ФАКУЛЬТЕТ ІНОФОРМАЦІЙНО-КЕРУЮЧИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Кафедра обчислювальної техніки та систем управління

ОСНОВИ ПРОГРАМУВАННЯ МОВОЮ С++. ІНТЕГРОВАНЕ СЕРЕДОВИЩЕ VISUAL C++

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з дисциплін

«ПРОГРАМУВАННЯ», «АЛГОРИТМІЗАЦІЯ ТА ПРОГРАМУВАННЯ», «ІНФОРМАТИКА»

Частина 2

Харків – 2018

Методичні вказівки розглянуто та рекомендовано до друку на засіданні кафедри обчислювальної техніки та систем управління 26 лютого 2018 р., протокол № 7.

Методичні вказівки призначено для студентів факультету ІКСТ університету денної та заочної форм навчання.

Укладачі:

доценти С. Є. Бантюков,
В. М. Бутенко,
О. В. Головко,
С. О. Бантюкова,
старш. викл. О. В. Чаленко

Рецензент

проф. С. В. Лістровий

ОСНОВИ ПРОГРАМУВАННЯ МОВОЮ С++. ІНТЕГРОВАНЕ СЕРЕДОВИЩЕ VISUAL C++

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт
з дисциплін
«ПРОГРАМУВАННЯ», «АЛГОРИТМІЗАЦІЯ ТА ПРОГРАМУВАННЯ», «ІНФОРМАТИКА»

Частина 2

Відповідальний за випуск Головко О. В.

Редактор Решетилова В. В.

Підписано до друку 05.04.18 р.

Формат паперу 60х84 1/16. Папір писальний.

Умовн.-друк.арк. 2,0. Тираж 50. Замовлення №

Видавець та виготовлювач Український державний університет залізничного транспорту,
61050, Харків-50, майдан Фейєрбаха, 7.
Свідоцтво суб'єкта видавничої справи ДК № 6100 від 21.03.2018 р.

1 ВИКОРИСТАННЯ СТРУКТУР В ПРОГРАМАХ МОВОЮ СІ

1 Мета роботи – набування навичок у визначенні і використанні структур даних.

2 Завдання та порядок виконання

2.1 Вивчити теоретичний матеріал.

2.2 Підготувати відповіді на контрольні запитання.

2.3 Розробити програми відповідно до одержаного завдання.

2.4 Ввести розроблені програми у ПЕОМ, відладити їх, виконати і отримати результати.

3 Контрольні запитання

3.1 Що розуміється під структурою в мові програмування Сі?

3.2 Загальна форма визначення структури.

3.3 Напишіть приклад визначення структури.

3.4 Як оголошуються змінні типу структури?

3.5 Як здійснюється звернення до окремих елементів структури?

3.6 Як оголошуються масиви структур?

3.7 Як здійснюється доступ до конкретної структури в масиві структур?

4 Зміст звіту

4.1 Номер роботи, її назва, визначення мети.

4.2 Стислий зміст основних теоретичних положень і відповіді на контрольні запитання.

4.3 Результати виконання завдання: схеми алгоритмів, їх стислий опис, тексти програм (переписаний або роздрукований), результати роботи програм (роздруковані або з екрану продемонстровані/переписані).

4.4 Висновки з роботи.

5 Навчальний матеріал

5.1 Поняття структури

В мові програмування Сі **структура** є сукупністю змінних різних типів, на яку посилаються за допомогою одного імені.

Структура забезпечує зручні засоби сумісного зберігання зв'язаної інформації. Визначення структури утворює маску, яку можна використовувати для створення змінних структури. Змінні, що складають структуру, називаються **елементами структури.**

Загальна форма визначення структури має такий вигляд:

```
struct <ім'я_типу_структури> {
                <ім'я_змінної1>;
                <ім'я_змінної2>;
                <ім'я_змінної3>;
                        . . .
                < ім'я_змінноїN>;
                                } [<змінні_структури>];
```

Звичайно всі елементи в структурі логічно зв'язані один з одним. Наприклад, в структурі можна подати інформацію про ім'я і адресу. Наведений нижче фрагмент коду оголошує маску структури, що визначає поля імені та адреси. Ключове слово *struct* повідомляє компілятору, що визначається маска структури.

```
struct addr  {
        char name [ 30 ];
        char street [ 40 ];
        char    city [ 20 ];
        char    state [ 3 ];
        int  zip;
            };
```

На даному етапі в цьому коді ніякі змінні в дійсності не оголошені. Він тільки визначає вигляд даних. Для оголошення **фактичної змінної** цієї структури необхідно записати

```
struct addr addr_info;
```

Цей рядок оголосить змінну структури типу *addr*, що називається *addr_info*. Допускається оголошувати одну чи більше змінних.

5.2 Звернення до елементів структури

До окремих елементів структури звертаються за допомогою **оператора . (крапка)**. Загальний формат має такий вигляд:

<ім'я_структури>. <ім'я_елемента>

Наприклад, наведений нижче код буде присвоювати поштовий індекс 12345 полю *zip* структурної змінної *addr_info*, що була оголошена раніше.

```
addr_info. zip = 12345;
```

Таким чином, для виведення на екран поштового індексу можна записати

```
printf ( "%d ", addr_info. zip );
```

У такому ж вигляді можна використати масив символів *addr_info.name* у виклику функції *scanf( )*, як це показано нижче

```
scanf ( "%s ", addr_info. name );
```

Це звернення до функції *scanf()* буде передавати символьний покажчик на початок елемента *name*.

5.3 Масиви структур

Загальним використанням структур є їх застосування в масивах структур. Для оголошення **масиву структур** необхідно спочатку визначити структуру, а після цього оголосити змінну масиву цього типу. Наприклад, для оголошення 100-елементного масиву структур *addr*, що був визначений раніше, можна написати

```
struct  addr  addr_info [ 100 ];
```

Цей рядок створює 100 наборів змінних, що організуються так, як це визначено в структурі *addr*.

Для доступу до конкретної структури індексується ім'я структури. Наприклад, щоб надрукувати поштовий індекс з структури 3, можна записати

```
printf ( "%d ", addr_info [ 2 ]. zip );
```

Подібно до всіх змінних типу масиву масиви структур починають індексування своїх елементів з нуля.

5.4 Приклад використання структур

Розробити програму, що використовує масив структур для зберігання і виведення інформації у вигляді відомості про товари: назву товару (15 позицій), назву магазину (10 позицій), кількість (чотири позиції), ціна одиниці (дев'ять позицій), загальна ціна (11 позицій). Також вивести на екран:

a) назви товарів, загальна ціна яких більше 100 грн;

b) загальну вартість всіх товарів;

c) назву товару з максимальною ціною за одиницю;

d) назву магазину, до якого відноситься найбільша кількість товарів.

Розв'язання.

Нам потрібно спроектувати структуру, що відображає товар. Створимо таблицю 1, в якій відобразимо властивості структури.

Таблиця 1 – Поля та структури

	Поля структури				
Сутність	назва товару	назва магазину	кількість	ціна одиниці	загальна ціна
Ім'я поля	tov[15]	mag[10]	k	cena	symm
Тип поля	char	char	int	float	float
Формат для виведення на друк	%15.15s	%10.10s	%4d	%9.2f	%11.2f

Поля, що відповідають за назву товару і магазину – символьні масиви, всі інші – окрема змінна. Назва структури TOVAR. До цього типу оголошуємо змінну *a* та масив *m[15]*, в який, власне, і розміщуємо вхідні дані.

