б) реалізує безальтернативне розгалуження (корекцію) за таким синтаксисом:

if (вираз) оператор;

тобто якщо вираз є істинним (має ненульове значення), то оператор виконується, якщо хибним (має нульове значення) – не виконується, і на цьому дія умовного оператора припиняється, наприклад:

```
if(x<=y) x+=y;
```

Логічні оператори **&&** і || об’єднують окремі вирази умовних операторів у складені вирази відношень у відповідності до правил для логічних «І» (AND) та «АБО» (OR) відповідно, наприклад:

```
if((x>=a) && (x<=b)) …; /* якщо a<=x<=b … */
if((x<=a) || (x>=b)) …; /* якщо x<=a або x>=b … */
```

У C/C++ існує також скорочений оператор if–else, який ще має назву умовний вираз або операція умови:

вираз_1 ? вираз_2 : вираз_3;

Програма обчислює вираз_1, і якщо він є істинним (має ненульове значення), то результат усього виразу дорівнює виразу_2, у противному разі результат дорівнює виразу_3.

Умовний вираз цілком відповідає наступному оператору альтернати-

вного розгалуження:

if (вираз_1) вираз_2; else вираз_3;

Отже наступні оператори, що знаходять найбільше з двох чисел, є цілком еквівалентними:

if(x>y) max=x; else max=y;

та

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

За допомогою умовного виразу можна, наприклад, наступним чином розв'язати задачу заміни нулем найменшого серед двох цілих чисел x і y:

(x>y)?x:y=0;

Гілки деякого розгалуження можуть містити інші розгалуження, наприклад:

```
if (вираз_1) оператор_1;
    else if вираз_2 оператор_2;
        else оператор_3;
```

Поліваріантний вибір (рис. 6.1, в), що дозволяє виконувати одну з декількох алгоритмічних гілок залежно від значення деякого виразу, у C/C++ реалізується оператором вибору за таким синтаксисом:

```
switch (вираз)
{
case значення_1 : оператор_1; break;
case значення_2 : оператор_2; break;
…
case значення_3, …, значення_N : оператор_3; break;

 default : оператор_замовчання;
}
```

Оператор вибору виконується за таким алгоритмом. Спочатку обчислюється значення виразу switch. Потім це значення порівнюється по черзі з запропонованими значеннями селекторів case (значення_1, значення_2, …, значення_N тощо). Щойно результат порівняння дасть значення «істина», одразу ж виконується відповідний оператор або набір операторів, аж до першого наявного оператора break. Якщо значення виразу switch

не збігається з жодним значенням селекторів case, то виконується необов'язкова гілка default, або, якщо її нема, виконання оператора вибору завершується. Оператор break у кожному блоці case негайно призводить до завершення оператора switch.

Приклад застосування оператора switch:

```
switch(i)
{
case 1:printf(″i=1\n″); break;
case 2:printf(″i=2\n″); break;
case 3,4,5:printf(″i=3–5\n″); break;
default:printf(″i=All\n″);
}
```

Для безумовного завершення поточної функції (у тому числі функції main()) використовуються оператори **exit()** і **abort()**, прототипи яких знаходяться у заголовному файлі stdlib.h, наприклад:

```
exit(0);
abort();
```

Приклад завдання 4.1:

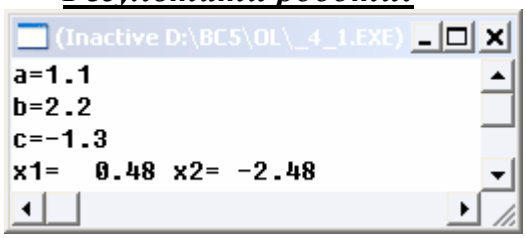
Знайти корені квадратного рівняння $ax^2+bx+c=0$, коефіцієнти якого є дійсними числами, що їх вводить користувач.

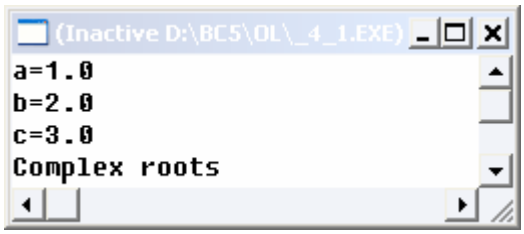
Розв'язок:

```
/* 4.1 */
#include <stdio.h>
#include <math.h>
main()
{
  float x1,x2,x,a,b,c,d;
  printf("a="); scanf("%f",&a);
  printf("b="); scanf("%f",&b);
  printf("c="); scanf("%f",&c);
  if((d=b*b–4*a*c)>=0)
    if(d>0)
    {
      x1=(–b+sqrt(d))/(2*a);
      x2=(–b–sqrt(d))/(2*a);
      printf("x1=%6.2f x2=%6.2f",x1,x2);
    }
    else
```

```
    {
      x=–b/(2*a);
      printf("x=%6.2f",x);
    }
  else printf("Complex roots");
  return 0;
}
```

Результати роботи:





Завдання:

1. Дано чотирьохзначне натуральне число. З'ясувати яка з двох його частин більша.
2. Дано чотирьохзначне натуральне число. З'ясувати яка сума більша, перших двох його цифр чи останніх.
3. Дано трьохзначне натуральне число. З'ясувати яка з його цифр менша, перша чи остання.
4. Дано натуральне число. З'ясувати, чи рівні між собою остання та передостання цифри.
5. Дано трьохзначне натуральне число. З'ясувати, чи всі його цифри різні.
6. Дано п'ятизначне натуральне число. Якщо остання його цифра більша за першу, то переставити їх місцями.
7. Дано два дійсних числа. З'ясувати, чи одного знаку ці числа.
8. Дано два дійсних числа. Розташувати їх таким чином, щоб на першому місці стояло те з них, у котрого дробова частина менша. Додаткову змінну не використовувати.
9. Дано три дійсних числа. Знайти суму двох найбільших серед них.
10. Дано три дійсних числа. Знайти серед них таке, що за значенням знаходиться між найбільшим та найменшим.
11. Дано три дійсних числа. З'ясувати які з них належать до діапазону [0,1].
12. Дано три натуральних числа. Розташувати їх таким чином, аби на першому місці стояло найменше з них, а на останньому – найбільше.
13. Дано три дійсних числа. З'ясувати що більше, квадрат мінімального чи максимальне з них.
14. Дано два натуральних числа. Розташувати їх таким чином, аби на першому місці стояло найбільше з них.
15. На площині задано три точки з координатами (x_1,y_1), (x_2,y_2), та (x_3,y_3). Визначити, яка з них ближча до початку координат.

16. Дано три натуральних числа. З'ясувати, чи є серед них хоча б два однакових.

17. Дано дійсні числа *a*, *b*, *c*. Подвоїти ці числа, якщо $a \ge b \ge c$ чи залишити їх без змін, якщо це не так.

18. Дано дійсне число. Якщо його дробова частина менша за 0,5, то округлити це число до найближчого більшого, а якщо ні, то знайти його цілу частину.

19. Дано три дійсних числа, що визначають довжину сторін деякого трикутника. З'ясувати, чи існує такий трикутник.

20. Дано натуральне число, що визначає деякий рік. З'ясувати, чи є він високосним. Високосний рік, ділиться на 4, за виключенням тих років, котрі діляться на 100 та не діляться на 400 (наприклад, 300, 1300 і 1900 не є високосними, а 1200 і 2000 – високосні роки).

21. Дано натуральне число від 1 до 12, що визначає деякий місяць року. З'ясувати скільки днів у цьому місяці.

22. Дано дійсне число *x*. Обчислити $z = \begin{cases} \sin(x), & \text{якщо} \quad x \le 0; \\ x^2 - x, & \text{якщо} \quad 0 < x \le 1; \\ x^2 - \sin(\pi x^2) - 1 & \text{інакше.} \end{cases}$

23. Дано дійсне число *x*. Обчислити $z = \begin{cases} 0.5x, & \text{якщо} \quad x < -1; \\ \sqrt{1 - x^2}, & \text{якщо} \quad -1 \le x \le 1; \\ 0.5x - 1 & \text{інакше.} \end{cases}$

24. Дано дійсне число *x*. Обчислити $z = \begin{cases} \sin(x), & \text{якщо} \quad x < 0; \\ \dfrac{e^x - e^{-x}}{e^x + e^{-x}}, & \text{якщо} \quad 0 \le x \le 3; \\ \ln(x) & \text{інакше.} \end{cases}$

25. Дано дійсне число *x*. Обчислити $z = \begin{cases} -x/5, & \text{якщо} \quad x < 0; \\ 2x^2 e^{-2x}, & \text{якщо} \quad 0 \le x < 1; \\ x^{0.2}/4 & \text{інакше.} \end{cases}$

<u>Контрольні запитання:</u>

1. Яка алгоритмічна конструкція називається розгалуженням?
2. Які існують різновиди конструкції розгалуження у C/C++ і як вони працюють?
3. Який синтаксис умовного оператора?
4. Який синтаксис та призначення умовного виразу або операції умови?
5. Що таке складений оператор?
6. Як поєднуються в умовних операторах прості вирази в складений?
7. Яке призначення та як функціонує оператор поліваріантного вибору?
8. Як підвищити ефективність алгоритмів розгалуження?
9. Як забезпечити безумовне завершення головної функції програми?

7. Практична робота **№5.**
Реалізація простих циклічних алгоритмів

***Мета роботи*:** вивчення основних прийомів роботи з алгоритмічною конструкцією повторення (циклами) засобами Borland C++; набуття навичок розв'язку задач, що потребують застосування ітераційних процесів.

***Теоретичні основи*.**

Під час розробки програми часто постає потреба виконати один і той самий оператор декілька разів. Для цього застосовують оператори циклів, що реалізують алгоритмічну конструкцію повторення. Кожне повторне виконання операторів, що являють собою тіло циклу, називається ітерацією.

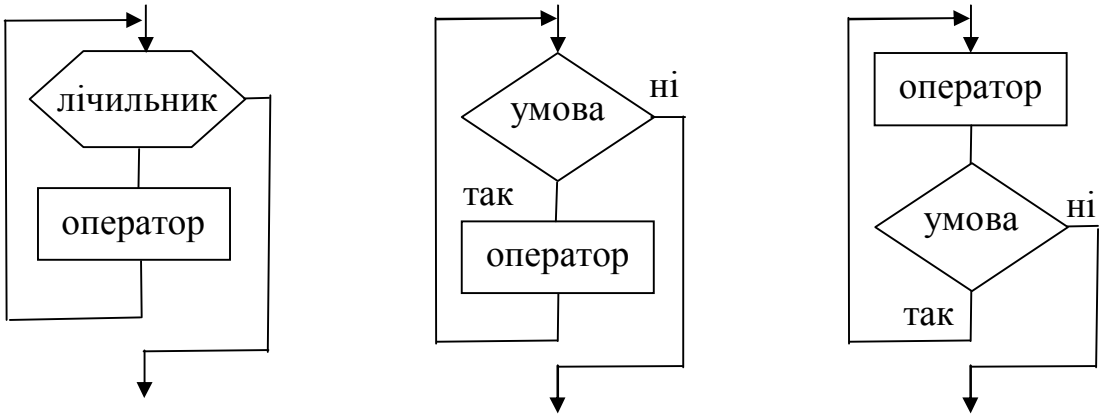
В програмах C/C++ застосовуються цикли трьох видів (рис. 7.1):

– із параметром-лічильником (**for**);

– із передумовою (**while**);

– із постумовою або післяумовою (**do–while**).

Вибір того чи іншого оператора циклу визначається застосованим алгоритмом розв'язку певної задачі.



а) б) в)

Рисунок 7.1 – Схеми операторів циклу

Основна форма циклу for має наступний синтаксис (рис. 7.1, а):

for (вираз_1; вираз_2; вираз_3) оператор;

де for – зарезервоване слово; вираз_1 (ініціалізуючий) виконується лише один раз, перед початком роботи циклу; вираз_2 зазвичай є виразом відношення, що визначає умову виконання оператору; вираз_3, як правило, застосовується для змінення параметра циклу (збільшення або зменшення).

Наприклад, наступні рядки програми виводять на екран спочатку числа від 0 до 9, а потім від 9 до 0 за допомогою оператора циклу for (кожне з

нового рядка):

```
for(i=0;i<=9;i++) printf(″%i\n″,i);
for(i=9;i>=0;i--) printf(″%i\n″,i);
```

Кожен із виразів заголовку оператора for може бути відсутнім, проте символ «;» вказується обов'язково. Наприклад, якщо вирази оператора for зовсім поминути, то отримаємо безкінечний цикл:

```
for(; ;) …;
```

Для об'єднання декількох послідовно записаних підвиразів оператора for в один загальний вираз застосовується операція «кома». Так, наприклад, наступний оператор обчислює значення суми s чисел від 1 до 10:

```
for(n=1,s=0;n<=10;n++) s+=n;
```

При використанні оператора for дозволяються внутрішні оголошення змінних циклу, але ці змінні можна використовувати лише у межах даного циклу, наприклад:

```
for(int i=1;i<=n;i++) …;
```

Цикл із передумовою (while) працює за наступним синтаксисом (рис. 7.1, б) і застосовується для повторного виконання оператора упродовж усього часу, доки значення виразу є істинним (ненульовим):

while (вираз) оператор;

У циклі з передумовою перевірка умови виконання циклу (значення виразу) відбувається ще до першого виконання його тіла. Якщо умова істинна (ненульове значення виразу), то виконується оператор (тіло циклу), інакше – цикл не виконується, тобто цикл виконується, доки умова не стане хибною. Це означає, що тіло циклу з передумовою може не виконатися жодного разу. Тіло циклу має містити хоча б один оператор, що впливає на умову закінчення ітераційного процесу, інакше відбудеться «зациклення».

Наступний фрагмент програми виводить на екран латинський алфавіт у верхньому регістрі:

```
int c=65;
while(c<=′Z′) printf(″%c″,c++);
```

Оператор циклу з постумовою (do–while) є «перевернутим» циклом

while, тобто виконується оператор, а вже потім перевіряється значення виразу (рис. 7.1, в). Цикл працює за наступним синтаксисом і повторюється доти, доки значення виразу буде істинним (ненульовим):

do оператор; while (вираз);

Отже цикл з постумовою за будь-яких обставин буде виконано принаймні один раз – в цьому і полягає його головна відмінність від циклу з передумовою.

Наступний фрагмент програми виводить на екран латинський алфавіт у верхньому регістрі, тобто виконує ті ж самі дії, що і попередній фрагмент, але за допомогою циклу do–while:

```
int c=65;
do print(″%c″,c++); while(c<=′Z′);
```

Зауважимо, що оператори тіла циклу з постумовою мають впливати на умову закінчення ітераційного процесу, інакше відбудеться «зациклення».

Для передчасного завершення будь-якого циклу застосовуються оператори переходу **break** та **continue**. Для примусового переривання циклу слід викликати оператор break. Оператор continue здійснює пропуск усіх інструкцій, записаних за її викликом в тілі циклу, та керування передається на перевірку виразу, що обумовлює завершення (або продовження) циклу.

У разі повторного виконання декількох операторів маємо складений оператор циклу. Його тіло обмежується фігурними дужками і може містити будь-які оператори, у тому числі і циклу.

Приклад завдання 5.1:

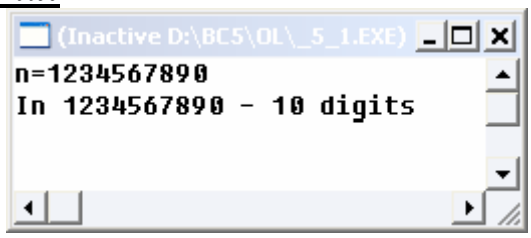
Дано натуральне число *n*. Визначити скільки цифр містить це число.

Розв'язок:

```
/* 5.1 */
#include <stdio.h>
main()
{
  unsigned int x=1;
  unsigned long int n,y;
  printf("n=");
  scanf("%li",&n);
  y=n;
  while((y/=10)>=1) x++;
  printf("In %li – %hi digits",n,x);
```

```
  return 0;
}
```

Результати роботи:



Завдання:

1. Дано натуральні числа *a* і *b*. Обчислити суму та добуток усіх чисел, котрі знаходяться між *a* і *b*.
2. Дано натуральні числа *a* і *b*. Обчислити середнє арифметичне усіх парних чисел, котрі знаходяться між *a* і *b*.
3. Два натуральних числа називаються «дружніми», якщо кожне з них дорівнює сумі дільників іншого за винятком самого числа. Дано натуральні числа *a* і *b*. З'ясувати, чи є задані числа дружніми. Приклади пар дружніх чисел: 220 і 284, 1184 і 1210, 2620 і 2924, 5020 і 5564, …
4. Дано натуральне число *n*. Обчислити факторіал *n* за формулою $n!=1\cdot 2\cdot\ldots\cdot n$.
5. Дано натуральне число *n*. Обчислити $S=-1/3+1/5-\ldots\pm 1/(2n+1)$
6. Дано натуральне число *n*. Обчислити $S=1+1/2+1/3+\ldots+1/n$.
7. Дано натуральне число *n*. Обчислити $S=1+1/2!+1/3!\ldots+1/n!$.
8. Дано натуральне число *n*. Обчислити $S=\sin 1+\sin^2 2+\ldots+\sin^n n$.
9. Дано натуральні числа *a* і *b*. Обчислити a^b без використання функцій.
10. Дано дійсні числа *a*, *b* і *c*. Обчислити значення функції $y=e^x\cos x$ з кроком *c* на відрізку $x\in[a,b]$.
11. Дано дійсне число *a*. Знайти серед чисел $1, 1+1/2, 1+1/2+1/3,\ldots$ таке, що більше за *a*.
12. Дано натуральне число *n*. Обчислити подвійний факторіал *n* за формулою $n!!=1\cdot 3\cdot 5\cdot\ldots\cdot n$ для непарного *n* або за формулою $n!!=2\cdot 4\cdot\ldots\cdot n$ для парного *n*.
13. Простим називається число, що має тільки два дільники – саме число й одиницю. З'ясувати, чи є задане натуральне число *n* простим.
14. Дано натуральне число *n* та дійсне число *x*. Обчислити $\sum_{i=1}^{n}\left(1/i!+\sqrt{|x|}\right)$.
15. Дано натуральне число *n* та дійсне число *x*. Обчислити $\prod_{i=1}^{n}\left(1+\sin(ix)/i!\right)$
16. Дано дійсне число x та натуральне число k ($0<k\le 10$). Обчислити приблизне значення $\sin(x)=x-x^3/3!+x^5/5!-\ldots$ з точністю 10^{-k}. Порівняти

отримане значення з результатом застосування відповідної бібліотечної функції.

17. Дано натуральне число *n*. Як найменшою кількістю монет можна виплатити грошову суму *n*, якщо в достатній кількості є монети гідністю в 1, 2, 5, 10, 25 та 50 коп.

18. Дано натуральне число *n*. Вивести усі його дільники.

19. Дано натуральне число *n*. Вивести це число у зворотному порядку.

20. Дано натуральне число *n*. Обчислити добуток його першої і останньої цифр.

21. Дано натуральні числа *n* і *m* (*m*<10). З'ясувати, чи входить цифра *m* в запис числа *n*.

22. Дано натуральне число *n*. Визначити суму його цифр.

23. Дано натуральні числа *n* і *m*. Визначити суму *m* останніх цифр числа *n*.

24. Дано натуральне число *n*. Визначити його першу цифру.

25. Дано натуральне число *n*. Обчислити $S = \frac{1}{1 \cdot 2} - \frac{1}{2 \cdot 3} + \ldots \pm \frac{1}{n(n+1)}$.

Контрольні запитання:

1. Коли застосовуються ітераційні процеси та що таке цикли?
2. Які існують різновиди циклів?
3. Який синтаксис операторів циклу C/C++?
4. Яке призначення мають вирази оператора for?
5. Як застосовується операція «кома» у заголовку циклу for?
6. Як виконуються внутрішні оголошення змінних циклу?
7. У чому полягає відмінність різних умовних циклів?
8. Що може спричинити «зациклення» програми?
9. Як працюють оператори переривання циклу break та continue?

8. Практична робота **№6.**
Реалізація вкладених циклічних алгоритмів

***Мета роботи*:** удосконалювання навичок роботи з алгоритмічною конструкцією повторення (циклами) засобами Borland C++; набуття навичок розв'язку задач, що потребують застосування вкладених ітераційних процесів.

***Теоретичні основи*.**

Якщо оператор циклу мстить інші циклічні оператори, то йдеться про вкладені цикли (цикли у циклі). Ступінь вкладеності при цьому може бути достатньо глибоким, в залежності від складності ітераційного процесу. Окрім того, цикли одного типу можуть бути вкладеними у цикл іншого типу.

Вкладені цикли працюють наступним чином. Виконується перша ітерація зовнішнього циклу і якщо у циклі знаходиться оператор вкладеного циклу, то програма передає керування на його виконання. Після завершення усіх ітерацій вкладеного циклу починається наступна ітерація зовнішнього циклу, під час якої знову передається керування на оператор вкладеного циклу і т.д. Алгоритм закінчується після завершення усіх зовнішніх ітерацій.

Приклади вкладених циклів:

```
for(i=first; i<=last; i++)
  for(j=v; j>=w; j--) …;

while(x<=n)
  for(i=last; i>=first; i-=x) …;
```

Неприпустимо перетинання окремих циклів.

Якщо для примусового переривання циклу використовується оператор переходу break, то переривається той цикл, у якому він викликається. Після переривання внутрішнього циклу керування передається заголовку зовнішнього циклу для перевірки умови його продовження. Аналогічно працює і оператор continue. Його виклик забезпечує виконання нової ітерації того циклу, в якому цей виклик здійснено.

Як і раніше, під час застосування вкладених циклічних алгоритмів необхідно забезпечувати скінченність ітераційного процесу аби уникнути можливого «зациклення».

Взагалі рекомендується уникати застосування операторів переходу для передчасного завершення як простих, так і вкладених циклічних процесів.

Приклад завдання 6.1:

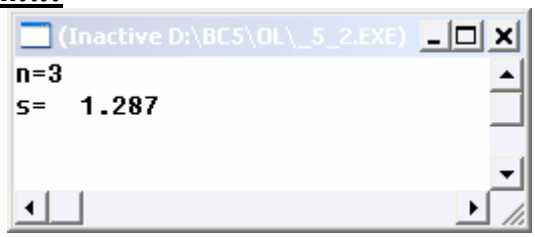
Дано натуральне число *n*. Обчислити суму ряду за формулою:

$$S = 1 + \left(\frac{1}{2}\right)^2 + \left(\frac{1}{3}\right)^3 + \ldots + \left(\frac{1}{n}\right)^n$$

Розв'язок:

```
/* 6.1 */
#include <stdio.h>
main()
{
  float s=0,a,p;
  unsigned int j,i,n;
  printf("n=");
  scanf("%li",&n);
  for(i=1;i<=n;i++)
  {
    p=a=1.0/i;
    for(j=2;j<=i;j++) p*=a;
    s+=p;
  }
  printf("s=%7.3f",s);
  return 0;
}
```

Результати роботи:



Завдання:

1. Будь-яке натуральне число *n* (*n*>7) можна отримати за формулою *n*=3*a*+5*b*, де *a* і *b* – натуральні числа. За заданим *n* знайти усі *a* і *b*.
2. Дано дійсне число *a* ($1 \le a \le 1{,}25$) . Обчислити $S = 1 + (1/2)^2 + (1/3)^3 + \ldots$, доки *S* не стане більше за *a*, не використовуючи бібліотечних функцій.
3. Дано натуральне число *n*. Обчислити $S = 1 + (1/2)^n + \ldots + (1/n)^n$, не використовуючи бібліотечних функцій.
4. Дано натуральне число *n*. Обчислити $P = (1/2)^n (1/3)^n \ldots (1/n)^n$, не використовуючи бібліотечних функцій.
5. Дано натуральне число *n*. Обчислити $P = (1+1/1)(1+1/2)^2 \ldots (1+1/n)^n$, не

використовуючи бібліотечних функцій.

6. Дано натуральне число *n*. Знайти всі Піфагорові трійки натуральних чисел, кожне з яких не перевищує *n*, тобто всі такі трійки натуральних чисел *a*, *b*, *c*, що $a^2+b^2=c^2$ ($a \le b \le c \le n$).

7. Дано натуральне число *n*. Обчислити $S = 1 + (1/2)^{n-1} + (1/3)^{n-2} + \ldots + (1/n)$, не використовуючи бібліотечних функцій.

8. Дано натуральне число *n* та дійсне число *s*. Обчислити значення функції $y = 1 + \sin x + \sin x^2 + \ldots + \sin x^n$ на відрізку $x \in [0, 2\pi]$ з кроком *s*.

9. Дано натуральне число *n* та дійсне число *s*. Обчислити значення функції $y = 1 + \cos x + \cos^2 x + \ldots + \cos^n x$ на відрізку $x \in [0, \pi]$ з кроком *s*.

10. Дано натуральне число *n*. Простим називається число, що має тільки два дільники – саме число й одиницю. Вивести всі прості числа, що менші за *n*.

11. Дано натуральне число *n*. Простим називається число, що має тільки два дільники – саме число й одиницю. Вивести *n* перших простих чисел.

12. Дано натуральне число *n*. Обчислити $\sum_{i=1}^{n} \frac{1}{(i^2)!}$.

13. Дано натуральне число *n*. Обчислити $\prod_{i,j=1}^{n} \sin(2i + j^2)$.

14. Два натуральних числа називаються «дружніми», якщо кожне з них дорівнює сумі дільників іншого за винятком самого числа. Дано натуральні числа *a* і *b*. З'ясувати, чи є задані числа дружніми. Приклади пар дружніх чисел: 220 і 284, 1184 і 1210, 2620 і 2924, 5020 і 5564, …

15. Дано натуральне число *n*. Обчислити $\sum_{i=1}^{n} \sum_{j=i}^{n} \frac{j - i + 1}{2i + j}$.

16. Дано натуральне число *n*. Обчислити $\sum_{i=1}^{n} \sum_{j=1}^{i} \frac{i}{i + 2j}$.

17. Дано натуральне число *n*. Обчислити $\sum_{i=1}^{n} i^i$, не використовуючи бібліотечних функцій.

18. Дано натуральне число *n*. Знайти всі двійки натуральних чисел *x* і *y*, якщо такі є, що $n = x^3 + y^3$.

19. Дано натуральне число *n*. Вивести послідовність $1, (1/2)^n, \ldots, (1/n)^n$, не використовуючи бібліотечних функцій.

20. Дано натуральне число *n*. Знайти число від 1 до *n*, що має якомога більше дільників.

21. Дано натуральне число *n*. Знайти число від 1 до *n*, що має якомога більшу суму дільників.

22. Дано натуральне число *n*. Знайти всі трійки натуральних чисел *x*, *y* і *z*, якщо такі є, що $n = x^2 + y^2 + z^2$.

23. Дано натуральне число *n*. Визначити суму цифр заданого числа без застосування операцій цілочислового ділення.

24. Дано натуральне число *n*. Визначити першу цифру заданого числа без застосування операцій цілочислового ділення.

25. Дано натуральне число. Отримати без застосування операцій цілочислового ділення це число, записане у зворотному порядку.

Контрольні запитання:

1. Що означає термін «вкладені цикли»? Як вони працюють?
2. Яка може бути ступінь вкладеності ітераційного процесу?
3. Чи можуть цикли перетинатись?
4. Чи можуть біти вкладеними цикли різних типів?
5. Як працюють оператори переходу у вкладених циклах?
6. Коли вкладений циклічний процес може «зациклитись»?
7. Чому бажано уникати застосування операторів переходу для передчасного завершення циклічних процесів?

9. Практична робота **№7.**
Обробка числових та символьних послідовностей

***Мета роботи*:** поглиблення знань та навичок роботи з циклами на прикладі розв'язку задач обробки числових та символьних послідовностей засобами C/C++; ознайомлення з символьним типом даних та типовими засобами їх обробки.

***Теоретичні основи*.**

Часто у практиці програмування виникають ситуації, коли для розв'язку певної задачі доводиться обробляти не окремі дані (наприклад, числових або символьного типів), а деякі їх послідовності.

Річ у тім, що для зберігання й обробки послідовностей однотипних даних у мовах програмування зазвичай застосовуються масиви. Насправді ж багато задач може бути розв'язано без використання будь-яких структур даних.

Серед задач такого типу частіше за все виступають алгоритми накопичення, тобто коли замість зберігання усієї послідовності даних (чисел, символів тощо) у програмі оголошується спеціальна змінна, в якій саме й накопичується результат кожної попередньої ітерації.

Так, наприклад, для обрахування суми або добутку значень деякої числової послідовності зовсім нема потреби в оголошенні масиву, бо такі дії цілком можуть бути виконані за допомогою лише одного циклу, під час введення даних (див. приклад завдання 7.1).

Значення символьного (літерного) типу char – непустий символ кодової таблиці ASCII (American Standard Code Information Interchange – американський стандартний код обміну інформацією) алфавіту ПК.

Оголошення змінної символьного типу у С-програмі здійснюється наступним чином:

```
char c;
```
або
```
unsigned char c;
```

Можна разом із оголошенням символьної змінної присвоїти їй початкове значення (в одинарних лапках), наприклад:

```
char c='A' ;
```
або
```
char c='\x41'; // 41 – шістнадцятковий ASCII-код символу 'A'
```
або
```
char c='\101'; // 101 – вісімковий ASCII-код символу 'A'
```

Зображення символів у вигляді їх шістнадцяткового або вісімкового ASCII-кодів зручно лише для керуючих символів (коди ASCII '\x01'–'\x1f'). Крім того, деякі керуючі символи можна записувати за допомогою так званих керуючих послідовностей, наприклад:

'\a'	– звуковий сигнал;
'\n'	– переведення рядка;
'\r'	– повернення каретки;
'\t'	– горизонтальна табуляція;
'\v'	– вертикальна табуляція.

Використання керуючих послідовностей символів можливо всюди, де можуть вживатись друковані символи. Наприклад, використання керуючої послідовності '\n' у операторі printf() після виведення переводить курсор на наступний рядок:

```
printf("Курсор буде переведено на наступний рядок\n");
```

Символьній змінній можна присвоювати значення іншої символьної змінної, а також можна порівнювати символьні змінні між собою. Символи вважаються рівними, якщо рівні їх ASCII-коди. Більшим є символ з більшим ASCII-кодом.

До значень символьного типу корисно застосовувати макроси класифікації символів **is…**, визначені у заголовному файлі ctype.h:

isalnum	– алфавітно-цифровий символ;
isalpha	– алфавітний символ;
isascii	– символ ASCII (0<=c<=0x7E);
iscntrl	– керуючий код ASCII;
isdigit	– цифра від '0' до '9';
isgraph	– друкований символ, крім пробілу;
islower	– мала літера від 'a' до 'z';
isprint	– друкований символ або пробіл;
ispunct	– символ пунктуації;
isspace	– пробіл, табуляція, повернення каретки, переведення рядка, вертикальна табуляція;
isupper	– велика літера від 'A' до 'Z';
isxdigit	– шістнадцяткова цифра від '0' до '9' або літера від 'A' до 'F' (від 'a' до 'f').

Ці макроси повертають ненульове значення (істину), якщо символ належить до заданої множини символів, або нуль (хибність) – у противному разі.

Дія цих макросів не розповсюджується на символи з кодами ASCII більше, ніж 0x7E (наприклад, кирилицю).

Використання макросу is… розглянуто у прикладі завдання 7.2.