Змінна *dl* – відображає довжину таблиці при даному запуску програми і має бути не більшою ніж 15 – розмір масиву структур *m[15]*. Значення вводиться з клавіатури.

```
// Програма, що використовує масив структур даних,
#include <conio.h>
#include <stdio.h>
```

```
#include <stdlib.h>
#include <string.h>
struct TOVAR {
char tov[15], mag[10];
int k;
float cena, symm;}
a, m[15];

int main (void)
{float s,b,z, max, max1, S;
 int i,maxi,j,v,k,dl;
 printf("Dlina tabl dl=");
 scanf("%d",&dl);
                        //vvod dat
 for (i=0; i<dl; i++)
   { printf("   Tov %d", i);
   scanf("%s",m[i].tov);
   printf("   Mag %d", i);
   scanf("%s",m[i].mag);
   printf("   Kol %d", i);
   scanf("%d", &m[i].k);
   printf("   Cena %d", i);
   scanf("%f", &s ); m[i].cena=s;
   m[i].symm= m[i].k*s;
   }
  clrscr();
   printf
("_____\n");
   printf ("|              Tov|      Mag|Kol|    Cena| Symm|\n");
   printf
("_____\n");
   for (i=0;i<dl; i++)
     {    printf("%15.15s|", m[i].tov);
   printf("%10.10s|%4d|%9.2f|%11.2f|\n", m[i].mag, m[i].k,
m[i].cena, m[i].symm);
     }
   printf
("_____\n");
```

```
                    //Pechat Tovar  doroge 100 grn
                    // Poisk obshei symm S
  printf("  Tovar  doroge 100 grn\n") ;
  S=0;
  for(i=0;i<dl;i++)
  {     S=S+m[i].symm;
       if (100<m[i].symm)     printf("     %15s\n",m[i].tov) ;
  }
  printf("  obshei symm S=%f\n",S);
                    //poisk tovara s max cena
  max=m[0].cena;          maxi=0;
  for(i=0;i<dl;i++)
    if (max<m[i].cena)
         {max=m[i].cena;        maxi=i;}
  printf("    Dorogoi %15s\n",m[maxi].tov) ;
                   //poisk magazina s max kolichectvom tovarov
   max1=1;      s=1;
   for (j=0;j<dl;j++)
   { max=0;
         for (i=j;i<dl;i++)
           {    if(strcmp(m[j].mag,m[i].mag)==0)
              {max=max+1;
                    }}
          if(max>max1)
            { max1=max;              s=j;          }
         }
    printf("    Magazin %10s", m[s].mag) ;
 getch();
 }
```

6 Варіанти індивідуальних завдань

Розробити алгоритм і програму, що використовує масив структур для зберігання і виведення інформації у вигляді відомості про дані відповідно наступній умові.

1-й рівень

а) результати легкоатлетичного кросу: прізвище (20 позицій); рік народження (чотири позиції); факультет (три позиції); результат (три позиції);

б) вартість будівельних матеріалів: назва (15 позицій); кількість (п’ять позицій); вартість (п’ять позицій);

в) видачу авансу робітникам цеху № N за M місяць R року: прізвище (20 позицій); табельний номер (шість позицій); сума (чотири позиції). Значення N, M, R вводяться в діалозі;

г) облік пропускання занять студентами групи G за M місяць R року: прізвище (20 позицій); дата пропускання (вісім позицій); кількість годин (три позиції). Значення G, M, R вводяться в діалозі;

д) план-графік заміни рейок на дільниці в M місяці R року: перегін (15 позицій); тип рейок (три позиції); довжина заміни (чотири позиції); дата початку робіт (вісім позицій); дата кінця робіт (вісім позицій). Значення M, R вводяться в діалозі.

2-й рівень

а) результати поточної успішності студентів: прізвище (20 позицій); контрольна точка 1 (одна позиція), контрольна точка 2 (одна позиція), контрольна точка 3 (одна позиція). Зробити перевірку на правильність введення оцінки;

б) результати складання іспиту з дисципліни N студентами G групи D дати: прізвище (20 позицій); номер залікової книжки (10 позицій); екзаменаційна оцінка (одна позиція). Значення N, G, D вводяться в діалозі. Зробити перевірку на правильність введення оцінки;

в) розрахунок перевезення вантажів по V відділенню, Z залізниці, на M місяць, R року: назва маршруту (30 позицій); вага вантажу, т (чотири позиції); дальність перевезення, км (чотири позиції); обсяг, ткм (шість позицій). Врахувати, що «обсяг» = «вага вантажу» ∗ «дальність перевезення». Значення V, Z, M, R вводяться в діалозі;

г) вартість матеріалів для ремонту: назва (15 позицій); кількість (три позиції); ціна за од. (чотири позиції); сума (шість позицій). Врахувати, що «сума» = «кількість» ∗ «ціна за од.». Визначити загальну вартість всіх матеріалів;

д) результати складання сесії студентами G групи: прізвище (20 позицій); оцінка з математики (одна позиція); оцінка з фізики (одна позиція); оцінка з обчислювальної техніки (одна позиція). Визначити середню оцінку з кожної дисципліни і дисципліну, що була складена найкраще. Значення G вводиться в діалозі.

3-й рівень

а) список студентів групи: прізвище (15 позицій); ім’я (10 позицій); по батькові (15 позицій); рік народження (чотири позиції). Вивести на екран інформацію про студентів за заданим ключем – роком народження;

б) студентів G групи K курсу F факультету: прізвище (15 позицій); ім’я (10 позицій); по батькові (15 позицій); стать (одна позиція); рік народження (чотири позиції). Зробити перевірку на правильність запровадження статі. Визначити, скільки відсотків у групі складають чоловіки і жінки. Значення G, K, F запроваджуються в діалозі;

в) розрахунок навчального навантаження студентів K курсу F факультету у S семестрі NR навчального року, тривалість семестру N тижнів: назва дисципліни (20 позицій); лекції, год (три позиції); практичні заняття, год (три позиції); лабораторні роботи, год (три позиції); всього з дисципліни, год (три позиції). Врахувати, що «всього з дисципліни» знаходиться як сума «лекції» + «практичні заняття» + «лабораторні роботи». Знайти тижневе завантаження студентів. Значення K, F, S, NR, N вводяться в діалозі;

г) результати легкоатлетичного кросу: прізвище студента (20 позицій); рік народження (чотири позиції); група (10 позицій); результат (три позиції). Визначити студентів з найкращим і найгіршим результатами;

д) список студентів групи: прізвище (15 позицій); ім’я (10 позицій); по батькові (15 позицій); рік народження (чотири позиції). Упорядкувати список студентів за зростанням року народження.

2 ПРОЕКТУВАННЯ ПРОГРАМ ОБРОБКИ МАСИВІВ З ВИКОРИСТАННЯМ ПОКАЖЧИКІВ

1 Мета роботи – вивчення методики проектування програм обробки масивів з використанням покажчиків в пам'яті з довільним доступом.

2 Завдання та порядок виконання

2.1 Вивчити навчальний матеріал і підготувати відповіді на контрольні запитання.

2.2 Виконати завдання 1-4 розділу 5.3

2.3 Розробити схему алгоритму і програму відповідно до свого варіанту завдань першого, другого і третього рівня розділу 6.

3 Контрольні запитання

3.1 Визначте поняття «покажчик».

3.2 За допомогою якого оператора відбувається виділення пам’яті?