До даних символьного типу можуть бути застосовані наступні функції:

Функція	Призначення	Приклад	#include
getc()	– читання символу з відображенням	c=getc(stdin);	stdio.h
getch()	– читання символу без відображення	c=getch();	conio.h
getchar()	– читання символу з відображенням	c=getchar();	stdio.h
getche()	– читання символу з відображенням	c=getche();	conio.h
putc()	– відображення символу	putc(c,stdout);	stdio.h
putch()	– відображення символу	putch(c);	conio.h
putchar()	– відображення символу	putchar(c);	stdio.h

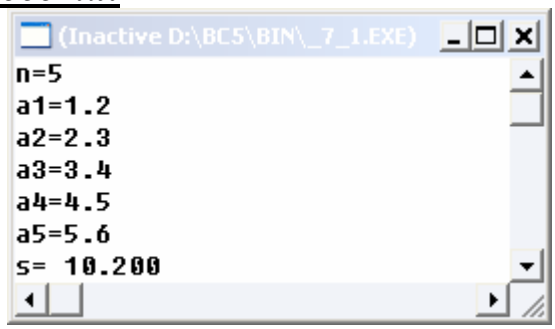
Приклад завдання 7.1:

Дано натуральне число n та дійсні числа a_1, a_2, …, a_n. Обчислити $S=a_1+a_3+\ldots+a_n$.

Розв'язок:

```
/* 7.1 */
#include <stdio.h>
main()
{
  float s=0,a;
  unsigned int i,n;
  printf("n=");
  scanf("%u",&n);
  for(i=1;i<=n;i++)
  {
    printf("a%u=",i);
    scanf("%f",&a);
    if(i%2)s+=a;
  }
  printf("s=%7.3f",s);
  return 0;
}
```

Результати роботи:



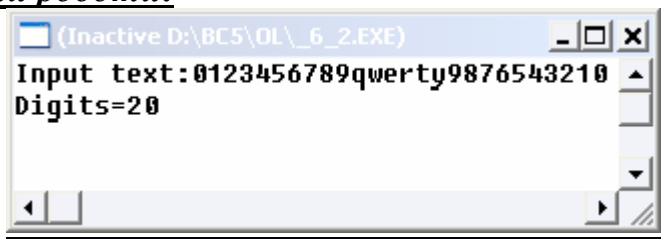
Приклад завдання 7.2:

Ввести довільну послідовність символів, закінчивши введення натисканням клавіші Enter. Визначити кількість введених цифрових символів.

Розв'язок:

```
/* 7.2 */
#include <stdio.h>
#include <conio.h>
#include <ctype.h>
main()
{
  const char enter='\r';
  unsigned int s=0;
  char c;
  printf("Input text:");
  do
  {
    c=getche(); // читання символу
    if(isdigit(c))s++; // макрос класифікації символів
  }
  while(c!=enter);
  printf("\nDigits=%u",s);
  return 0;
}
```

Результати роботи:



Завдання:

1. Дано натуральне число n та натуральні числа a_1, a_2, …, a_n. Знайти найбільший спільний дільник для заданої послідовності. Рекомендовано застосування алгоритму Евкліда.
2. Дано натуральні числа n, k, m та послідовність натуральних чисел a_1, a_2, …, a_n. Визначити кількість членів заданої послідовності, кратних k і некратних m.
3. Дано натуральне число n та послідовність натуральних чисел a_1, a_2, …, a_n. Визначити кількість парних і непарних членів заданої послідовності.
4. Дано натуральне число n та послідовність натуральних чисел a_1, a_2, …, a_n. Визначити кількість членів заданої послідовності, котрі мають парні

порядкові номери та є непарними числами.

5. Простим називається число, що має тільки два дільники – саме число й одиницю. Дано натуральне число n та послідовність натуральних чисел a_1, a_2, …, a_n. Визначити кількість простих чисел – членів заданої послідовності.

6. Дано натуральне число n та дійсні числа a_1, a_2, …, a_n. Обчислити $S = a_1^n - a_2^{n-1} + a_2^{n-2} - \ldots \pm a_n$.

7. Дано натуральне число n та послідовність дійсних чисел a_1, a_2, …, a_n. В заданій послідовності знайти максимальне і мінімальне числа.

8. Дано натуральні числа n, m та послідовність натуральних чисел a_1, a_2, …, a_n. Якщо у заданій послідовності є хоча б один член, котрий дорівнює m, то обчислити суму тих членів, що йдуть за першим таким числом; у противному разі – обчислити суму усіх членів заданої послідовності.

9. Дано натуральне число n та послідовність дійсних чисел a_1, a_2, …, a_n. Знайти у заданій послідовності таке число, котре якомога наближене до цілого числа.

10. Дано натуральне число n та послідовність дійсних чисел a_1, a_2, …, a_n. Визначити у заданій послідовності число сусідств двох додатних чисел.

11. Дано натуральне число n, дійсне число k та послідовність дійсних чисел a_1, a_2, …, a_n. Визначити скільки членів заданої послідовності із номерами 1, 2, 4, 8 тощо мають значення менше, ніж k?

12. Дано натуральне число n, дійсне число k та послідовність дійсних чисел a_1, a_2, …, a_n. Знайти у заданій послідовності таке число, котре якомога наближене до числа k.

13. Дано натуральне число n та дійсні числа a_1, a_2, …, a_n. Обчислити $P = (1 + a_1^n)(1 + a_2^n)\ldots(1 + a_n^n)$.

14. Ввести довільну послідовність символів. Визначити у заданій послідовності кількість символів, котрі належать до будь-якої абетки.

15. Ввести натуральне число x розміром у один байт та довільну послідовність символів. Визначити у заданій послідовності кількість символів, значення ASCII-коду яких збігається з числом x.

16. Ввести довільну послідовність символів. Визначити у заданій послідовності номер першого цифрового символу, якщо такий там є.

17. Ввести натуральні числа a і b розміром у один байт Сформувати таблицю ASCII-кодів символів з номерами від a до b.

18. Ввести довільну послідовність символів. Визначити у заданій послідовності символи з мінімальним та максимальним ASCII-кодами, та ці коди.

19. Ввести довільну послідовність символів. Визначити суму і добуток ASCII-кодів членів заданої послідовності.

20. Ввести довільну послідовність символів. Визначити у заданій послідовності кількість символів, котрі є цифровими та мають парні порядкові

номери.

21. Ввести натуральне число x розміром у один байт та довільну послідовність символів. Визначити у заданій послідовності такий символ, значення ASCII-коду якого якомога наближене до числа x.

22. Ввести два довільних символи c_1 і c_2. Вивести послідовність символів, значення ASCII-кодів котрих знаходяться між ASCII-кодами c_1 і c_2.

23. Ввести довільну послідовність символів. Визначити суму ASCII-кодів голосних і добуток ASCII-кодів приголосних латинських символів заданої послідовності.

24. Ввести довільну послідовність символів. Визначити у заданій послідовності кількість символів, що не є цифровими та мають значення ASCII-коду більше за код першого з введених символів.

25. Ввести довільну послідовність символів. Визначити символ, ASCII-код котрого якомога наближений до середнього арифметичного усіх членів заданої послідовності.

Контрольні запитання:

1. Що таке послідовність даних?
2. Яким чином обробляються послідовності?
3. Як за допомогою послідовностей уникають використання масивів?
4. Що таке кодова таблиця ASCII?
5. Яке призначення мають керуючі символи? Які їхні коди?
6. Як оголошуються та використовуються у програмі C/C++ змінні символьного типу?
7. Що означають та як застосовуються керуючі послідовності C/C++?
8. Коли і як застосовуються макроси класифікації символів C/C++?
9. Які стандартні функції C/C++ можуть бути застосовані до даних символьного типу?

10. Практична робота **№8.**
Розробка функцій користувача

***Мета роботи*:** ознайомлення з означенням і основними характеристиками функцій мови; засвоєння основних прийомів використання в програмах користувацьких функцій; набуття навичок написання структурованих програм C/C++.

***Теоретичні основи*.**

Функції є самостійними, логічно завершеними одиницями програми, що виконують певні дії відповідно до алгоритму розв'язку конкретної задачі. Завдяки застосування функцій спрощується процес програмування, а самі програми стають більш структурованими, зрозумілішими та лаконічними.

Кожна С-програма містить як мінімум одну функцію, а саме функцію main(), та може використовувати інші, наприклад, бібліотечні функції. Для розв'язку у програмі окремої самостійної задачі бажано розробити для неї й окрему функцію (так звану функцію користувача).

Для використання в С-програмі функції користувача її необхідно:
– оголосити;
– визначити (описати);
– викликати.

Оголошення функції зазвичай виконується перед початком функції main() і містить тип результату функції, її ім'я та список параметрів (зазначених у круглих дужках через кому):

тип Ім'я_функції (список_параметрів);

Таке оголошення називається прототипом функції, наприклад:

```
float Cube(float x);
int Fun(float y,int z);
void F1(int, double);
void F2(void);
```

Тип функції вказує тип значення, що повертається функцією в програму. Зазвичай, це один з числових типів (int, float тощо), рядок або вказівник на більш складний тип мови. Іноді функція взагалі не повертаю результату. Така функція має тип void.

Ім'я функції призначене для її ідентифікації у програмі і має бути записане за правилами побудови ідентифікаторів мови. Бажано аби ім'я функції відображувало її призначення.

Списком формальних параметрів позначаються вхідні дані, значення

яких передаються функції під час виклику у вигляді фактичних параметрів (аргументів). Кількість, черговість та типи параметрів функції має бути узгоджено з оператором виклику функції у програмі. Для кожного з параметрів необхідно вказати тип, навіть коли типи послідовних параметрів збігаються. Типи параметрів функції можуть бути довільними типами мови. Параметром функції, що не опрацьовує жодних вхідних значень, зазвичай виступає ключове слово void.

Як правило прототип функції цілком збігається з її заголовком у визначенні, проте імена параметрів у прототипі не мають ніякого значення. Їх взагалі можна не вказувати, зазначивши лише тип.

Визначення функцій (їх описи) виконуються після оголошення прототипів. Рекомендується такі визначення розташовувати після функції main().

Опис функції починається з заголовка, який містить тип результату функції, її ім'я та список параметрів (зазначених у круглих дужках через кому) і зазвичай співпадає з прототипом (але без прикінцевої крапки з комою).

Вміст самої функції (її тіло) обмежується фігурними дужками, на кшталт функції main(), між якими можна оголошувати змінні, викликати інші функції, виконувати оператори тощо. Не дозволяється у функції створювати внутрішні функції.

Додаткові змінні, що можуть бути оголошені у функції, називаються локальними або автоматичними і зберігаються у стеку, тобто є доступними лише у даній функції під час її роботи. Такі змінні рекомендується розташовувати перед першим оператором функції.

Функція завершує своє виконання або коли досягнуто її кінця (виконано усі оператори), або завдяки оператору **return**. Типізовані функції обов'язково мають містити хоча б один оператор return, одразу за яким зазначається відповідне значення, що повертається цією функцією у програму. Лише void-функції можуть використовувати оператор return без завершального значення, або зовсім його не містити.

Приклад опису функції, що обчислює третю степінь переданого їй аргументу:

```
float Cube(float x)
{
  return x*x*x;
}
```

Звертання до функції у необхідному місці програми відбувається за допомогою оператора виклику – вказується ім'я функції та перелічуються у круглих дужках її фактичні параметри (аргументи). Якщо функція повертає результат, то її виклик можна використовувати у виразах як звичайний

операнд, тип якого збігається з типом значення функції. Окрім того, виклик такої функції можна здійснити автономно, тобто через окремий оператор. У такому разі значення, що повертає функція, ігнорується. Виклик void-функції здійснюється виключно автономно.

Під час звертання до функції всі її формальні параметри отримують копії значень відповідних фактичних параметрів (аргументів), заданих оператором виклику. Тому кількість, порядок розташування та тип фактичних і формальних параметрів функції, як зазначалося раніше, повинно бути узгоджено (за винятком спеціальних функцій зі змінною кількістю параметрів). Гарантується недоторканість фактичних параметрів (аргументів) під час роботи функції. Останнє правило не розповсюджується на параметри-масиви, параметри-рядки та на деякі ще особливі параметри.

Приклади виклику функцій:

```
d=a*a+Cube(b)/3;
F1(x1,x2);
printf(″Result=%f″,Cube(a+b));
```

У першому прикладі обчислюється значення змінної d за формулою: $d=a^2+b^3/3$. Для визначення кубу змінної b застосовано виклик функції Cube(b), опис якої показано раніше за текстом. Виклик даної функції передає їй аргумент b через фактичний параметр x. Функція обчислює третю степінь параметра x і повертає її за допомогою оператора return у вираз програми у якості результату.

У другому прикладі здійснюється виклик функції, яка отримує два параметра x1 і x2 та, скоріш за все, не повертає жодного значення, тобто має тип void.

Третій приклад демонструє виклик раніше розглянутої функції для обчислення третьої степені суми двох дійсних чисел у якості аргументу іншої функції, у даному разі – printf().

Оператори виклику функцій можуть розташовуватись як у межах функції main(), так і у межах інших функцій.

Локальні змінні та формальні параметри функцій запам’ятовуються у стеку. Область дії таких змінних розповсюджується лише на дану функцію і після завершення функцій їх значення стають невідомими. У стеку запам’ятовуються також адреси повернення з функцій

На відміну від локальних, глобальні змінні запам’ятовуються у сегменті даних програми і доступні з будь-якої її функції. Використання глобальних змінних може бути корисним для оголошення великих структур даних з метою економії стекової пам’яті, але звертання у функціях до глобальних змінних програми порушує засади структурного програмування і може призвести до помилок, які дуже складно викрити та виправити.

Якщо оператор у функції містить виклик цієї ж функції, така функція

називається рекурсивною. Наприклад, обчислення факторіала числа n (n!=1·2·3·…·n) дуже зручно виконувати за допомогою рекурсивної функції:

```
double Factorial(int n)
{
  if(n>1) return n*Factorial(n–1);
  return 1;
}
```

Головною вимогою до рекурсивних функцій є необхідність забезпечення коректного виходу з них, аби не призвести до переповнення стеку і «зависанню» системи.

Передача у функції масивів, рядків та більш складних структур даних має певні особливості та потребує застосування спеціальних прийомів програмування та частково розглянуто у відповідних наступних роботах.

Приклад завдання 8.1:

Дано дійсні числа *a*, *b*, *c*, *d*, *f*, що визначають сторони та *g*, *h*, що визначають діагоналі п’ятикутника (рис. 8.1). Визначити функцію обчислення площі трикутника за його трьома сторонами та за допомогою неї обчислити загальну площу заданого п’ятикутника. Розміри сторін та діагоналей п’ятикутника ввести двома окремими рядками чисел через пробіл.