3.3 Наведіть приклади ініціалізації покажчиків на тип int; float;

3.4 За допомогою якого оператора відбувається звільнення пам’яті?

3.5 Як відбувається виділення пам’яті під одновимірний масив довільної довжини?

3.6 Як відбувається виділення пам’яті під двовимірний масив довільної довжини?

4 Зміст звіту

4.1 Номер роботи, назва, визначення мети.

4.2 Короткі відповіді на контрольні запитання.

4.3 Схеми алгоритмів пошуку ,вибірки і короткий їх опис.

4.4 Висновки по роботі.

5 Навчальний матеріал

5. Покажчик-змінна

Коли компілятор опрацьовує оператор визначення змінної int q=10; він виділяє пам'ять відповідно до типу int та ініціалізує її значенням 10. Надалі всі звернення до змінної q замінюються компілятором на адресу пам’яті. При проектуванні програми можна визначити власні змінні для зберігання адрес ділянки пам’яті. *Змінні призначені для зберігання адрес ділянки пам’яті, називаються* ***покажчиками.***

В більшості випадків це адреса розміщення в пам'яті іншої змінної. Коли одна змінна містить адресу іншої змінної, кажуть, що перша змінна показує на другу (посередня адресація).

Якщо змінна повинна містити покажчик, то її необхідно оголосити відповідним чином. Загальний формат оголошення покажчик-змінної має вигляд

*<тип> *<ім'я_змінної>;*

де *тип* – будь-який з допустимих в мові Сі базових типів;
ім'я_змінної – ім'я покажчик-змінної.

Базовий тип покажчика визначає, на який тип змінної він може показувати. Наприклад, в наступних операторах оголошуються покажчики на символьну і цілі змінні

```
char  *p;
int  *temp, *p21;
```

В мові С++ розрізняють покажчики на об'єкт, на функцію, і на void, відрізняються властивостями і набором допустимих операцій. Приклад покажчика на об'єкт:

int *b;
float *c;
покажчики b на змінну типу int, с на змінну типу float.

5.2 Оператори з покажчиками

З покажчиками застосовують два оператори: * і &. Оператор & повертає адресу ячейці пам'яті свого операнда. Наприклад,

addr_uk = &uk;

поміщає у змінну *addr_uk* адресу байта пам'яті змінної *uk*. Ця адреса указує на місце розташування змінної в пам'яті комп'ютера. Треба мати на увазі, що адреса змінної не має нічого спільного із значенням змінної. Наприклад, якщо змінна *uk* розміщується за адресою *4000*, то після виконання зазначеного вище оператора змінна *addr_uk* буде мати значення *4000*.

Оператор * повертає значення змінної, що розташована за даною адресою. Наприклад, якщо *addr_uk* містить адресу байта пам'яті змінної *uk*, тоді

*val = *addr_uk;*

розмістить значення змінної *uk* у змінну *val*.

Унарна операція *p – розадресація, операнд p – покажчик. Бере значення за адресою пам’яті, вказаною в покажчику p. Обернена операція &f – взяття адреси змінної f. Якщо p=&f, то здійснювати доступ до пам’яті (значення) f можна як до f, так і як до *p.

Оператори & та * мають більш високий пріоритет, ніж усі інші арифметичні оператори, за винятком унарного мінуса, з яким за пріоритетом вони рівні.

Приклад програми, в якій використовуються два наведених оператори присвоєння для виведення на екран числа 100:

```
#include <stdio.h>
main ( )
{
int  *addr_uk,  uk=100,  val;
addr_uk = &uk;          // присвоєння адреси uk
val  =  *addr_uk;                  // присвоєння значення, що знаходиться за
                        //  цією адресою
printf ( "%d ", val );    //  виведення на екран числа 100
}
```

5.3 Вирази з покажчиками

Як і в разі з будь-якою змінною, покажчик можна записувати в правій частині оператора присвоювання для присвоювання значення покажчику іншої змінної, як показано в такому фрагменті:

```
int x, *p1, *p2;
p1 = &x;
p2 = p1;
printf ( "%p ", p2 );   // друк адреси  x
```

Специфікатор формату *%p* у функції *printf()* визначає, що буде виведено значення покажчика в шістнадцятковому коді.

Над покажчиками можна виконувати тільки дві арифметичні операції: «+» та «–» . Припустимо, що *p1* - покажчик на ціле значення з поточною величиною *4000*. Після виконання виразу на 16-бітних системах *p1++*; вміст *p1* буде *4002*, так як при кожному

прирості покажчик показує на наступне ціле. Те ж саме справедливо і для від'ємного приросту. Наприклад, при виконанні *p1--*; *p1* буде мати значення *3998.*

До покажчиків можна додавати і віднімати від них цілі величини. Так, у виразі

p1 = p1 + 9;

p1 буде показувати на дев'ятий елемент базового типу *p1* по відношенню до поточного.

Ініціалізація покажчиків:

```
int b=5;
float c[5]={3.5, 6, -2, 0.4, 4.6};
int *pb=&b;
float *pc=c;
```

де покажчики *pb* вказують на змінну *b* типу *int*, тобто до значення *b* можна звертатись **pb*. *pc* вказує на масив *c* типу f*loat*, тепер до елементів масиву *c[0]=3.5, c[1]=6, c[2]=-2* можна звертатись **pc, *(pc+1), *(pc+2)* і т. д. Кажуть, що *pc* вказує на нульовий елемент масиву *c*. Значення покажчика можна змінювати. При цьому він буде вказувати на іншу ділянку пам’яті. Якщо

pc++;

то *pc* буде вже вказувати на перший елемент масиву *c, (pc+1)* на другий, *(pc+3)* на третій.

Виділення динамічної пам’яті і присвоєння її адреси в покажчик.

```
int *b;
float *c;
b= new int;              //1
c= new float[7];         //2
```

В операторі 1 new виконує виділення достатньої для розміщення величини типу int ділянки динамічної пам’яті і записує адресу початку цієї ділянки в змінну b. Пам'ять під саму змінну b була виділена на етапі компіляції.

В операторі 2 new виконує виділення пам’яті для розміщення 10 величин типу float в пам’яті і записує адресу початку цієї ділянки в змінну c, яка може використовуватись як ім’я масиву.

Звільнення виділеної пам’яті здійснює оператор delete. Для покажчиків з операторів 1 і 2 звільнення пам’яті матиме вигляд:

delete b; ***delete[] c;***

Значення покажчиків можна друкувати, використовуючи формат %p.

Приклад 1

```
#include <stdio.h>
void main()
{ float c[5]={3.5, 6, -2, 0.4, 4.6};          //3
  float *pc=c;                                //4
 int i;
printf("1   pc= %p \n ",pc);                        //5
for(i=0;i<3;i++)                              //6
printf("  %f  ",*(pc+i));                  //7
printf("\n 2  pc= %p \n ",pc);                      //8
}
```

В прикладі 1 наведена програма, яка друкує значення pc, значення перших трьох елементів масиву c, і знову значення pc.

При цьому значення покажчика pc не змінилось. Він продовжує вказувати на нульовий елемент масиву c.

Завдання 1. Набрати наведену в прикладі 1 програму на комп’ютері і запустити її на виконання. Записати текст програми в зошит. Записати результат в зошит. Змінити на власні дані значення в масиві c (оператор 3). Запустити програму на виконання. Записати результат в зошит і порівняти з попереднім.

Завдання 2. В програмі завдання 1 змінити оператор //7 на

```
printf("  %f  ",*pc+i);                  //7
```

Запустити програму на виконання. Записати результат в зошит і порівняти з попереднім.

Завдання 3. В програмі завдання 2 змінити оператор //7 на

```
printf("  %f  ",*pc++);                  //7
```

Запустити програму на виконання. Записати результат в зошит і порівняти з попереднім. Звернути увагу на значення pc.

Завдання 4. В програмі завдання 3 змінити оператор //7 на

```
printf("  %f  ",*(pc++));                //7
```

Запустити програму на виконання. Записати результат в зошит і порівняти з попереднім. Звернути увагу на значення pc.

5.4 Генератор випадкових чисел

В мові C++ для генерації випадкових чисел використовують функції з бібліотеки stdlib.h. Розглянемо їх детально:

randomize() – ініціалізує генератор випадкових чисел деякою випадковою величиною;

rand() – повертає псевдовипадкове ціле число в діапазоні від 0 до RAND_MAX;

RAND_MAX – значення залежить від бібліотек, але гарантовано буде принаймні 32767 на будь-якій стандартній реалізації бібліотеки;

random(N) повертає випадкове число в інтервалі від 0 до N-1.

Якщо нам потрібно отримати дійсні (дробові) значення з інтервалу [a, b] і виконана умова b>a, то використовуємо формулу

x=(b-a)(float)rand()/32767+a;*.

Якщо нам потрібно отримати цілі значення з інтервалу [c, d] і виконана умова d>c, то використовуємо формулу

y= random((d-c)+1)+c;.

Розглянемо задачу1.

В масиві Q з n цілих елементів знайти максимальний елемент (останній, якщо максимальний елемент не один) і обчислити суму елементів масиву, розташованих після нього. Заповнення масиву реалізувати за допомогою генератора випадкових чисел з даного інтервалу [-20, 20].

Розв’язання

Позначимо змінною ***max*** максимальний елемент, змінною ***imax*** – його індекс, змінною ***sum*** – суму елементів масиву, розташованих після максимального елемента.

Приклад 2

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

void main()
{ int n, *pq, max, imax,sum=0;   //1
 int i;
```

```
printf("\n Input n= ",);          //2 визначення розміру
scanf("%d", &n);                  //  масиву

pq=new int[n];                    //3 виділення пам'яті
                                  //під масив
randomize();
for(i=0;i<n;i++)                  //4 заповнення
{                                 //масиву значеннями
      *(pq+i)=random(41)-20;
      printf("\n q[%d]=%f ", i, *(pq+i));
}                                 //і друк на екран

max=*pq;       imax=0;
for(i=0;i<n;i++)                  //5 знаходимо мах
if(max<=*(pq+i)){
                  max=*(pq+i);
                  imax=i;
            }
printf("\n max=%d imax=%d", max,imax);

if(imax<n-1){                     //чи є елементи за мах
                                  //якщо є, рахуємо суму
      for(i=imax+1;i<n;i++)
      sum=sum+*(pq+i);
      printf("\n sum=%d", sum);
            }                     //інакше друкуємо
                                  //повідомлення
else printf("\n За мах елементів немає");

delete[] pq;
}
```

Розглянемо задачу 2.

Дано масив S з n цілих елементів. З його від'ємних елементів, що мають непарні індекси, сформувати масив Z. Обчислити суму елементів Z(j)<=-5. Заповнення масиву реалізувати за допомогою генератора випадкових чисел з даного інтервалу [-20, 10].

Розв’язання

Позначимо змінною nz – максимально можливу кількість елементів у масиві Z, змінною ***i*** - індекс поточного елемента масиву S, змінною j – індекс поточного елемента масиву Z, змінною ***sum*** – суму елементів Z(j)<=-5.

Приклад 3

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

void main()
{ float *S, *Z, sum=0;        //1
 int i, n, j, nz;

printf("\n Input n= ",);              //2 визначення розміру
scanf("%d", &n);                      //  масиву S
nz=n/2+1;                     //  масиву Z
S=new float[n];               //3 виділення пам’яті
                              //під масив S
Z=new float[nz];              //4 виділення пам’яті
                              //під масив Z

randomize();
for(i=0;i<n;i++)              //5 заповнення
{                             //масиву S значеннями
      *(S+i)= 30*(float)rand()/32767-20;.
      printf("\n S[%d]=%f ", i, *(S+i));
}                             //і друк на екран

j=0;
for(i=1;i<n;i=i+2)            //6  формуємо Z
if(0>*(S+i)){
            *(Z+j)=*(S+i);
            printf("\n Z[%d]=%f ", j, *(Z+j));
            if(*(Z+j)<=-5)  //7 знаходимо sum
                  sum=sum+*(Z+j);
            j++;
             }
```

```
printf(“\n sum=%f”, sum);
delete[] Z;
delete[] S;
}
```

Так як двовимірний масив фактично є одновимірним масивом з одновимірних масивів рядків, тому оголошуємо його як покажчик на покажчик. Виділення пам'яті під двовимірний масив К з цілими елементами, кількість рядків якого ***r***, а кількість стовпців ***s***, має вигляд

```
int r,s;
cout<<"r=";     cin>>r;
cout<<"s=";     cin>>s;
int **K= new int *[r];
for(int i=0; i<r; i++)
        K[i]= new int[s];
```

Розглянемо задачу 3.

Дано масив Т з цілих елементів, кількість рядків якого ***r***, а кількість стовпців ***s***. Для кожного стовпця знайти суму від'ємних елементів, і сформувати з них одновимірний масив Z. Обчислити кількість елементів Z(j)<-2. Заповнення масиву Т реалізувати за допомогою генератора випадкових чисел з даного інтервалу [-10, 10].

Розв’язання

Зазначимо, що кількість елементів у масиві Z дорівнює ***s*** кількості стовпців масиву Т, а також, що Z(j) є сумою від'ємних елементів масиву Т в поточному j -му стовпці. Позначимо змінною k – кількість елементів Z(j)<-2.

Приклад 4

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

void main()
{ int i, j, k, sum,r, s;          //2 визначення розміру Т
cout<<"r=";     cin>>r;           // кількість рядків
cout<<"s=";     cin>>s;           // кількість стовпців
```

```
int **T= new int *[r];                    //3 виділення пам'яті
for( i=0; i<r; i++)                       //під масив T
      T[i]= new int[s];

int *Z;
Z=new int[s];                             //4 виділення пам'яті
                                          //під масив Z
cout<<"\n";
randomize();
for(i=0;i<r;i++)                          //5 заповнення
      {for(j=0;j<s;j++)                   //масиву T значеннями
      {T[i][j]= random(21)-10;.
      cout <<T[i][j]<<"\t";
      }                          //і друк на екран
cout<<"\n";
}
for(j=0;j<s;j++)                          //вибираємо стовпець
      {*(Z+j) =0;                         //обнуляємо сумму
            for(i=0;i<r;i++)              //додаємо в суму
            if(0>T[i][j]))                //значення <0
                  *(Z+j) =*(Z+j) + T[i][j];
      cout<<"\n Z["<<j<<"]="<< *(Z+j));
      if(*(Z+j)<-2)    k=k+1;             //рахуємо кількість k
      }
cout<<"\n k="<<k;