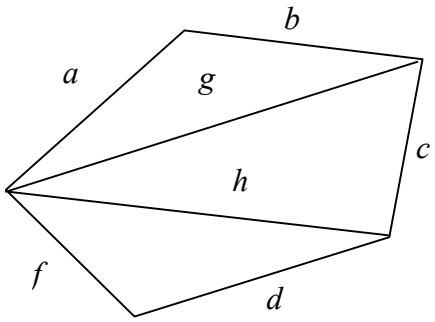


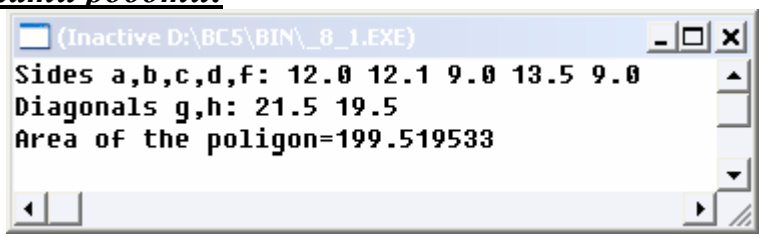
Рисунок 8.1 – До завдання 8.1

Розв’язок:

```
/* 8.1 */
#include <stdio.h>
#include <math.h>
float Triangle(float x,float y,float z);
main()
{
   float a,b,c,d,f,g,h;
   printf("Sides a,b,c,d,f: ");
   scanf("%f%f%f%f%f",&a,&b,&c,&d,&f);
   printf("Diagonals g,h: ");
   scanf("%f%f",&g,&h);
   printf("Area of the poligon=%f",
  Triangle(a,b,g)+Triangle(g,c,h)+Triangle(h,d,f));
  return 0;
}
```

```
float Triangle(float x,float y,float z)
{
  float p;
  p=0.5*(x+y+z);
  return sqrt(p*(p-x)*(p-y)*(p-z));
}
```

Результати роботи:



Завдання:

1. Дано дійсні числа a, b і c. Визначити функцію $\max(x,y)$ знаходження більшого з двох чисел та обчислити $\frac{b}{c}\max^5(a^3,b)+c^2\sqrt{\max(a+c,b)}$.
2. Дано натуральні числа a, b і c. Визначити функцію $НСД(x,y)$ знаходження найбільшого спільного дільника для двох довільних чисел та обчислити $НСД^3(a+c,b^2)+НСД^2(b,c)-НСД(ab,ac)\cdot НСД(a+b+c,b^2c)$. Рекомендовано застосування алгоритму Евкліда.
3. На площині задано n точок з дійсними координатами (x_1,y_1), (x_2,y_2), … (x_n,y_n). З'ясувати, яка з них ближча до початку координат, визначивши відповідну функцію. Кожну пару координат точок ввести через пробіл.
4. Дано дійсні числа a, b і c. Визначити функцію $\min(x,y)$ знаходження меншого з двох чисел та знайти більше серед $\min^3(a^2,b+c)$ та $\min^2(ab,c^3)$.
5. Простим називається число, що має тільки два дільники – саме число й одиницю. Дано натуральне число n та послідовність натуральних чисел a_1, a_2, …, a_n. Визначити функцію ідентифікації простого числа та за допомогою неї знайти середнє арифметичне простих чисел – членів заданої послідовності.
6. Дано три трикутника зі сторонами a_1, b_1, c_1; a_2, b_2, c_2 та a_3, b_3, c_3. Визначити функцію ідентифікації прямокутного трикутника та за допомогою неї обчислити загальну площу заданих прямокутних трикутників (якщо такі є). Числа в кожному з наборів сторін трикутника ввести через пробіл.
7. Дано натуральне число n та послідовність натуральних чисел a_1, a_2, …, a_n. Визначити функцію ідентифікації парного/непарного числа та за допомогою неї з'ясувати, чи є парним числом сума непарних чисел – членів заданої послідовності.

8. Дано натуральні числа n і x та послідовність генерованих натуральних чисел a_1, a_2, …, a_n (a_i=0…x). Визначити функцію обчислення суми цифр довільного натурального числа та за допомогою неї з'ясувати, який з членів генерованої послідовності має найбільшу таку суму.

9. На площині задано три точки з дійсними координатами (x_1,y_1), (x_2,y_2), та (x_3,y_3). Знайти площу заданого зазначеними координатами трикутника, визначивши функцію розрахунку довжини його сторін. Кожну пару координат точок ввести через пробіл.

10. На площині задано n точок з дійсними ненульовими координатами (x_1,y_1), (x_2,y_2), … (x_n,y_n). Знайти номери координатних чвертей розташування зазначених точок, визначивши відповідну функцію. Кожну пару координат точок ввести через пробіл.

11. Дано натуральні числа n і x та послідовність генерованих натуральних чисел a_1, a_2, …, a_n (a_i=0…x). Визначити функцію обчислення кількості цифр довільного натурального числа та за допомогою неї з'ясувати, який з членів генерованої послідовності має найменшу таку кількість.

12. Дано цілі числа a, b і c ($a<b<c$). Визначити функцію $Sum(x,y)$ знаходження суми всіх цілих чисел від x до y включно та обчислити $Sum^3(a,b)+3Sum^2(b,2c)-ab^5Sum(ab,bc)\sqrt{Sum(a+b,b^2+c)}$.

13. Дано дійсні числа a, b, c ($a<b<c$). Визначити функцію $Sum(x,y,h)$ знаходження суми всіх чисел від x до y включно з кроком h та обчислити $\dfrac{ab^3Sum^5(a,b,a/b)+2c^2Sum^3(b,3c,a/4c)-\sqrt[3]{Sum(ab,bc,a/(b+c))}}{abc\sqrt{Sum(2b-a,b^2+c,a/3)}}$.

14. Два натуральних числа називаються «дружніми», якщо кожне з них дорівнює сумі дільників іншого за винятком самого числа. Дано натуральні числа a і b. Визначити функцію обчислення суми дільників довільного натурального числа та за допомогою неї з'ясувати, чи є задані числа дружніми. Приклади пар дружніх чисел: 220 і 284, 1184 і 1210, 2620 і 2924, 5020 і 5564, …

15. Дано ненульове дійсне число x та n кругів, радіус яких задано послідовністю генерованих дійсних чисел a_1, a_2, …, a_n ($0<a_i\le x$). Визначити функцію обчислення площі круга за заданим радіусом та за допомогою неї з'ясувати, який з заданих кругів займає найбільшу поверхню.

16. Дано дійсні числа a і b. Знайти: $a+b$, $a-b$, $a\cdot b$ та a/b, визначивши функцію $Calc(x,y,op)$ для виконання відповідної операції op над двома довільними числами. Операції задати відповідними символьними константами.

17. Дано натуральні числа n, k, m та послідовність натуральних чисел a_1, a_2, …, a_n. Визначити функцію з'ясування кратності/некратності натуральних чисел та за допомогою неї порахувати кількість членів заданої послідовності, кратних k і некратних m.

18. Дано генеровані ненульові натуральні числа a, b і c. Визначити функцію

знаходження останньої цифри довільного натурального числа та за допомогою неї обчислити добуток останніх цифр генерованих чисел.

19. Дано ненульові натуральні числа *a*, *b* і *c*. Визначити функцію знаходження першої цифри довільного натурального числа та за допомогою неї обчислити суму перших цифр заданих чисел.

20. Дано дійсне число *a*. Визначити функцію $MyExp(x,\varepsilon)$ знаходження приблизного значення $e^x = 1 + x + x^2/2! + x^3/3! + \ldots$ з точністю ε та за допомогою неї обчислити значення e^a для $\varepsilon = 10^{-k}$ (k=1, 2, …, 5). Порівняти отримані значення з результатом застосування відповідної бібліотечної функції.

21. Дано дійсне число *a*. Визначити функцію $MySin(x,\varepsilon)$ знаходження приблизного значення $\sin(x) = x - x^3/3! + x^5/5! - \ldots$ з точністю ε та за допомогою неї обчислити значення $\sin(a)$ для $|\varepsilon| = 10^{-k}$ (k=1, 2, …, 5). Порівняти отримані значення з результатом застосування відповідної бібліотечної функції.

22. Дано дійсне число *a*. Визначити функцію $MyCos(x,\varepsilon)$ знаходження приблизного значення $\cos(x) = 1 - x^2/2! + x^4/4! - \ldots$ з точністю ε та за допомогою неї обчислити значення $\cos(a)$ для $|\varepsilon| = 10^{-k}$ (k=1, 2, …, 5). Порівняти отримані значення з результатом застосування відповідної бібліотечної функції.

23. Дано дійсне число *a*. Визначити функцію $Arctg(x,k)$ знаходження приблизного значення $\operatorname{arctg}(x) = x - x^3/3 + x^5/5 - \ldots$ з урахуванням k перших членів заданого ряду та за допомогою неї обчислити значення $\operatorname{arctg}(a)$. Кількість значущих членів ряду ввести з клавіатури. Порівняти отримане значення з результатом застосування відповідної бібліотечної функції.

24. Дано дійсне число *a*. Визначити функцію $Ln(x,k)$ знаходження приблизного значення $\ln(1+x) = x - x^2/2 + x^3/3 - \ldots$ з урахуванням k перших членів заданого ряду та за допомогою неї обчислити значення $\ln(a)$. Кількість значущих членів ряду ввести з клавіатури. Порівняти отримане значення з результатом застосування відповідної бібліотечної функції.

25. Дано ненульове дійсне число *x*, що задає радіус певного круга. Знайти площу даного круга, визначивши відповідну функцію, в якій число π обчислюється за допомогою ряду Лейбніца: $\pi/4 = 1 - 1/3 + 1/5 - \ldots$ з заданою точністю ε. Порівняти отримане значення з результатом застосування відповідної бібліотечної символічної константи для π.

Контрольні запитання:

1. Що таке функція та яке її призначення в програмі?
2. Які види функцій може містити програма C/C++?
3. Що необхідно для використання в програмі функції користувача?
4. Що таке прототип, формальні та фактичні параметри (аргументи) функції?
5. Яку структуру має функція користувача, та як здійснюється в програмі її виклик?
6. Який тип можуть мати функції та їх параметри? Що таке void-функції та void-параметри?
7. Яка функція називається рекурсивною та які вимоги для її використання в програмі?

11. Практична робота **№9.**
Робота з одновимірними числовими масивами

***Мета роботи*:** ознайомлення зі структурами даних, зокрема з одновимірними числовими масивами (векторами); набуття навичок роботи з числовими масивами у програмах C/C++.

***Теоретичні основи*.**

У масивах об'єднуються елементи одного типу. Елементи масиву розташовуються у пам'яті послідовно, починаючи з першого елементу (перший елемент має індекс 0) .

Оскільки перший індекс масиву завжди дорівнює нулю, то для будь-якого оголошення на зразок:

```
t name[n];
```

де t – тип даних, припустимі значення індексів знаходяться у діапазоні від 0 до n–1. Використання індексу за межами цього діапазону призведе до звернень до областей пам'яті, що не належать масиву.

Отже для оголошення масиву необхідно оголосити змінну (ім'я масиву) та додати слідом у квадратних дужках ціле число, котре буде визначати кількість елементів у масиві. Наприклад, оголошення:

```
int m[100];
```

визначає масив на ім'я m, здатний зберігати до 100 значень типу int.

Типові приклади оголошення вищезгаданого масиву:

```
#define MAX 100
…
int m[MAX];
```

або

```
const max=100;
int m[max];
```

Аби сформувати вираз, що має відношення до конкретного елементу масиву, необхідно після імені масиву зазначити індекс цього елементу (у квадратних дужках). У якості індексу має бути цілий вираз – константа, змінна, результат функції тощо. Наприклад, оператор:

```
m[5]=55;
```

присвоює значення 55 шостому елементу масиву m.

Послідовний доступ до декількох елементів масиву здійснюється за допомогою циклічних алгоритмів. Наприклад, наступний фрагмент програми присвоює нуль значенням перших n елементів масиву:

```
for (int i=0;i<n;i++) m[i]=0;
```

Оголошення масивів можуть бути глобальними і локальними. Якщо змінна-масив – глобальна, то елементи масиву автоматично ініціалізуються нулями. Якщо ж масив локальний у функції, то його значення є непередбачуваними і потребують ініціалізації.

Для ініціалізації необхідно оголосити масив (як зазвичай), після чого у фігурних дужках записати (через кому) список значень констант. Наприклад, оголошення:

```
int m[10]={0,1,2,3,4,5,6,7,8,9};
```

створює масив із десяти цілих значень на ім’я m і послідовно присвоює значення від 0 до 9 кожному елементу масиву від m[0] до m[9].

Можна переписати розглянуте оголошення наступним чином:

```
int m[]={0,1,2,3,4,5,6,7,8,9};
```

У такому разі порожні квадратні дужки дозволяють компілятору автоматично визначати точний розмір пам’яті, необхідний для зберігання перерахованих елементів.

При явному завданні розміру масиву не є обов’язковим завдання значень для кожної позиції масиву. Наприклад, оголошення:

```
int m[10]={0,1,2,3,4};
```

створює масив із десяти цілих значень, але ініціалізує лише перші п’ять.

Завдання зайвої кількості елементів у фігурних дужках призводить до помилки компіляції.

Якщо m – це ім’я певного масиву, то вираз sizeof(m) визначає число байт, що займає масив m у пам’яті, а sizeof(m[0]) – число байт, що займає один елемент масиву.

Під час попередньої ініціалізації масиву можна використовувати sizeof() для коректного визначення числа елементів масиву, наприклад:

```
int m[]={0,1,2,3,4};
#define MAX sizeof(m)/sizeof(m[0])
```

Аби не допустити подальших корегувань масиву, необхідно перед

його оголошенням вказати ключове слово const, наприклад:

```
const float mas[]={0,1,2,3,4,5,6,7,8,9};
```

Зазвичай початковими значеннями масив заповнюється з клавіатури або файлу за допомогою циклу. Так, наприклад, наступний фрагмент програми демонструє заповнення масиву цілих чисел з клавіатури. Накопичення значень завершується коли користувачем введено число нуль, або коли вичерпано число комірок (MAX), попередньо виділене для зберігання масиву директивою #define:

```
…
n=0; /* n – лічильник елементів масиву */
do
{
  printf("m[%i]=",n);
  scanf("%i",&m[n]);
}
while((m[n++])&&(n!=MAX));
n=(m[n–1])?n:--n;
…
```

Масиви можна передавати у користувацькі функції. Так, наступна функція обраховує і повертає суму значень деякого масиву дійсних чисел:

```
double Sum(double data[], int k)
{
  double s=0;
  while(k>0) s+=data[--k];
  return s;
}
```

Викликати цю функцію можна, наприклад, наступним чином:

```
printf(″Result=%lf\n″,Sum(m,n));
```

де n – кількість елементів у масиві m.

При передачі масиву у функцію передається не вміст масиву, а лише його адреса (ім’я масиву). Це означає, що функція оперує безпосередньо з параметром-масивом, тобто може змінювати його значення. Для уникнення таких змін використовуються параметри-константи, наприклад:

```
void Func(const double data[], int k)
```

Можна також формальний параметр, через який задається масив, оголосити як вказівник, а для звертання у функції до елементів масиву використовувати операцію розадресації.

Задача пошуку у масиві полягає у визначенні номерів певних елементів масиву або їх значень. Розрізняють задачі пошуку в упорядкованому та неупорядкованому масивах. В неупорядкованому масиві пошук можна здійснити лише за допомогою перегляду всього масиву (лінійний пошук). Якщо масив упорядкований (відсортований), то до нього може бути застосовано спеціальні методи пошуку, наприклад, бінарний пошук тощо.

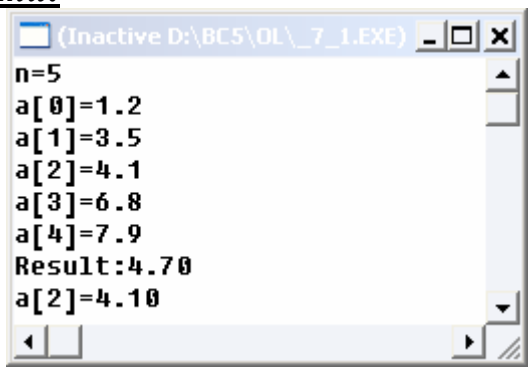
Приклад завдання 9.1:

Дано натуральне число n та масив дійсних чисел $a_1, a_2, \ldots, a_n$. Знайти серед членів масиву таке число (і його індекс), якомога наближене до середнього арифметичного усіх елементів масиву.