Z=delete[];                               //звільнення пам'яті для Z
for( i=0; i<r; i++)                       //для  масиву T
      T[i]= delete[];
}
```

6 Варіанти для самостійного виконання

Рівень 1

1 В масиві T з n дійсних елементів знайти максимальний елемент (останній, якщо максимальний елемент не один) і обчислити добуток додатних елементів масиву, розташованих після нього. Заповнення масиву реалізувати за допомогою генератора випадкових чисел з даного інтервалу [-10, 20].

2 В масиві S з k дійсних елементів знайти мінімальний елемент (останній, якщо мінімальний елемент не один) і обчислити добуток від'ємних елементів масиву, розташованих після нього. Заповнення масиву реалізувати за допомогою генератора випадкових чисел з інтервалу [-10, 15].

3 В масиві D з m цілих елементів знайти максимальний елемент і обчислити добуток від'ємних елементів масиву, розташованих до нього. Заповнення масиву реалізувати за допомогою генератора випадкових чисел з інтервалу [-20, 10].

4 В масиві H з p цілих елементів знайти мінімальний елемент і обчислити добуток додатних елементів масиву, розташованих до нього. Заповнення масиву реалізувати за допомогою генератора випадкових чисел з інтервалу [-15, 10].

5 В масиві F з x цілих елементів знайти максимальний і мінімальний елемент і обчислити суму елементів масиву, розташованих поміж ним. Заповнення масиву реалізувати за допомогою генератора випадкових чисел з даного інтервалу [-10, 20].

Рівень 2

1 Дано масив D з n цілих елементів. З його від'ємних елементів, що мають парні індекси, сформувати масив X. Обчислити середнє арифметичне елементів X(j). Заповнення масиву реалізувати за допомогою генератора випадкових чисел з даного інтервалу [-15, 15].

2 Дано масив V з m елементів типу float. З його від'ємних елементів сформувати масив Y. Обчислити кількість елементів Y(j)<=-5. Заповнення масиву реалізувати за допомогою генератора випадкових чисел з інтервалу [-12, 10].

3 Дано масив G з p цілих елементів. З його додатних елементів, що мають парні індекси, сформувати масив X. Обчислити добуток елементів X(j). Заповнення масиву реалізувати за допомогою генератора випадкових чисел з даного інтервалу [-15, 10].

4 Дано масив W з q елементів типу float. З його додатних елементів сформувати масив Y. Обчислити добуток елементів Y(j)=5. Заповнення масиву реалізувати за допомогою генератора випадкових чисел з інтервалу [-12, 12].

5 Дано масив F з елементів типу float. З його елементів, що мають парні індекси, сформувати масив W. Обчислити середнє арифметичне елементів W(j). Заповнення масиву реалізувати за допомогою генератора випадкових чисел з даного інтервалу [0, 30].

Рівень 3 (двовимірні масиви)

1 Дано масив D з цілих елементів, кількість рядків якого ***r***, а кількість стовпців ***s***. Для кожного стовпця знайти добуток елементів >3, і сформувати з них одновимірний масив X. Обчислити суму елементів X(j). Заповнення масиву реалізувати за допомогою генератора випадкових чисел з інтервалу [-5, 15].

2 Дано масив V з цілих елементів, кількість рядків якого ***n***, а кількість стовпців ***m***. Для кожного рядка знайти суму елементів і сформувати з них одновимірний масив Y. Обчислити добуток елементів Y(j)<=-5. Заповнення масиву реалізувати за допомогою генератора випадкових чисел з інтервалу [-12, 10].

3 Дано масив G з цілих елементів, кількість рядків і стовпців якого ***s***. Для кожного стовпця знайти кількість від'ємних елементів і сформувати з них одновимірний масив X. Обчислити добуток елементів X(j). Заповнення масиву реалізувати за допомогою генератора випадкових чисел з даного інтервалу [-15, 10].

4 Дано масив W з цілих елементів, кількість рядків і стовпців якого q. Для кожного рядка знайти суму додатних елементів і сформувати з них одновимірний масив Y. Обчислити добуток елементів Y(j)=5. Заповнення масиву реалізувати за допомогою генератора випадкових чисел з даного інтервалу [-12, 10].

5 Дано масив F з цілих елементів, кількість рядків якого ***x***, а кількість стовпців ***s***. Для кожного рядка знайти кількість додатних елементів і сформувати з них одновимірний масив W. Обчислити середнє арифметичне елементів W(j). Заповнення масиву реалізувати за допомогою генератора випадкових чисел з даного інтервалу [0, 30].

3 ПРОЕКТУВАННЯ ПРОГРАМ З ВИКОРИСТАННЯМ ФУНКЦІЙ

1 Мета роботи – вивчення методики проектування програм з використанням покажчиків функцій.

2 Завдання та порядок виконання

2.1 Вивчіть навчальний матеріал і підготуйте відповіді на контрольні запитання.

2.2 Розробіть схему алгоритму і програму відповідно до завдань 1-5 розділу 6.

3 Контрольні запитання

3.1 Визначте поняття «функція».

3.2 Чим об'явлення відрізняється від опису функції?

3.3 Наведіть приклади передачі параметрів в функцію типу int; float;

3.4 За допомогою якого оператора відбувається повернення значення функції в місце її виклику? Де визначається тип значення, що повертає функція?

3.5 Чим відрізняється передача параметра в функцію за значенням від передачі за адресою?

3.6 Чим відрізняється фактичний параметр функції від формального?

4 Зміст звіту

4.1 Номер роботи,назва, визначення мети.

4.2 Короткі відповіді на контрольні запитання.

4.3 Схеми алгоритмів пошуку ,вибірки і короткий їх опис.

4.4 Висновки з роботи.

5 Навчальний матеріал

5.1 Функції. Об'явлення і опис

Кожна програма в мові С складається з функцій, одна з яких має ім'я main() (з неї починається виконання програми) .

Для написання великих програм, як показує досвід, краще користуватися функціями. Програма буде складатися з окремих функцій. Окремих, тому що робота окремої функції не залежить

від роботи якої-небудь іншої. Тобто алгоритм у кожній функції функціонально достатній і не залежатиме від інших алгоритмів програми. Одного разу написавши функцію, її можна буде з легкістю переносити в інші програми. **Функція (в програмуванні)** — це фрагмент коду або алгоритм, реалізований якоюсь мовою програмування з метою виконання певної послідовності операцій для виконання певної завершеної підзадачі. Отже, функції дозволяють зробити програму модульною, тобто розділити програму на кілька маленьких підпрограм (функцій), які в сукупності виконують поставлену задачу. Ще один величезний плюс функцій в тому, що їх можна багаторазово використовувати. Функція починає виконуватись в момент **виклику**. Будь-яка функція має бути **об'явлена** і **описана** (визначена). Об'явлень може бути декілька, а опис лише один.

Крім виклику функцій із стандартних заголовних файлів, в мові програмування С передбачена можливість створення власних функцій. В мові програмування С є два типи функцій:

а) функції, які не повертають значень;

б) функції, які повертають значення.

Функції, що не повертають значення, завершивши свою роботу, жодної відповіді програмі не дають. Розглянемо структуру оголошення таких функцій.

Прототипи функцій. **Об'явлення** функції (прототип, заголовок) задає її ім'я, тип значення, що повертається, і список параметрів, які передаються. Припустимо, що в деякому місці у вашому коді відбувається виклик певної функції. Задамося питанням, за яких умов при компіляції цієї ділянки коду компілятор не видає помилки. На це питання існує проста відповідь. Десь в тому ж файлі, в якому здійснюється виклик функції, перед операцією, до якої здійснюється виклик, повинні бути присутніми прототип або опис функції. Крім того, аргументи і тип значення, що повертається у виклику, повинні відповідати списку аргументів і типу значення, що повертається в прототипі і в описі функції.