Розв'язок:

```
/* 9.1 */
#include <stdio.h>
#include <math.h>
#define MAX 100
main()
{
  float a[MAX],x,y,s=0;
  unsigned int i,n,k;
  printf("n="); scanf("%i",&n);
  for(i=0;i<n;i++)
  {
    printf("a[%i]=",i); scanf("%f",&a[i]);
    s+=a[i];
  }
  x=fabs(a[0]–s/n);
  y=a[0];
  k=0;
  for(i=1;i<n;i++)
    if(x=(fabs(a[i]–s/n)<x))
    {
      y=a[i];
      k=i;
    }
  printf("Result:%–7.2f\n",s/n);
  printf("a[%i]=%–7.2f",k,y);
  return 0;
}
```

Результати роботи:



Завдання

(**в задачах введення кожного елементу масиву завершується натисканням Enter, введення припиняється введенням числа 0**):

1. Дано два довільні одновимірні масиви цілих чисел M_1 і M_2 з однаковою кількістю елементів. Сформувати масив M_3, елементами якого є числа, що утворюються за формулою $M_3[i]=M_1[i]+M_2[i]$.
2. З довільного одновимірного масиву дійсних чисел M_1 сформувати масиви: M_2, який складається з додатних, і M_3, який складається з від'ємних елементів M_1.
3. З довільного одновимірного масиву дійсних чисел M_1 сформувати масив M_2, що складається з елементів M_1, які більші за середнє арифметичне елементів M_1.
4. У довільному одновимірному масиві натуральних чисел M знайти та вивести першу наявну послідовність чисел Фібоначчі, тобто таку, що задовольняє умовам: $M[i]=1$, $M[i+1]=1$, $M[i+2]=M[i]+M[i+1]$.
5. З двох довільних одновимірних масивів цілих чисел M_1 і M_2 сформувати масив M_3, що складається лише з тих елементів, котрі присутні і у M_1 і у M_2.
6. Дано два довільні одновимірні масиви цілих чисел M_1 і M_2. Сформувати масив M_3, елементами якого є ті члени M_2, що не присутні у M_1.
7. Дано натуральне число n. Сформувати масив, елементами якого є цифри числа n.
8. У довільному одновимірному масиві дійсних чисел M обчислити довжину останньої послідовності, що задовольняє умові $M[i+1]=M[i]$.
9. Знайти найбільший спільний дільник для елементів довільного одновимірного масиву цілих чисел. Рекомендовано застосування алгоритму Евкліда.
10. У довільному одновимірному масиві цілих чисел M знайти і вивести останню послідовність, що задовольняє умові $M[i+1]=M[i]+1$.
11. У довільному одновимірному масиві цілих чисел M знайти і вивести першу послідовність, що задовольняє умові $M[i]=M[i+1]+1$.
12. У довільному одновимірному масиві цілих чисел подвоїти непарні та потроїти парні елементи.

13. У довільному одновимірному масиві цілих чисел M видалити елементи, які менші за середнє арифметичне елементів M. Визначити кількість видалень.

14. Дано довільні одновимірний масив цілих чисел M та натуральне число n. Визначити, якщо такі є, кількість чисел n у масиві M та їх індекси.

15. Дано довільні одновимірний масив дійсних чисел M_1 та дійсні числа x та y ($x<y$). З масиву M_1 сформувати масив M_2, що складається з елементів M_1, які більші за число x та менші за число y.

16. Дано довільні одновимірний масив цілих чисел M та натуральне число n. З масиву M_1 сформувати масив M_2, що складається з елементів M_1, які не дорівнюють числу n.

17. Дано два довільні одновимірні масиви цілих чисел M_1 і M_2 з однаковою кількістю елементів n. Сформувати масив M_3, елементами якого є числа, що утворюються за формулою $M_3[i]=\left|M_1[i]-M_2[n-i-1]\right|$.

18. Дано два довільні одновимірні масиви дійних чисел M_1 і M_2 та дійсні числа x і y ($x<y$). Сформувати масив M_3, елементами якого є ті члени M_1, котрі більші за x, та ті члени M_2, котрі менші за y. Спочатку розташувати елементи масиву M_1.

19. Дано довільні одновимірний масив цілих чисел M та натуральне число n. У масиві M видалити елементи, які дорівнюють n. Визначити кількість видалень.

20. Дано довільні одновимірний масив дійсних чисел M_1 та дійсні числа x та y ($x<y$). З масиву M_1 сформувати масив M_2, в якому замінено нулями усі числа, котрі більші за x та менші за y.

21. З довільного одновимірного масиву цілих чисел M_1 сформувати масив M_2, що складається з таких елементів, що не повторювалися у M_1.

22. У довільному одновимірному масиві цілих чисел визначити який з елементів (максимальний чи мінімальний) знаходиться ближче до середини масиву.

23. У довільному одновимірному масиві цілих чисел розташувати на початку масиву мінімальний, а на при кінці максимальний за значенням елементи масиву, не змінюючи порядку розташування інших елементів.

24. Дано довільні одновимірний масив дійсних чисел M та натуральне число n. У масиві M збільшити у n разів ті елементи, що мають парні порядкові номери та менші за n але більші за середнє арифметичне серед елементів M з непарними порядковими номерами.

25. Дано два довільні одновимірні масиви цілих чисел M_1 і M_2 та натуральне число n. Сформувати масив дійсних чисел M_3, елементами якого є числа, що утворюються за формулою $M_3[i]=M_1[i]\cdot M_2[i]$. Елементи, яких недостає у меншому за розміром масиві, вважати рівними n.

Контрольні запитання:

1. Коли у програмах C/C++ застосовуються масиви?
2. Який синтаксис оголошення масиву у C/C++?
3. Що таке розмір та що таке розмірність масиву?
4. Як можна ініціалізувати масив у C/C++?
5. Як здійснюється індексація елементів масиву?
6. Як здійснюється у програмі звертання до окремих елементів масиву?
7. Яким чином можна опрацьовувати одразу декілька елементів масиву?
8. Як здійснюється передача і опрацювання масивів у функціях?
9. Що таке задача пошуку у масиві?

12. Практична робота **№10.**
Робота з числовими матрицями

***Мета роботи*:** ознайомлення зі структурами даних, зокрема з двовимірними числовими масивами (матрицями); набуття навичок роботи з числовими матрицями у програмах C/C++.

***Теоретичні основи*.**

Масив може мати два або й більше вимірів. Такі масиви називаються багатовимірними.

Найбільш розповсюдженим на практиці є використання двовимірних масивів або так званих матриць. Елементи таких масивів запам'ятовуються рядок за рядком. Можна вважати, що двовимірний масив є масивом одновимірних масивів (підмасивів).

Так, наприклад, після оголошення у програмі матриці дійсних чисел:

```
int x[3][4];
```

розташування її елементів x[i][j] буде відповідати наступній таблиці:

x[0][0]	x[0][1]	x[0][2]	x[0][3]
x[1][0]	x[1][1]	x[1][2]	x[1][3]
x[2][0]	x[2][1]	x[2][2]	x[2][3]

де перший індекс i є номером рядка, а другий j – номером стовпця (індексація кожного з вимірів починається з нуля).

У пам'яті комп'ютера елементи такого масиву розташується наступним чином: x[0][0], x[0][1], x[0][2], x[0][3], x[1][0], x[1][1], x[1][2], x[1][3], x[2][0], x[2][1], x[2][2], x[2][3] і займають 24 (12×2) байт пам'яті.

Ініціалізувати такий масив можна в один із наступних способів:

```
float x[3][4]={0.0,0.1,0.2,0.3,1.0,1.1,1.2,1.3,2.0,2.1,2.2,2.3};
```

або

```
float x[3][4]=
{
  {0.0,0.1,0.2,0.3},
  {1.0,1.1,1.2,1.3},
  {2.0,2.1,2.2,2.3}
};
```

де, наприклад, вираз x[1][2] відповідає значенню 1.2.

До речі, в оголошенні багатовимірного масиву дозволяється залишати порожніми перші квадратні дужки, тобто опускати розмірність найста-

ршого виміру, наприклад:

```
float x[][4]={0.0,0.1,0.2,0.3,1.0,1.1,1.2,1.3,2.0,2.1,2.2,2.3};
```

У цьому випадку кількість рядків матриці обчислюється як частка від ділення кількості даних (у даному прикладі їх дванадцять) на відому кількість елементів у кожному рядку матриці (у даному прикладі – чотири).

Матриця може складатися з одного рядка, або з одного стовпця, або навіть мати лише один елемент. В останньому випадку оголошення та ініціалізація матриці може виглядати, наприклад, наступним чином:

```
int x[1][1]={100};
```

Для доступу до елементів багатовимірного масиву застосовується класична індексна форма звертання, тобто вказавши в окремих квадратних дужках значення індексу для кожного з вимірів, наприклад:

```
x[1][2]=(a+b)/c;
```

або

```
int i,j,n=3,m=4;
for(i=0;i<n;i++)
  for(j=0;j<m;j++) printf("%f ",x[i][j]);
```

В останньому прикладі розглянуто код виведення на екран дванадцяти дійсних чисел – елементів двовимірного масиву.

Як вже зазначалося, при передачі масиву у функцію передається не вміст масиву, а лише його адреса (ім’я масиву).

Для передачі багатовимірного масиву у якості аргументу функції необхідно задати мінімум інформації, достатньої для перетворення індексних виразів в адреси пам’яті. Зокрема для двовимірного масиву достатньо передати кількість стовпців аби компілятор зміг самостійно обчислити адреси елементів на початку кожного рядка.

Визначимо, наприклад, символічні константи ROWS і COLS, що задають кількість рядків та стовбців відповідно у двовимірному масиві:

```
#define ROWS 5
#define COLS 8
```

Тепер можна оголосити, наприклад, наступну функцію AnyFn() з масивом у якості параметра:

```
void AnyFn(int data[ROWS][COLS]);
```

Можна навіть не зазначати кількість рядків:

```
void AnyFn(int data[][COLS]);
```

В обох випадках компілятор перетворить вираз data[n][m] в адресу.

Аби організувати передачу в одну й ту ж саму функцію масивів з різною кількістю рядків необхідно в оголошенні функції зазначити додатковий параметр, наприклад:

```
long Sum(int data[][COLS],int numRows);
```

Тут параметр numRows повідомляє функції AnyFn(), скільки рядків містить масив data. Глобальна константа COLS, як і раніше, інформує компілятор і функцію про кількість стовпців в одному рядку.

Так, наступна функція обраховує і повертає суму значень деякого двовимірного масиву цілих чисел:

```
long Sum(int data[][COLS],int numRows)
{
  long s=0;
  for(int r=0;r<numRows;r++)
    for(int c=0;c< COLS;c++) s+=data[r][c];
  return s;
}
```

Викликати цю функцію можна тепер для двовимірних масивів з різною кількістю рядків, наприклад, наступним чином:

```
printf(″Result1=%li\n″,Sum(matrix1,n1));
printf(″Result2=%li″,Sum(matrix2,n2));
```

де n1 і n2 – відповідна кількість рядків у двовимірних масивах matrix1 і matrix2, попередньо оголошених, наприклад, наступним чином:

```
…
#define COLS 8
…
const int n1=5,n2=7;
int matrix1[n1][COLS], matrix2[n2][COLS];
```

Матриці доволі часто застосовуються у математиці, особливо під час розв’язку задач, що містять системи алгебраїчних та диференційних рівнянь.

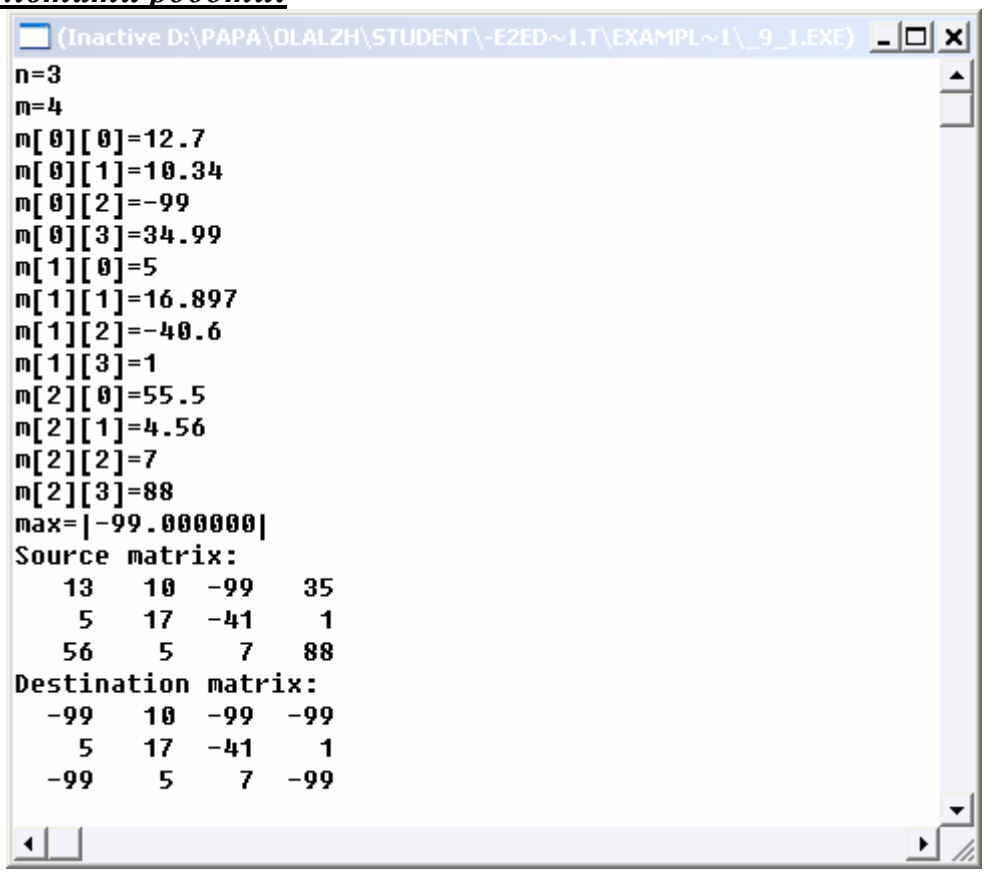
Приклад завдання 10.1:

Дано натуральні числа *n* та *m*, що визначають відповідно число рядків та стовпців у деякій заданій матриці дійсних чисел. Знайти серед членів цієї матриці максимальний за абсолютним значенням елемент та замінити ним кутові елементи.

Розв'язок:

```
/* 10.1 */
#include <stdio.h>
#include <math.h>
#define ROWS 10
#define COLS 10
main()
{
  float matrix[ROWS][COLS],a;
  unsigned int i,j,n,m;
  printf("n="); scanf("%i",&n);
  printf("m="); scanf("%i",&m);
  for(i=0;i<n;i++)
    for(j=0;j<m;j++)
    {
      printf("m[%i][%i]=",i,j);
      scanf("%f",&matrix[i][j]);
      if((!i)&&(i==j)) a=matrix[i][j];
      else if(fabs(matrix[i][j])>fabs(a)) a=matrix[i][j];
    }
  printf("max=|%f|\n",a);
  printf("Source matrix:\n");
  for(i=0;i<n;i++)
  {
    for(j=0;j<m;j++) printf("%5.0f",matrix[i][j]);
    printf("\n");
  }
  matrix[0][0]=matrix[0][m–1]=matrix[n–1][0]=matrix[n–1][m–1]=a;
  printf("Destination matrix:\n");
  for(i=0;i<n;i++)
  {
    for(j=0;j<m;j++) printf("%5.0f",matrix[i][j]);
    printf("\n");
  }
  return 0;
}
```

Результати роботи:



Завдання:

1. Дано натуральні числа *n* та *m*, що визначають відповідно кількість рядків та стовпців у деякій заданій матриці дійсних чисел. Знайти серед членів цієї матриці максимальний за абсолютним значенням елемент та замінити ним елементи першого стовпця.
2. Дано натуральні числа *n* та *m*, що визначають відповідно кількість рядків та стовпців у деякій заданій матриці дійсних чисел. Знайти серед членів цієї матриці мінімальний за абсолютним значенням елемент та замінити ним елементи останнього рядка.
3. Дано натуральні числа *n* та *m*, що визначають відповідно кількість рядків та стовпців у деякій заданій матриці дійсних чисел, а також довільне дійсне число *x*. Знайти серед членів цієї матриці усі, більші за x, елементи та їхні індекси.
4. Дано натуральні числа *n* та *m*, що визначають відповідно кількість рядків та стовпців у деякій заданій матриці дійсних чисел, а також генероване дійсне число *x*. Знайти серед членів цієї матриці усі, менші за x, елементи та їхню суму.
5. Дано натуральні числа *n* та *m*, що визначають відповідно кількість рядків та стовпців у деякій заданій матриці дійсних чисел. Знайти периметр цієї матриці (суму елементів першого та останнього стовпців та рядків).

6. Дано натуральні числа n та m, що визначають відповідно кількість рядків та стовпців у деякій заданій матриці дійсних чисел, а також генероване дійсне число x. Замінити числом x усі елементи, що знаходяться на периметрі цієї матриці (елементи першого та останнього стовпців та рядків).

7. Дано натуральне число n, що визначає кількість рядків та стовпців у деякій генерованій квадратній матриці дійсних чисел. Знайти слід (суму елементів головної діагоналі) цієї матриці.

8. Дано натуральне число n, що визначає кількість рядків та стовпців у деякій генерованій квадратній матриці дійсних чисел. Знайти добуток елементів головної та побічної діагоналей цієї матриці.

9. Дано натуральні числа n та m, що визначають відповідно кількість рядків та стовпців у деяких генерованих матрицях натуральних чисел M_1 і M_2. Отримати матрицю M_3 шляхом додавання матриць M_1 і M_2.

10. Дано натуральне число n, що визначає кількість рядків та стовпців у деякій заданій квадратній матриці цілих чисел M_1. Виконати транспонування цієї матриці у матрицю M_2, тобто рядки матриці зробити стовпцями і навпаки.

11. Дано натуральне число n, що визначає кількість рядків та стовпців у деякій генерованій квадратній матриці дійсних чисел. Переставити місцями елементи головної та побічної діагоналей цієї матриці.

12. Дано натуральні числа n та m, що визначають відповідно кількість рядків та стовпців у деякій заданій матриці дійсних чисел M_1. Отримати матрицю M_2 шляхом обертання матриці M_1 на 90^0 за годинниковою стрілкою.

13. Дано натуральні числа n та m, що визначають відповідно кількість рядків та стовпців у деякій заданій матриці дійсних чисел M_1. Отримати матрицю M_2 шляхом обертання матриці M_1 на 90^0 проти годинниковою стрілки.

14. Дано натуральні числа n та m, що визначають відповідно кількість рядків та стовпців у деякій заданій матриці дійсних чисел M_1. Отримати матрицю M_2 шляхом обертання матриці M_1 на 180^0.

15. Дано натуральне число n, що визначає кількість рядків та стовпців у деякій генерованій квадратній матриці цілих чисел M_1. Отримати матрицю M_2, рядками якою є головна та побічна діагоналі матриці M_1.

16. Дано натуральне число n, що визначає кількість рядків та стовпців у деякій заданій квадратній матриці дійсних чисел M_1. Отримати матрицю M_2, стовбцями якою є побічна та головна діагоналі матриці M_1.

17. Дано натуральне число n, що визначає кількість рядків та стовпців у деякій заданій квадратній матриці цілих чисел. Переставити місцями головну та побічну діагоналі цієї матриці.

18. Дано натуральні числа n та m, що визначають відповідно кількість рядків та стовпців у деякій заданій матриці цілих чисел, а також генерова-

не ціле число x. Замінити нулем усі елементи цієї матриці, що дорівнюють x (якщо такі є).

19. Дано натуральні числа n та m, що визначають відповідно кількість рядків та стовпців у деякій заданій матриці дійсних чисел M_1, а також генероване натуральне число x. Отримати матрицю M_2 шляхом множення M_1 на скаляр x.

20. Дано натуральні числа n та m, що визначають відповідно кількість рядків та стовпців у деякій генерованій матриці дійсних чисел. Переставити місцями перший і останній рядки цієї матриці.

21. Дано натуральні числа n та m, що визначають відповідно кількість рядків та стовпців у деякій заданій матриці дійсних чисел. Знайти серед членів цієї матриці мінімальний за значенням елемент та вивести рядок значень, в якому він знаходиться.

22. Дано натуральні числа n та m, що визначають відповідно кількість рядків та стовпців у деякій генерованій матриці натуральних чисел. Знайти серед членів цієї матриці максимальний за значенням елемент та вивести стовпець значень, в якому він знаходиться.

23. Дано натуральне число n, що визначає кількість рядків та стовпців у деякій заданій квадратній матриці цілих чисел, а також генероване ціле число x. З'ясувати, чи є серед діагональних елементів цієї матриці число x та визначити його індекси.

24. Дано натуральне число n, що визначає кількість рядків та стовпців у деякій заданій квадратній матриці дійсних чисел, та натуральне число x ($x<n$). Вивести спочатку рядок, а потім стовпець значень з номером x.

25. Дано натуральні числа n та m, що визначають відповідно кількість рядків та стовпців у деякій генерованій матриці дійсних чисел, а також довільне дійсне число x. Замінити числом x усі елементи, за виключенням елементів периметру (елементів першого та останнього стовпців та рядків) цієї матриці.

Контрольні запитання:

1. Який масив вважається багатовимірним?
2. Що таке числова матриця?
3. Який синтаксис оголошення багатовимірного масиву у C/C++?
4. Як зберігаються елементи багатовимірного масиву і який обсяг пам'яті вони займають?
5. Як здійснюється індексація елементів багатовимірного масиву?
6. Яким чином можна ініціалізувати багатовимірний масив у C/C++?
7. Яким чином можуть опрацьовуватися багатовимірні масиви у програмах C/C++?
8. Як здійснюється передача і опрацювання багатовимірних масивів у функціях?
9. Наведіть приклади практичного застосування багатовимірних масивів.

13. Практична робота **№11.**
Реалізація алгоритмів сортування масивів

***Мета роботи*:** удосконалювання навичок роботи з одновимірними числовими масивами; набуття навичок розв'язку задач із застосуванням прямих (елементарних) методів сортування масивів у програмах C/C++.

***Теоретичні основи*.**

При роботі з масивами найбільш розповсюдженими є задачі сортування (упорядкування).

Під час розв'язку задачі сортування зазвичай висувається вимога мінімального використовування додаткової пам'яті, що не припускає застосування додаткових (допоміжних) масивів. Упорядкування масиву – це зміна порядку розташування його елементів за певним критерієм. Наприклад, в числовому масиві це може бути упорядкування елементів за зростанням значень або за їх спаданням, для символьного масиву це розташування елементів за абеткою тощо.

Для оцінки швидкодії різних алгоритмів сортування, використовують, як правило, два показники – кількість присвоєнь та кількість порівнянь.

Всі методи сортування можна поділити на дві великі групи – прямі (елементарні) та поліпшені (удосконалені) методи.

Прямі методи сортування по принципу, який лежить в основі методу, поділяються на три підгрупи:
– сортування вставкою (включенням);
– сортування вибором (виділенням);
– сортування обміном («бульбашкове» сортування).

Поліпшені методи сортування ґрунтуються на тих самих принципах, що і прямі, але використовують деякі оригінальні математичні ідеї для прискорення процесу сортування. Серед удосконалених методів сортування найчастіше використовуються методи Хоара, Шелла, пірамідальне сортування (за допомогою дерева), метод злиття тощо.

Прямі методи на практиці використовуються дуже рідко, бо мають відносно низьку швидкодію. Однак вони вдало ілюструють суть заснованих на них поліпшених методів. Окрім того, у деяких випадках (зазвичай при невеликій довжині масиву і (або) особливому початковому розташуванні його елементів) деякі із прямих методів можуть навіть перевершувати поліпшені методи.

Під час сортування вставкою масив умовно поділяється на дві частини – відсортовану і не відсортовану. Елементи із не відсортованої частини по черзі вибираються та вставляються у відсортовану частину так, щоб не порушити в ній наявну упорядкованість елементів. На початку роботи алгоритму у якості відсортованої частини масиву приймають тільки один пе-

рший елемент (елемент із нульовим індексом), а у якості не відсортованої частини – усі інші елементи.

Для реалізації наведеного методу можна запропонувати декілька алгоритмів, що будуть відрізнятися способом пошуку позиції вставки. Розглянемо один з них (упорядкування елементів за зростанням значень):

```
…
for(i=1;i<n;i++) /* n – кількість елементів у масиві */
  for(j=0;j<=i–1;j++)
    if (m[i]<m[j]) /* > – для упорядкування за спаданням */
    {
      t=m[i];
      for(k=i;k>=j+1;k--) m[k]=m[k–1];
      m[j]=t;
    }
…
```

На першому етапі алгоритму сортування вибором перший елемент масиву (елемент із нульовим індексом) по черзі порівнюється з наступними і при невиконанні поставленої умови упорядкування міняється з ними місцями. В результаті першого проходу на першому місці масиву буде знаходитися найбільший або найменший елемент (в залежності від умови упорядкування). На наступних етапах до другого, третього і т.д. аж до останнього елементів масиву буде застосовуватися аналогічний алгоритм обробки. При цьому відсортована частина масиву на кожному наступному етапі збільшується на один елемент та із обробки виключається.

Розглянутий алгоритм може мати наступний вигляд (упорядкування елементів за зростанням значень):

```
…
for(i=0;i<n–1;i++) /* n – кількість елементів у масиві */
  for(j=i+1;j<n;j++)
    if(m[j]<m[i]) /* > – для упорядкування за спаданням */
    {
      t=m[i];
      m[i]=m[j];
      m[j]=t;
    }
…
```

Під час сортування обміном зліва направо по черзі порівнюємо два сусідніх елемента, і якщо їх розташування не відповідає заданій умові упорядкування, то вони обмінюються місцями. Після першого проходу на

останній позиції масиву буде стояти найбільший або найменший елемент (в залежності від умови упорядкування), тобто «спливла» перша «бульбашка». Кожний наступний прохід обмінів виконується аналогічно, але тільки до вже упорядкованої частини масиву.

Розглянемо алгоритм сортування обміном (упорядкування елементів за зростанням значень):

```
…
for(i=0;i<n–1;i++) /* n – кількість елементів у масиві */
  for(j=0;j<n–i–1;j++)
    if (m[j+1]<m[j]) /* > – для упорядкування за спаданням */
    {
      t=m[j];
      m[j]=m[j+1];
      m[j+1]=t;
    }
…
```

У кожному конкретному випадку, в залежності від умов поставленої задачі, програміст сам вирішує питання щодо застосування того чи іншого алгоритму сортування масивів.

Приклад завдання 11.1:

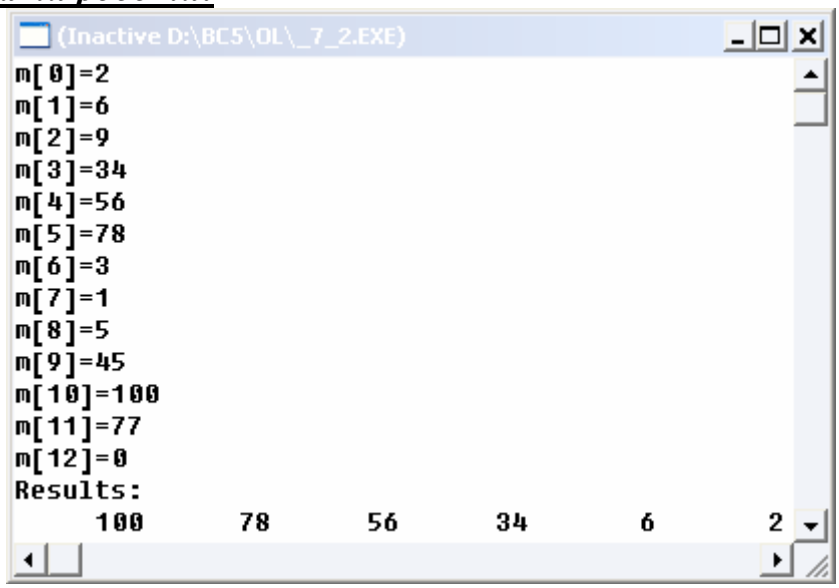
Після введення з клавіатури одновимірного масиву цілих чисел вивести на екран лише парні його елементи, відсортовані за спаданням. Введення припинити введенням числа 0.

Розв'язок:

```
/* 11.1 */
#include <stdio.h>
#define MAX 100
main()
{
  unsigned int i,j,k=0,n=0;
  int m[MAX],t;
  do
  {
    printf("m[%i]=",n);
    scanf("%i",&m[n]);
  }
  while((m[n++])&&(n!=MAX));
  n=(m[n–1])?n:--n; /* n – кількість елементів масиву */
  /* sortuvannya vyborom */
```

```
  for(i=0;i<n–1;i++)
    for(j=i+1;j<n;j++)
      if(m[j]>m[i])
      {
        t=m[i];
        m[i]=m[j];
        m[j]=t;
      }
  printf("Results:\n");
  for(i=0;i<n;i++)
    if(!(m[i]%2))
    {
      k++;
      printf("%8i",m[i]);
    }
  if(!k)printf("No results!");
  return 0;
}
```

Результати роботи:



Завдання

(в усіх задачах введення кожного елементу масиву завершується натисканням Enter, введення масиву припиняється введенням числа 0):

1. Вивести упорядковані за спаданням повторювані елементи довільного одновимірного масиву цілих чисел та число цих повторень.
2. Упорядкувати за зростанням довільний одновимірний масив цілих чисел. Під час виведення масиву однакові елементи не повторювати.
3. У довільному одновимірному масиві цілих чисел видалити повтори елементів. Масив упорядкувати.

4. Для двох однакових за розміром довільних одновимірних масивів дійсних чисел з'ясувати, який з них потребує більше дій під час упорядкування за спаданням. Вивести назву застосованого методу сортування та значення критерію оцінювання його швидкодії.
5. Дано два довільні одновимірні масиви дійсних чисел M_1 і M_2. Сформувати злиттям цих масивів упорядкований масив M_3.
6. Дано довільний одновимірний масив цілих чисел. Продемонструвати усі етапи його упорядкування за зростанням та назву застосованого методу сортування.
7. Дано довільний одновимірний масив цілих чисел. Визначити який прямий метод сортування потребує найменше дій під час упорядкування за зростанням та значення критерію оцінювання його швидкодії.
8. Дано довільний одновимірний масив цілих чисел. З'ясувати під час якого з упорядкувань (за зростанням чи за спаданням) виконується більше дій. Вивести назву застосованого методу сортування та значення критерію оцінювання його швидкодії.
9. З довільного одновимірного масиву дійсних чисел сформувати масив, в якому першу половину елементів упорядковано за спаданням, а другу – за зростанням.
10. З довільного одновимірного масиву дійсних чисел сформувати упорядкований масив, в якому видалені максимальний та мінімальній за значеннями елементи.
11. З довільного одновимірного масиву дійсних чисел сформувати упорядкований масив, в якому від'ємні числа піднесено у квадрат.
12. Дано довільний одновимірний масив цілих чисел. З'ясувати, чи є він упорядкованим. Відповідь сформулювати: «Yes >>» – якщо масив є упорядкованим за зростанням, «Yes <<» – якщо масив є упорядкованим за спаданням або «No».
13. Упорядкувати за зростанням довільний одновимірний масиві цілих чисел M та потім вивести найдовшу в ньому послідовність чисел, що задовольняє умові $M[i+1]=M[i]+1$.
14. Упорядкувати за зростанням довільний одновимірний масиві цілих чисел M та потім вивести останню в ньому послідовність чисел, що задовольняє умові $M[i+1]=M[i]+1$.
15. Упорядкувати за спаданням довільний одновимірний масиві цілих чисел M та потім вивести найпершу в ньому послідовність чисел, що задовольняє умові $M[i+1]=M[i]-1$.
16. З довільного одновимірного масиву дійсних чисел M_1 сформувати однаково упорядковані масиви: M_2, який складається з додатних, і M_3, який складається з від'ємних елементів M_1.
17. З довільного одновимірного масиву дійсних чисел M_1 сформувати масиви: M_2, який складається з парних, і M_3, який складається з непарних елементів M_1. Масив M_2 відсортувати за зростанням, а масив M_3 за спа-

данням.

18. З довільного одновимірного масиву дійсних чисел M_1 сформувати упорядкований за зростанням масив M_2, який містить елементи масиву M_1 з парними індексами.

19. Дано довільний одновимірний масив цілих чисел M_1 та натуральне число n. З масиву M_1 сформувати упорядкований за спаданням масив M_2, який містить елементи M_1, що при цілочисловому діленні на n є парними числами.

20. З довільного одновимірного масиву дійсних чисел M_1 сформувати упорядкований за спаданням масив M_2, який містить елементи M_1, ціла частина яких є непарними числами.

21. З довільного одновимірного масиву дійсних чисел сформувати масив, який містить упорядковані за зростанням цілі частини елементів початкового масиву.

22. З довільного одновимірного масиву дійсних чисел M сформувати масив, який містить упорядковані за зростанням елементи, що утворюються за формулою $M[i]=M[i]-s$, де s – середнє арифметичне додатних елементів M.