```
 // прототип функції
void function(int arg1, double arg2); // в кінці об'явлення завжди
                                      //    ставиться крапка з комою
```

Опис функції (визначення) містить, крім об'явлення, тіло функції, що подано, як послідовність операторів і описів у фігурних дужках.

тип им'я ([список параметрів])

{ тіло функції }

Розглянемо складові визначення:

тип значення, що повертає функція, може бути будь-яким, крім масиву (але може бути покажчиком на масив). Якщо функція не повертає значення, то вказуємо тип void .

Список параметрів визначає величини, які потрібно передати в функцію при її виклику. Елементи списку розділяються комами. Для кожного параметра вказуємо його тип та ім'я (в об'явленні імена можна опустити).

Як видно, в прототипі зазначаються по порядку тип значення, що повертається, (у даному прикладі void), назва функції (в даному випадку function) і список параметрів у дужках. Оголошення прототипу повинно закінчуватися крапкою з комою.

Для чого потрібен прототип функції? Прототип та опис функції використовуються компілятором для того, щоб виклик функції відбувався правильним чином. Для цього компілятор спочатку дивиться ім'я функції, що викликається, і шукає у файлі прототип або опис цієї функції. Якщо знайдений прототип або опис, то перевіряються аргументи, що передаються функції у виклику, і використання значення, що повертається.

Для чого потрібен саме прототип? Чому не можна обмежитися використанням одного опису функції? Прототип став необхідний після того, як стандарти мови змінилися таким чином, що перед викликом функції у файлі необхідно якимось чином її описати. Проблема полягає в тому, що ім'я функції має глобальну область видимості (якщо її опис знаходиться поза всяких локальних областей). Припустимо, що опис функції знаходиться в окремому вихідному файлі. Також припустимо, що необхідно здійснити виклик цієї функції в декількох інших вихідних файлах. Якщо немає прототипу, то кожен такий вихідний файл необхідно включити в повний опис функції. Компілятор буде інтерпретувати це як перевизначення. Якщо ж

ми використовуємо прототип, то ми можемо включати цей прототип в стільки вихідних файлів, скільки нам необхідно.

5.2 Передача параметрів у функцію

Обмін інформацією між функцією, що викликається, і функцією, що викликає, здійснюється за допомогою механізму передачі параметрів. Список змінних, зазначений у заголовку функції, називається формальними параметрами або просто параметрами функції. Список змінних в операторі виклику функції — це фактичні параметри або аргументи.

Передача параметрів виконується наступним чином. Обчислюються вирази, що стоять на місці фактичних параметрів. Потім формальним параметрам присвоюються значення фактичних. Виконується перевірка типів і при необхідності виконується їх перетворення.

Передача параметрів у функцію може здійснюватися за значенням і за адресою.

При передачі даних за значенням функція працює з копіями фактичних параметрів, і доступу до початкових значень аргументів у неї немає. При передачі за адресою в функцію передається не змінна, а її адреса, і, отже, функція має доступ до комірок пам'яті, в яких зберігаються аргументів. Таким чином, дані, передані за значенням, функція змінити не може, на відміну від даних, переданих за адресою.

При передачі даних за адресою функція працює з копіями адрес фактичних параметрів, і має доступ до значень аргументів. Розглянемо приклад з функцією , що має параметри за значенням і за адресою.

```
#include <iostream.h>
void f(int i, int *j);
int main()
{           int i = 1, j = 2;
            cout << "i  j \n"
            cout << i << " " << j <<"\n";
            f( i , j );
            cout << i << " " << j <<"\n";
            return 0;
}
```

```
void f(int i, int *j) {
        i++; (*j)++;
}
```

Результат роботи програми:

i j

1 2

1 3

Перший параметр (i) передається за значенням. Його зміна в функції не впливає на вихідне значення. Другий параметр (j) передається за адресою з допомогою покажчика, при цьому для передачі в функцію адреси фактичного параметра використовується операція взяття адреси.

Якщо параметр функції – масив, то в функцію передається покажчик на його перший елемент. При цьому інформація про кількість елементів масиву втрачається. Тому при передачі багатовимірних масивів у функцію всі розмірності передаються як параметри.

5.3 Значення, що повертається

Механізм повернення значення з функції до функції, що її викликала, реалізується оператором:

return [вираз];

Функція може містити декілька операторів return (що визначається алгоритмом). Якщо функція описана як void, вираз не вказується. Оператор return можна взагалі не вказувати.

Вираз вказаний після return, неявно перетворюється до типу функції, що повертає значення, і передається в місце виклику функції.

Приклади:

```
int f1() { return 1;}       //  правильно
void f2() { return 1;}      //не правильно, f2 не має повертати значення
float f1() { return 1;}     // правильно, 1 перетвориться в тип float
```

Не можна повертати з функції покажчик на локальну змінну, так як пам'ять, виділена під локальну змінну при вході в функцію, звільняється після повернення з неї.

```
int *f1(){
int a;
 return &a;  }       // не можна!!!
```

Розглянемо приклад проектування програми для розв'язання такої задачі:

дано двовимірний масив A 5×3 . В кожному рядку знайти значення максимального елемента і вивести його на екран.

Розв'язання

Визначимо підзадачу, розв'язання якої винесемо в окрему функцію. Це знаходження максимуму з трьох значень. Функція з іменем ***max*** ,а відповідні аргументи - це змінні.

Приклад 1

```
#include <stdio.h>
void main()
{ float c[5]={3.5, 6, -2, 0.4, 4.6};          //3
  float *pc=c;                                //4
 int i;
printf("1   pc= %p \n ",pc);                        //5
for(i=0;i<3;i++)                            //6
printf("  %f  ",*(pc+i));                   //7

printf("\n 2  pc= %p \n ",pc);                      //8

}
```

Напишемо програму на задачу 1 з попередньої лабораторної роботи. Нагадаємо умову.

Розглянемо задачу1.

В масиві Q з n цілих елементів знайти максимальний елемент (останній, якщо максимальний елемент не один) і обчислити суму елементів масиву, розташованих після нього. Заповнення масиву реалізувати за допомогою генератора випадкових чисел з даного інтервалу [-20, 20].

Розв'язання.

Визначимо підзадачі, розв'язання яких винесемо в окремі функції. Це знаходження максимуму і його розташування в масиві. Позначимо цю функцію іменем ***max***, а відповідні

аргументи – це покажчик **mas** – покажчик на масив, **К** – розмір масиву, змінна **ind** – індекс максимального елемента. Тоді її заголовок функції має вигляд

int max(int *mas, int K, int *ind)

Також винесемо в окрему функцію введення даних в масив. Позначимо цю функцію іменем **vved** ,а відповідні аргументи – це покажчик **mas** – покажчик на масив, **К** – розмір масиву:

void vved(int *mas, int K)

І окремо запишемо функцію виведення даних з масиву на екран. Позначимо цю функцію іменем **prin**, а відповідні аргументи – це покажчик **mas** – покажчик на масив, **К** – розмір масиву:

void prin(int *mas, int K)

Приклад 2