23. З довільного одновимірного масиву дійсних чисел M сформувати масив, який містить упорядковані за спаданням елементи, що утворюються за формулою $M[i]=\left| M[i]- s \right|$, де s – максимальний елемент M.

24. З довільного одновимірного масиву дійсних чисел M_1 сформувати масив M_2, який містить упорядковані за спаданням округлені парні елементи масиву M_1.

25. Дано довільний одновимірний масив дійсних чисел M_1 та генероване випадкове дійсне число n (n=0…1). З масиву M_1 сформувати упорядкований за зростанням масив M_2, який містить елементи M_1, дробова частина яких більша за n.

Контрольні запитання:

1. Що таке сортування масиву?
2. Які відомі методи сортування масивів і як оцінюється їхня швидкодія?
3. Навести алгоритм одного з прямих методів сортування масиву.
4. Як поліпшити швидкодію методу сортування масиву?
5. На яких засадах ґрунтуються поліпшені методи сортування масивів?
6. Як може виглядати функція, яка здійснює сортування масиву?

14.Практична робота **№12.**
Робота з символьними рядками

***Мета роботи*:** подальше ознайомлення зі структурами даних, зокрема з символьними рядками; набуття навичок роботи з рядками та застосування рядкових функцій у програмах C/C++.

***Теоретичні основи*.**

Символьний рядок – особливий вид масиву, елементами якого є значення ASCII-кодів символів. Останнім символом рядка повинен бути так званий нуль-символом (ASCII 0) – ознака кінця рядка.

Оголосити символьний рядок можна двома способами – як масив типу char або як рядковий вказівник на тип char, наприклад:

```
char s[128];
char *sPtr;
```

У першому випадку оголошується символьний масив s, що може зберігати від 1 до 128 елементів типу char з урахуванням останнього нуль-символу. Розмір пам'яті, що займає рядок s, суворо зафіксовано (у даному разі це 128 байт), у той час, як поточна довжина такого рядка може змінюватися в зазначених межах під час виконання програми.

Рядковий вказівник (див. друге оголошення) вказує на масив значень типу char, тобто є адресою місцезнаходження першого символу рядка.

Глобальні рядкові змінні оголошуються за межами усіх функцій програми, доступні із усіх її функцій та не потребують ініціалізації. Неініціалізовані локальні рядки доступні лише у своїх функціях та зберігають непередбачувані значення.

Символьні рядки можна ініціалізувати безпосередньо під час оголошення, наприклад:

```
char str1[80]= ″Borland C++″;
char str2[10]= ″″; /* літеральний нульовий рядок */
char str3[]=″My string″;
char str4[]={′s′,′t′,′r′,′i′,′n′,′g′,0};
char *sPtr=″string pointer″;
const char *str=″Літеральний рядок, довжиною у\
декілька рядків екрану″;
```

Якщо граничну кількість символів рядка не вказано або рядок оголошено як рядковий вказівник (наприклад, str3, str4 або sPtr, str), то розмір такого рядка встановлюється компілятором автоматично за кількістю елементів-ініціалізаторів з урахуванням нуль-символу.

Нульовий рядок містить лише нуль-символ, а довжина такого рядка дорівнює нулю.

Доступ до окремих символів рядка здійснюється через індексні вирази, тобто так само, як і до елементів масиву, наприклад:

```
str1[9]='\0'; /* str1="Borland C" */
str3[3]='S'; /* str3="My String" */
sPtr[7]='p'; /* sPtr="string pointer" */
```

З рядковими вказівниками можна працювати так само, як і зі звичайними вказівниками. Виходячи з цього, наступний оператор буде виконувати ті ж самі дії, що й останній оператор попереднього прикладу:

```
*(sPtr+7)='p'; /* вказівник на сьомий символ рядка sPtr */
```

Необхідно зауважити, що у C/C++ оператор присвоєння можна застосовувати лише до рядкових вказівників, наприклад:

```
sPtr="new string pointer";
```

Для присвоєння значень рядкам, оголошеним як символьні масиви, необхідно застосовувати спеціальні бібліотечні функції копіювання рядків (див. табл. 14.1).

Для введення символьного рядка застосовуються функції **scanf()** та **gets()**, прототипи яких знаходяться у заголовному файлі stdio.h, наприклад:

```
scanf("%s",s1);
gets(s2);
```

За допомогою функції scanf() можна ввести рядок до першого введеного пробілу, символу табуляції або натискання клавіші Enter. Функція gets() дозволяє ввести рядок, що містить будь-які розділові символи, проте не захищає користувача від введення зайвої кількості символів, ніж обумовлено під час оголошення рядка.

Для виведення символьного рядка застосовуються функції **printf()** та **puts()**, прототипи яких знаходяться у заголовному файлі stdio.h, наприклад:

```
printf("%s",s1);
puts(s2);
```

У табл. 14.1 розташовано за абеткою стандартні функції мови, призначені для опрацювання символьних рядків.

Таблиця 14.1 – Стандартні функції опрацювання символьних рядків

Функція	Призначення	#include
atof()	– перетворення рядка у дійсне число типу double	stdlib.h
atoi()	– перетворення рядка у ціле число типу int	stdlib.h
atol()	– перетворення рядка у довге ціле число типу long	stdlib.h
itoa()	– перетворення цілого числа у рядок	stdlib.h
ltoa()	– перетворення довгого цілого числа у рядок	stdlib.h
ultoa()	– перетворення беззнакового довгого цілого числа у рядок	stdlib.h
stpcpy()	– копіювання рядка в рядок	string.h
strcat()	– додавання рядка до рядка (конкатенація)	string.h
strchr()	– пошук заданого символу, починаючи з початку рядка	string.h
strcmp()	– порівняння двох рядків	string.h
strcmpi()	– порівняння двох рядків без урахування регістру	string.h
strcpy()	– копіювання рядка в рядок	string.h
strcspn()	– пошук підрядка, що не містить задані символи	string.h
strdup()	– копіювання рядка у динамічну пам’ять	string.h
stricmp()	– порівняння двох рядків без урахування регістру	string.h
strlen()	– визначення довжини рядка	string.h
strlwr()	– перетворення у рядку великих літер у малі	string.h
strncat()	– додавання частини рядка до рядка (конкатенація)	string.h
strncmp()	– порівняння частини двох рядків	string.h
strncmpi()	– порівняння частини двох рядків без урахування регістру	string.h
strncpy()	– копіювання частини рядка в рядок	string.h
strnicmp()	– порівняння частини двох рядків без урахування регістру	string.h
strnset()	– заміна частини символів рядка заданим символом	string.h
strpbrk()	– пошук у рядку першого будь-якого символу іншого рядка	string.h
strrchr()	– пошук заданого символу, починаючи з кінця рядка	string.h
strrev()	– змінення порядку розташування символів на зворотній	string.h
strset()	– заміна усіх символів рядка заданим символом	string.h
strspn()	– пошук підрядка, що містить задані символи	string.h
strstr()	– пошук заданого підрядка	string.h
strtod()	– перетворення рядка у дійсне число типу double	stdlib.h
strtok()	– розкладання рядка на окремі підрядки	string.h
strtol()	– перетворення рядка у довге ціле число типу long	stdlib.h
strtoul()	– перетворення рядка у беззнакове довге ціле число	stdlib.h
strupr()	– перетворення у рядку малих літер у великі	string.h

У разі використання динамічної пам’яті (купи) для роботи з рядковими вказівниками необхідно скористатися спеціальними функціями **malloc()** або **calloc() –** для виділення та **free() –** для вивільнення пам’яті, з прототипами у заголовному файлі stdlib.h (див. приклад завдання 12.1).

В програмах C++ можна застосовувати також альтернативні оператори керування пам’яттю, а саме **new** – для виділення та **delete** – для вивільнення пам’яті у купі.

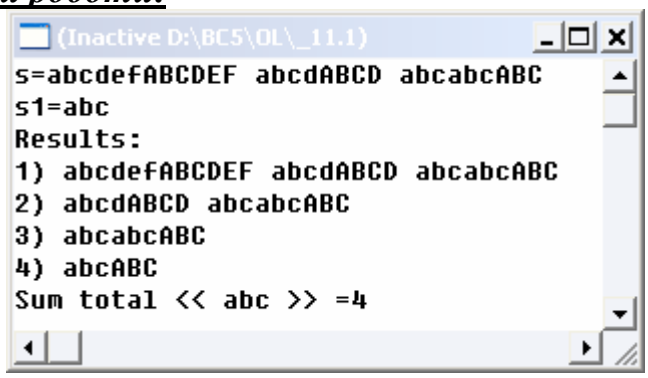
Приклад завдання 12.1:

Дано довільний рядок та деякий підрядок. Визначити, якщо такі є, усі входження цього підрядка в рядок та їхню кількість.

Розв’язок:

```
/* 12.1 */
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <stdlib.h>
#define MAXSTR 128
main()
{
  const char enter='\r';
  int i=0;
  char s[MAXSTR],s1[MAXSTR],*p;
  printf("s=");
  do s[i]=getche(); while(s[i++]!=enter);
  s[i–1]='\0';
  i=0;
  printf("\n");
  printf("s1=");
  gets(s1);
  puts("Results:");
  p=(char *)malloc(MAXSTR);
  while(p=strstr(s,s1))
  {
    printf("%i) ",++i);
    puts(p);
    strcpy(s,p+strlen(s1));
  }
  free(p);
  printf("Sum total << %s >> =%i",s1,i);
  return 0;
}
```

Результати роботи:



Завдання:

1. Дано рядок та деякий символ. Підкреслити у рядку всі входження вказаного символу, якщо такі там є.
2. Дано рядок та підрядок. Підкреслити у рядку всі входження вказаного підрядка, якщо такі там є.
3. Дано рядок та підрядок. Видалити всі входження підрядка в рядку, якщо такі там є, підрахувавши кількість таких видалень.
4. Дано рядок та підрядок. Приєднати цей підрядок до початку та кінця початкового рядка.
5. Дано рядок та деякий символ. Виокремити у рядку всі входження вказаного символу з обох сторін пробілами, якщо такі там є.
6. Дано рядок та підрядок. Виокремити у рядку всі входження вказаного підрядка з обох сторін пробілами, якщо такі там є.
7. Дано рядок та підрядок. Вставити вказаний підрядок у середину початкового рядка, виокремивши його з обох сторін двома пробілами.
8. Дано рядок та деякий символ. Вставити вказаний символ у початковий рядок через кожні два символи.
9. Дано рядок. Розділити всі символи вказаного рядка пробілами.
10. Дано рядок та підрядок. Вставити вказаний підрядок у початковий рядок через 2, 4, 8 … символів.
11. Дано рядок, підрядок та натуральне число *n*. Вставити вказаний підрядок у початковий рядок через кожні *n* символів.
12. Дано рядок та натуральне число *n*. Сформувати рядок, який би містив усі підрядки, що починаються з 2, 4, 8-го … символів початкового рядка та мають довжину *n* символів.
13. Дано рядок та два підрядки *s*1 і *s*2. Замінити у рядку всі входження підрядка *s*1, якщо такі там є, підрядком *s*2, підрахувавши кількість таких замін.
14. Дано рядок. Сформувати рядок, який би містив лише цифрові символи початкового рядка, якщо такі там є, та знайти кількість і суму цих символів.
15. Дано рядок. Сформувати рядок, який би містив відокремлені пробілами номери входжень цифрових символів початкового рядка, якщо такі там

є, та знайти суму цих номерів.

16. Дано рядок. Сформувати рядок, який би містив відокремлені пробілами значення ASCII-кодів символів початкового рядка, та знайти добуток цих кодів.

17. Дано рядок у вигляді послідовності відокремлених комами натуральних чисел 32…255. Сформувати рядок, вважаючи числа початкового рядка відповідними ASCII-кодами символів результуючого рядка.

18. Дано декілька рядків. Розташувати ці рядки за зростанням їх довжини та додати до кінця кожного з них, відокремивши символом «–», їх довжину.

19. Дано декілька рядків. Розташувати ці рядки за спаданням суми ASCII-кодів їх символів та додати до кінця кожного з них, відокремивши символом «–», цю суму.

20. Дано декілька рядків. Об'єднати вказані рядки та відсортувати в результуючому рядку символи за зростанням їх ASCII-кодів.

21. Дано декілька рядків. Об'єднати вказані рядки за принципом, що менший рядок приєднується до кінця більшого.

22. Дано декілька рядків довжиною не більше як розмір текстового вікна. Вирівняти наведений текст за правим краєм більшого з рядків.

23. Дано декілька рядків довжиною не більше як розмір текстового вікна. Додати до кінця кожного з наведених рядків необхідну кількість символів «*» аби вирівняти наведений текст за правим краєм більшого з рядків.

24. Дано декілька рядків та деякий підрядок. Вивести той з наведених рядків, якщо такий там є, який містить якомога більше входжень шуканого підрядка.

25. Дано рядок, котрий закінчується деяким числом, та натуральне число *n*. З'ясувати, чи дорівнює те число *n*.

Контрольні запитання:

1. Що таке символьний рядок?
2. Які типи рядкових змінних використовуються у C/C++?
3. Що таке глобальні та локальні рядкові змінні?
4. Як оголошуються та ініціалізуються рядкові змінні різних типів?
5. Що таке загальна та поточна довжина рядка?
6. Як здійснюється доступ до окремих символів рядка?
7. Як присвоюються значення рядковим змінним?
8. За допомогою яких функцій вводяться та виводяться рядки?
9. Яке призначення має заголовний файл string.h?
10. Які основні стандартні функції застосовуються для обробки рядків?
11. Яким чином вказівники використовуються для обробки рядків?
12. Як використовується динамічна пам'ять для зберігання і обробки символьних рядків?

15.Практична робота **№13.**
Реалізація файлового обміну даними

***Мета роботи*:** подальше ознайомлення зі структурами даних, зокрема з файлами; набуття навичок обробки текстових та двійкових файлів у програмах C/C++.

***Теоретичні основи*.**

Файл у мові C/C++ – абстрактне поняття, яким позначається сукупність даних на зовнішньому носії інформації або певний термінальний пристрій (наприклад, клавіатура, дисплей, принтер тощо). У першому випадку здійснюється прямий (блочний), а у другому – послідовний (побайтовий) файловий обмін.

Високорівневий обмін даними у С-програмах є уніфікованим завдяки системі буферизації. Послідовність байтів, що надходить від певного файлу (логічного пристрою) або передається до файлу, називається потоком. Проміжна область оперативної пам’яті, через яку відбувається обмін потоками між даними програми і файлами, називається буфером.

Для здійснення обміну даними між програмою та файлом необхідно насамперед створити відповідний потік у програмі, пов'язаний безпосередньо з цим файлом, тобто відкрити файл для доступу (читання або запису):

```
FILE *f;
f=fopen(fileName,fileMode);
```

де fopen() – стандартна (stdio.h) функція відкриття існуючого (або створення нового) фізичного файлу на ім’я fileName з заданим режимом обміну fileMode та його зв’язування з файловою змінною f.

Параметр fileName задає ім’я файлу, записане за правилами відповідної операційної системи, наприклад:

```
char fName1[]="student.txt";
char *fName2="D:\\MYFILES\\pr1.cpp";
```

Параметр fileMode задає один з наступних режимів відкриття файлу:

Режим	Двійковий	Текстовий	Опис
"r"	"rb"	"rt"	– відкриття файлу для читання
"w"	"wb"	"wt"	– відкриття або створення файлу для запису
"a"	"ab"	"at"	– відкриття файлу для доповнення даними

Прим. Режим без додаткового символу b або t є текстовим за замовчанням.

Для режиму "r" передбачається, що файл, призначений для відкриття, вже існує. Режим "w" дозволяє створити новий або перезаписати старий

файл с таким самим ім'ям.

До ключової літери кожного з режимів можна додати знак «+», а саме: ″r+″, ″w+″, ″a+″ або ″rb+″, ″wb+″, ″ab+″ тощо, після чого відкриті файли можна буде використовувати як для читання, так і для запису.