```
#include <stdio.h>
#include <stdlib.h>
//-----------------------------------об'явлення функцій
int max(int *, int , int *);
void vved(int *, int );
void prin(int *mas, int K);
//-----------------------------головна функція
void main()
{ int n, *pq, max, imax,sum=0;   //1
 int i;

printf("\n Input n= ",);            //2 визначення розміру
scanf("%d", &n);                    //  масиву

pq=new int[n];                 //3 виділення пам'яті
      //---------------------------виклик функцій
vved(pq,n);                     //   заповнення масиву значеннями
prin(pq,n);               //     друк на екран
      //---------------------------виклик функції    max
printf("\n max=%d ", max(pq,n,&imax));
printf("\n  imax=%d",imax);

if(imax<n-1){                  //чи є елементи за мах
```

```
                                    //якщо є, рахуємо суму
for(i=imax+1;i<n;i++)
        sum=sum+*(pq+i);
        printf("\n sum=%d", sum);
              }                     //інакше друкуємо
                                    //повідомлення
else printf("\n За мах елементів немає");
delete[] pq;
}
//-------------------------------описи функцій
//------------------------------- знаходимо мах
int max(int *mas, int K, int *ind)
{ int m,i;
m=*mas;  *ind=0;
for(i=0;i<K;i++)
if(m<=*(mas+i)){
                    m=*(mas+i);
                    *ind=i;
              }
return m;
}
//------------------------------------ заповнення масиву
void vved(int *mas, int K)
{int i;
randomize();
for(i=0;i<K;i++)
        *(mas+i)=random(41)-20;
}
//------------------------------------- друк на екран