Функція fopen() у разі успішного відкриття потоку повертає адресу (вказівник) створеної структури FILE, або повертає порожній вказівник NULL, якщо структуру створити неможливо (невірне ім'я файлу, хибний шлях тощо), наприклад:

```
…
char fName[]=″file.txt″;
FILE *f;
if((f=fopen(fName, ″r″))==NULL)
{
  printf(″Error: File %s not found″,fName);
  exit(1); /* stdlib.h */
}
… /* опрацювання відкритого файлу */
fclose(f);
…
```

Після нормального завершення програми всі відкриті потоки автоматично закриваються, а їх буфери вивільнюються, проте коректніше буде застосування для цього стандартної функції **fclose()** (див. останній приклад). Після успішного виконання ця функція повертає нуль, а у разі помилки – макроконстанту **EOF** (End Of File – кінець файлу).

Читання і запис текстових файлів. Існують два фундаментальні засоби обробки текстових файлів: посимвольний і рядковий.

Основними функціями посимвольного зчитування з файлових потоків, відкритих для читання у текстовому режимі (″rt″), є стандартні функції **getc()** і **fgetc()**. У разі успішного виконання ці функції повертають зчитаний з потоку символ або EOF (у разі досягнення кінця файлу або неможливості зчитування даних).

Для посимвольного запису у файлові потоки, відкриті для запису у текстовому режимі (″wt″), використовуються стандартні функції **putc()** і **fputc()**.

Наприклад, наступний оператор відображає на екрані вміст довільного текстового файлу, зв'язаного у програмі з файловою змінною f1, та посимвольно копіює цей файл в інший, зв'язаний зі змінною f2:

```
while((c=fgetc(f1))!=EOF)
{
  putchar(c);
```

```
  fputc(c,f2);
}
```

Оскільки текстові файли організовано у вигляді символьних рядків, то для підвищення швидкодії алгоритмів роботи з ними більш придатним буде їх рядкове опрацювання. Для цього застосовуються стандартні функції **fgets()** (для читання) і **fputs()** (для запису) рядків у файли, відкриті для опрацювання у відповідному текстовому режимі.

Функція fgets() потребує у якості параметрів: рядкову змінну (буфер); кількість символів (включаючи завершальний нуль-символ), що необхідно прочитати у цей буфер; змінну файлового потоку, відкритого для читання у текстовому режимі. Після успішного виконання ця функція повертає вказівник на зчитаний з файлу рядок або NULL – у разі помилки.

Функція fputs() потребує лише два параметри – рядкову змінну і змінну файлового потоку, відкритого для запису у текстовому режимі. Після успішного виконання ця функція повертає ціле додатне значення або EOF – у разі помилки.

Наступний фрагмент демонструє рядкове копіювання вмісту одного текстового файлу в інший:

```
#define SIZE 255
…
FILE *f1,*f2;
char buffer[SIZE];
…
while(fgets(buffer,SIZE,f1)!=NULL) fputs(buffer,f2);
…
```

Необхідно зауважити, що рядки, розмір яких у вхідному файлі буде більшим за SIZE, «урізатимутся».

Форматне введення/виведення даних по відношенню до текстових файлів здійснюється за допомогою функцій **fscanf()** і **fprintf()** відповідно. Ці функції працюють аналогічно консольним функціям scanf() і printf(), наприклад:

```
fscaf(fin,″%i″,&x);
fprintf(fout,″%5.2f\n″,y);
```

але у якості першого параметру записується відповідний текстовий потік.

Читання і запис двійкових файлів здійснюється за допомогою стандартних функцій блочного обміну даними **fread()** і **fwrite()** відповідно.

Функція fread() потребує наступні чотири параметри: адресу буфера (змінної або масиву змінних), що приймає дані з файлового потоку; розмір

у байтах одного елементу даних, що зчитуються з файлу; кількість таких елементів; змінну файлового потоку, відкритого для читання у двійковому режимі (″rb″). Функція повертає кількість реально зчитаних елементів або нуль (у разі помилки або коли досягнуто кінця файлу).

Функція fwrite() потребує наступні чотири параметри: адресу буфера (змінної або масиву змінних), що передає дані до файлового потоку; розмір у байтах одного елементу даних, що записуються у файл; кількість таких елементів; змінну файлового потоку, відкритого для запису у двійковому режимі (″wb″). Функція повертає кількість реально записаних елементів.

Наступний фрагмент програми демонструє блочне читання даних (″rb″) з файлу f1 довільного розміру у буфер-масив, побайтове опрацювання елементів цього масиву і подальший їх блочний запис (″wb″) у інший файл f2:

```
…
while(res=fread(&buffer,1,sizeof(buffer),f1))
{
  for(i=0;i<res–1;i++) … /* опрацювання масиву buffer */
  fwrite(&buffer,1,res,f2);
}
…
```

Для розв’язку деяких задач обміну даними виникає необхідність у застосуванні функцій позиціювання потоків:

fseek() – встановлює покажчик поточної позиції файлу;
rewind() – встановлює покажчик поточної позиції файлу на його початок;
ftell() – повертає значення покажчика поточної позиції файлу;
fgetpos() – зберігає значення покажчика поточної позиції файлу;
fsetpos() – відновлює значення покажчика поточної позиції файлу, збереженого раніше fgetpos().

Більшість файлових функцій повертають результат, за яким можна перевірити коректність виконаної файлової операції. Проте встановити причину можливого збою можна за допомогою функцій аналізу помилок файлового обміну:

feof() – перевіряє досягнення кінця файлу;
ferror() – перевіряє ознаку помилки в операціях звертання до потоків;
perror() – виводить на екран інформацію про причини виникнення помилки в операціях звертання до потоків, наприклад:

```
if(ferror(f1)) perror(″Warning″);
```

Існує також багато інших файлових функцій. Наприклад, бібліотека dir.h містить спеціальні функції для роботи з файлами і каталогами.

Приклад завдання 13.1:

Після зчитування із заздалегідь підготовленого текстового файлу одновимірного масиву цілих чисел вивести його вміст на екран, а також забезпечити виведення на екран та збереження у текстовому та двійковому файлах цих даних, відсортованих за спаданням.

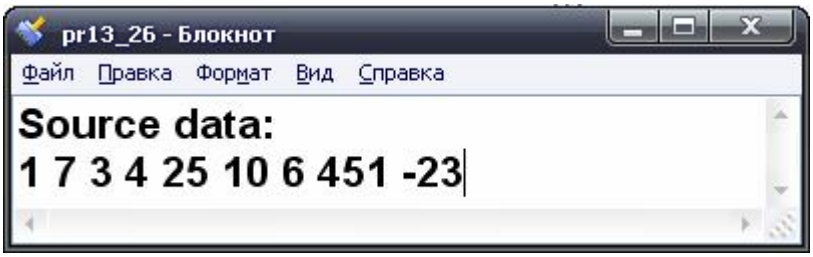


Рисунок 15.1 – Вміст вхідного текстового файлу до завдання 13.1

Розв'язок:

```
/* 13.1 */
#include <stdio.h>
#include <stdlib.h>
#define MAX 100
main()
{
  int i,j,n=0,m[MAX],t;
  FILE *f1,*f2;
  char fName1[]="pr13_26.txt",fName2[]="pr13_26.dta",s[MAX];
  if((f1=fopen(fName1,"r+"))==NULL)
  {
    printf("Error: File %s not found",fName1);
    exit(1);
  }
  f2=fopen(fName2,"wb");
  fgets(s,MAX,f1);
  printf("%s",s);
  while(fscanf(f1,"%i",&m[n])!=EOF)printf("%5i",m[n++]);
  /* sortuvannya vyborom */
  for(i=0;i<n–1;i++)
    for(j=i+1;j<n;j++)
      if(m[j]>m[i])
      {
        t=m[i];
        m[i]=m[j];
        m[j]=t;
      }
```

```
  printf("\nResult data:\n");
  fprintf(f1,"\nResult data:\n");
  for(i=0;i<n;i++)
  {
    printf("%5i",m[i]);
    fprintf(f1,"%5i",m[i]);
  }
  fwrite(&m,sizeof(m[0]),n,f2);
  fclose(f1);
  fclose(f2);
  return 0;
}
```

Результати роботи:

Завдання:

Виконати завдання до практичної роботи за №11, здійснивши зчитування вхідних даних із заздалегідь підготовленого текстового файлу. Забезпечити виведення вхідних та вихідних даних на екран, а також збереження вихідних даних у текстовому та двійковому файлах (див. приклад завдання 13.1).

Контрольні запитання:

1. Що таке файл у мові програмування?
2. Що таке файловий потік і як він створюється засобами C/C++?
3. Які основні режими відкриття файлового потоку у програмі C/C++?
4. Як застосовується та працює у програмі C/C++ функція fopen()?
5. Як закривається файловий потік у програмі C/C++?

6. Як засобами C/C++ здійснюються посимвольне та рядкове опрацювання текстових файлів?
7. Як засобами C/C++ здійснюється форматний обмін даними між програмою і файлами?
8. Як засобами C/C++ здійснюється опрацювання двійкових файлів?
9. Що таке позиціювання файлових потоків?
10. Які є засоби аналізу помилок файлового обміну у програмі C/C++?

Перелік посилань

1. TIOBE Programming Community Index for February 2013.– Режим доступа: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html
2. Яшина О.В., Жулковський О.О. Обчислювальна техніка та програмування: Навч. посібник.– Дніпродзержинськ: ДДТУ, 2007.– 309с.
3. Павловская Т.А. С/С++. Программирование на языке высокого уровня.– СПб.: Питер, 2003.– 461с.
4. Павловская Т.А., Щупак Ю.А. С/С++. Структурное программирование: Практикум.– СПб.: Питер, 2003.– 240с.
5. Березин Б.И., Березин С.Б. Начальный курс С и С++.– М.: «ДИАЛОГ–МИФИ», 2007.– 288с.
6. Керниган Б., Ритчи Д. Язык программирования Си / Пер. с англ.– СПб.: «Невский Диалект», 2001.– 352с.
7. Костюкова Н.И., Калинина Н.А. Язык Си и особенности работы с ним 2006.– 208с.
8. Прата С. Язык программирования С: Лекции и упражнения.– К.: «ДиаСофт», 2000.– 432с.
9. Болски М.И. Язык программирования Си: Справочник.– М.: «Радио и связь», 1988.– 96с.
10. Сван Т. Освоение Borland C++ 4.5: Практический курс.– К.: «Диалектика», 1996.– 543с.
11. Сван Т. Освоение Borland C++ 4.5: Энциклопедия функций.– К.: «Диалектика», 1996.– 320с.
12. Хэзфилд Р., Кирби Л. Искусство программирования на С.– К.: «ДиаСофт», 2001.– 736с.
13. Шмидский Я.К. Программирование на С/С++: Самоучитель.– К.: «Диалектика», 2004.– 352с.
14. Подбельский В.В., Фомин С.С. Программирование на языке Си: Учеб. пособие.– М.: Финансы и статистика, 2004.– 600с.
15. Шпак З.Я. Програмування мовою С.– Львів: Оріяна-Нова, 2006.– 432с.
16. Абрамов С.А., Гнездилова Г.Г., Капустина Е.Н., Селюн Н.И. Задачи по программированию.– М.: Наука, 1988.– 280с.

Додаток **А**

Порядок виконання та захисту практичних робіт

1. Практичні роботи виконується студентами на практичних заняттях з дисципліни в обчислювальному залі з використанням ПК та в часи самостійної роботи.
2. Для студентів денної форми навчання виконання всіх практичних робіт даного навчального видання є обов'язковим. Студенти заочної форми навчання виконують лише роботи, передбачені робочою програмою навчальної дисципліни.
   Перелік і кількість запропонованих до розв'язку задач визначається викладачем, який веде практичні заняття, відповідно до робочої програми дисципліни.
3. Перед виконанням роботи студент зобов'язаний опрацювати та стисло законспектувати теоретичні основи, відповідні розділи рекомендованої літератури (список додається) і конспекту лекцій, а також ознайомитися з прикладами розв'язку типових задач за темою практичного заняття.
4. Підготовлена практична робота подається викладачу на перевірку у вигляді звіту, обов'язковими розділами якого є: номер та тема роботи, стислий конспект теоретичних основ за темою роботи, умова задачі (або задач), розв'язок задачі у вигляді блок-схеми (за необхідністю), роздрукованого на принтері тексту програми та протоколу її реалізації.
5. Звіт бажано оформлювати на папері формату А4 (297х210мм) з однієї сторони. Якщо звіт виконується у зошиті у рукописному вигляді, то тексти програм та протоколи їх реалізації мають бути роздруковані на принтері та вклеєні у зошит.
6. Необхідно також подати до перевірки рекомендований викладачем носій інформації з текстами налагоджених програм та їх виконавчими файлами. Імена файлів мають відповідати номерам практичних робіт і завдань до них (наприклад: pr12_v10.cpp та pr12_v10.exe, тобто практична робота №12, завдання №10).
7. Робота, в якій виявлено суттєві помилки або недоробки, повертається студенту для доопрацювання та усунення письмово зазначених викладачем зауважень. Доопрацьована або перероблена робота подається для повторної перевірки без видалення раніше зазначених зауважень.
8. Захист роботи здійснюється під час практичних занять і консультацій. За результатами захисту за кожну роботу виставляється оцінка за спеціальною шкалою оцінювання, наведеною у робочій програмі з дисципліни.
9. Бали, отримані за окремі роботи, формують загальну суму балів за практикум, яка вказується на титульному аркуші звіту за підписом викладача та враховується у підсумкову оцінку за модуль та семестр.

Додаток **Б**

Порядок виконання та захисту домашніх контрольних робіт (для студентів-заочників)

1. Домашня контрольна робота (ДКР) виконується студентами самостійно на основі засвоєного теоретичного матеріалу та набуття практичних навичок з навчальної дисципліни.
2. Формування індивідуальних завдань до ДКР здійснюється відповідальним за дисципліну викладачем на базі даного навчального видання. Завдання складаються з теоретичних запитань та практичних задач.
3. Завдання до ДКР студент отримує особисто від викладача під час установчої сесії або на кафедрі (у разі пропущення сесії). Консультування з приводу виконання ДКР здійснюється у міжсесійний період у відповідності до графіку консультацій.
   Приклади розв'язку типових задач, а також перелік допоміжної літератури до виконання ДКР містяться у даному навчальному виданні.
4. Виконана за своїм варіантом ДКР оформлюється у вигляді звіту та подається на кафедру для перевірки разом з електронним носієм, де зберігаються виконані на комп'ютері задачі.
   Звіт бажано оформлювати на папері формату А4 (297х210мм) з однієї сторони. Якщо звіт виконується у зошиті у рукописному вигляді, то тексти програм та протоколи їх реалізації мають бути роздруковані на принтері та вклеєні у зошит.
   Рекомендований викладачем електронний носій інформації повинен містити тексти налагоджених програм та їх виконавчі файли. Імена файлів мають відповідати номерам варіантів завдань та задач ДКР (наприклад: v10_z1.cpp та v10_z1.exe, тобто варіант №10, задача №1).
5. Робота, в якій виявлено суттєві помилки або недоробки, повертається студенту для доопрацювання та усунення письмово зазначених викладачем зауважень. Доопрацьована або перероблена робота подається для повторної перевірки без видалення раніше зазначених зауважень.
6. Зарахування правильно виконаної ДКР здійснюється шляхом її захисту за комп'ютером. За результатом захисту ДКР студент отримує допуск до семестрового контролю.

НАВЧАЛЬНЕ ВИДАННЯ

Методичні вказівки до практичних занять з дисципліни «Обчислювальна техніка та програмування» (Частина 2) для студентів, що навчаються за напрямами: 6.050701 – «Електротехніка та електротехнології»; 6.050702 – «Електромеханіка»

Укладачі:
Жулковський Олег Олександрович,
Жулковська Інна Іванівна

Підписано до друку *2014 р.*
Формат *А4* Обсяг *6,0* др. арк.
Наклад *25* прим. Замовлення ___.
51918, м. Дніпродзержинськ,
вул. Дніпробудівська, 2.