void prin(int *mas, int K)
{int i;
for(i=0;i<K;i++)                    //4 заповнення
        printf("\n q[%d]=%d ", i, *(mas+i));
}
```

В програмі формальним параметрам **mas, K** у функціях **vved(), prin(), max()**, при виклику відповідають фактичні параметри **pq, n**.

6 Варіанти для самостійного виконання

Переписати задачу з попередньої лабораторної роботи, розділивши її на окремі функції. Обов'язково мають бути функції введення даних, обчислення шуканої в задачі величини.

В задачах 2 рівня забезпечити виведення результуючого масиву з використанням окремої функції.

4 ПРОЕКТУВАННЯ ПРОГРАМ З ВИКОРИСТАННЯМ ГРАФІКИ

1 Мета роботи - вивчення методики проектування програм з використанням графіки.

2 Завдання та порядок виконання

2.1 Вивчити навчальний матеріал і підготувати відповіді на контрольні запитання.

2.2 Розробити схему алгоритму і програму відповідно до завдань 1-5 розділу 6.

3 Контрольні запитання

3.1 Що характеризує графічний режим екрана?

3.2 Що характеризує текстовий режим екрана?

3.3 Наведіть функції, що забезпечують увімкнення і вимкнення графічного режиму.

3.4 За допомогою яких функцій відбувається управління кольором?

3.5 Як організована система координат в графічному режимі екрана?

4 Зміст звіту

4.1 Номер роботи, назва, визначення мети.

4.2 Короткі відповіді на контрольні запитання.

4.3 Схеми алгоритмів пошуку, вибірки і короткий їх опис.

4.4 Висновки з роботи.

5 Навчальний матеріал

5.1 Загальні відомості

В мові програмування Сі є графічна бібліотека, яка містить велику кількість функцій, що дозволяють здійснювати різноманітні графічні операції. Настроювання цих функцій на роботу з конкретним відеоадаптером досягається шляхом підключення необхідного графічного драйвера. Адаптер - це електронна схема, що управляє зовнішнім приладом ПЕОМ. Драйвер - це спеціальна програма для управління тим чи іншим приладом комп'ютера.

Графічні драйвери знаходяться в окремих файлах з розширеннями BGI (Borland Graphics Interface).

При необхідності використання графічних функцій у вхідному модулі програми треба помістити директиву *#include <graphics.h>*, що підставляє на своє місце вміст файла заголовка з прототипом функцій.

5.2 Графічний і текстовий режими функціонування екрана

Екран користувача в мові С може перебувати в двох режимах: графічному і текстовому. В текстовому режимі екран розділяється на прямокутники (знакомісця), в кожному з яких може бути розміщений один символ. По вертикалі розміщується 25 рядків. В кожному рядку по 80 знакомісць. Введена система координат. Знакомісце з координатами (0, 0) розташоване в лівому верхньому кутку екрана. Перша координата позначає рядок (положення по вертикалі), а друга координата позначає стовпчик (положення по горизонталі). Цей режим є активним за замовчуванням. Функції, що відповідають за символьну графіку, знаходяться в бібліотеці conio.h.

В графічному режимі екран розділяється на крапки (пікселі). По вертикалі розміщується 480 пікселів. По горизонталі 640. Введена система координат. Піксель з координатами (0, 0) розташований в лівому верхньому кутку екрана. Перша координата позначає стовпчик (положення по горизонталі) і

лежить в межах від 0 до 639, а друга координата позначає рядок (положення по вертикалі) і лежить в межах від 0 до 479.

#include<graphics.h> - відповідає за підключення функцій роботи з екраном в графічному режимі.

Функції, що забезпечують увімкнення і вимкнення графічного режиму.

initgraph( )- функція забезпечує перехід в графічний режим.

closegraph( ) - функція забезпечує вихід з графічного режиму.

5.3 Ініціалізація графічної системи. Перемикання режимів

Виконання кожної програми починається в текстовому режимі. Для переведення системи у графічний режим необхідне виконання функції *initgraph ( )*:

initgraph(&<графічний_драйвер>, &<графічний_режим>, "<шлях_до_bgi_файлів>");

Якщо BGI-драйвери знаходяться в поточній директорії, то замість параметра *"<шлях_до_bgi_файлів>"* можна задати пустий рядок :

initgraph (&gdriver, &gmode, "");

У противному разі шлях до драйвера записується в такому форматі:

".. \\bgi\\drivers "

Існує функція *graphresult()*, яка аналізує причини неправильного виконання графічної операції :

graphresult ( );

Якщо виконання графічної функції завершилося успішно, функція повертає значення 0. У противному разі значення, що повертається, ідентифікує причину невдачі.

Існує можливість надрукувати повідомлення, що відповідає значенню коду помилки. Для цього використовується функція

grapherrormsg (<код_помилки>);

Наприклад,

error = graphresult( );

printf ("Помилка графіки : %s ", grapherrormsg (error));

Перемикання режимів. Для тимчасового переходу у текстовий режим в графічній бібліотеці існує функція

restorecrtmode ( );

що відновлює той текстовий режим, який був перед зверненням до функції *initgraph()*. Повернутися у графічний режим можна за допомогою функції

setgraphmode (<графічний_режим>);

де *графічний_режим* – цілочисельний номер режиму, що допущений для даного драйвера. Попередній графічний режим можна відновити, якщо його номер був своєчасно визначений за допомогою функції

getgraphmode( );

і збережений в програмі.

Одержати максимальне значення номера графічного режиму, що допущений для поточного драйвера, можна за допомогою функції

getmaxmode ( );

Після закінчення роботи з графічною системою її необхідно відключити. Це виконує функція

closegraph ( );

5.4 Робота з вікнами і координатами

Після встановлення графічного режиму весь екран за умовчанням стає графічним вікном. Всередині основного графічного вікна (екрана) можна виділити прямокутне вікно зі своєю системою координат. Виконується це за допомогою функції

setviewport (<x1>, <y1>, <x2>, <y2>, <відсікання>);

де *x1, y1* - координати лівого верхнього кута вікна;

x2, y2 - координати правого нижнього кута вікна.

Якщо параметр *<відсікання>* дорівнює 1, то ті елементи зображення, що не вміщаються у вікні, будуть відсічені, а якщо 0, то межі вікна проігноруються.

Очищення графічного вікна виконує функція

clearviewport ( );

Очищення екрана виконується за допомогою функції

cleardevice ( );

Всі встановлені раніше графічними процедурами параметри скидаються і набувають значення за умовчанням.

Для відновлення параметрів, прийнятих за умовчанням, є функція

graphdefaults ( );

Відновлюються параметри, встановлені функцією *initgraph ( ).*

Координати лівого верхнього кута графічного екрана (0, 0), а координати правого нижнього кута визначаються функціями

getmaxx ( ); і *getmaxy ( );* .

Ці функції повертають максимальну координату по горизонталі і по вертикалі відповідно.

Координати графічного курсора завжди відраховуються щодо лівого верхнього кута вікна. Графічний курсор є невидимим, але його поточні горизонтальні і вертикальні координати можуть бути визначені за допомогою функцій

getx ( ); і *gety ( );* .

Ці координати подаються даними типу *int* і можуть бути як додатними, так і від'ємними.

Переустановлення покажчика позиції виконує функція

moveto (<x>, <y>);

де *x, y* - нові координати в системі координат вікна.

5.5 Установлення кольорів, стилів ліній і зафарбовування, шрифтів

Існує можливість встановлювати окремо основний колір, що рисує, і колір фону. Поточне значення основного кольору (колір символів і ліній, що виводяться) встановлюється функцією

setcolor (<колір>);

Кольором фону можна управляти за допомогою функції

setbkcolor (<колір>);

де колір - один з перелічених кольорів чи код.

КОД	КОЛІР	ЗНАЧЕННЯ
0	BLACK	Чорний
1	BLUE	Синій
2	GREEN	Зелений
3	CYAN	Бірюзовий

КОД	КОЛІР	ЗНАЧЕННЯ
4	RED	Червоний
5	MAGENTA	Малиновий
6	BROWN	Коричневий
7	LIGHTGRAY	світло-сірий
8	DARKGRAY	темно-сірий
9	LIGHTBLUE	світло-синій
10	LIGHTGREEN	світло-зелений
11	LIGHTCYAN	світло-бірюзовий
12	LIGHTRED	світло-червоний
13	LIGHTMAGENTA	світло-маліновий
14	YELLOW	Жовтий
15	WHITE	Білий

Максимальне значення кольору повертає функція

getmaxcolor ( );

Для установлення характеру і товщини ліній геометричних об'єктів використовується функція

setlinestyle (<стиль>, <зразок>, <товщина>);

Коди для параметра *стиль* (тільки для кусково-лінійних графічних примітивів)

Код	**Стиль**
0	Solid_line (суцільна)
1	Dotted_line (з точок)
2	Center_line (з точок і тире)
3	Dashed_line (пунктирна)
4	Userbit_line (що визначається користувачем)

Параметр *зразок* задається тільки тоді, коли *стиль* дорівнює 4 (в інших випадках він ігнорується, тому його можна робити рівним 0).

Допущені значення для параметра *товщина*:

1 Norm_width (лінія в один піксель)
3 Thick_width (лінія в три пікселі)

Існує можливість зафарбувати виділену на екрані замкнуту ділянку певним засобом. Для установлення стилю і кольору зафарбування використовується функція

setfillstyle (<стиль>, <колір>);

Допущені значення параметра *стиль* :

0	Empty_fill	Заповнення пустотою
1	Solid_fill	Заповнення суцільним кольором
2	Line_fill	Заповнення горизонтальною лінією ---
3	Ltslash_fill	Заповнення ///
4	Slash_fill	Заповнення потовщеними лініями ///
5	Bkslash_fill	Заповнення потовщеними лініями \\\
6	Ltbkslash_fill	Заповнення \\\
7	Hatch_fill	Горизонтальні грати
8	Xhatch_fill	Скісні грати
9	Interleave_fill	Коса лінія з прошарками
10	Wide_dot_fill	Рідко розташовані точки
11	Close_dot_fill	Щільно розташовані точки

При виведенні текстового повідомлення існує можливість вибору одного з декількох шрифтів, розміру символів, що виводяться, і напрямку тексту. Ці параметри задаються за допомогою функції

settextstyle (<шрифт>, <напрямок>, <розмір_символів>);

Допущені значення параметра *шрифт* :

0	Default_font (стандартний)
1	Triplex_font (типу тріплекс)
2	Small_font (зменшений)
3	Sans_serif_font (прямий)
4	Gothic_font (готичний)

Допущені значення параметра *напрямок* :

	Horiz_dir (зліва_направо)
	Vert_dir (знизу_вгору)

Існує можливість вирівняти текст за допомогою функції
settextjustify (<горизонтально>, <вертикально>);

Допущені значення параметрів *горизонтально* і *вертикально:*

Параметр	**Назва**	**Значення**	**Вирівнювання**
горизонтально	Left_text	(0)	ліворуч <
	Center_text	(1)	> в центрі <
	Right_text	(2)	> праворуч
вертикально	Bottom_text	(0)	знизу вгору
	Center_text	(1)	в центрі
	Top_text	(2)	зверху до низу

Виведення тексту у вікно можливе за допомогою стандартних функцій *printf ( )* і *puts ( )*. Але вони обмежені виглядом і розміром символів шрифту, а також можливістю розміщення символів тільки в тих позиціях екрана, що допускаються в текстовому режимі. Спеціальні ж графічні функції виведення тексту дозволяють працювати з ним, як з повноправним елементом графіки.

Функцій виведення графічного тексту – дві:

outtext (<рядок>);

виводить рядок символів, починаючи з поточної графічної позиції;

outtextxy (<x>, <y>, <рядок>);

виводить рядок символів, починаючи із позиції *(x, y)*.

5.6 Функції , що рисують графічні примітиви:

circle(x, y, R) – функція рисує коло з центром в точці з координатами (x , y) та радіусом R;

line(x1, y1, x2, y2) – функція рисує лінію з точки з координатами (x1, y1), в точку з координатами (x2, y2);

lineto(x, y) – функція рисує лінію з точки, в якій на даний момент розташований курсор (поточне положення), в точку з координатами (x, y) – нове поточне положення курсора;

rectangle(x1, y1, x2, y2) – функція рисує прямокутник з верхнім лівим кутом в точці з координатами (x1, y1) та нижнім правим кутом в точці з координатами (x2, y2);

bar(x1, y1, x2, y2) – функція рисує зафарбований прямокутник з верхнім лівим кутом в точці з координатами (x1, y1) та нижнім правим кутом в точці з координатами (x2, y2).

Заповнення кольором обмеженої області виконується функцією

floodfill (<x>, <y>, <колір_межі>);

де *x, y* – координати точки, з якої починається заповнення замкнутої області в усіх напрямках до *кольору_межі.*

Встановлення кольору заповнення слід здійснювати раніше за допомогою функції *setfillstyle ( )*, наприклад,

```
setcolor ( 4 );
rectangle ( 10, 10, maxx-20, maxy-20 );
setfillstyle ( Solid_fill, 2 );
floodfill ( 12, 12, 4 );
```

5.7 Функції, що працюють в символьному режимі:

cprintf ( ) – функція виводить різнокольоровий текст на екран в символьному режимі;

gotoxy( ) функція встановлює курсор в певну точку екрана в символьному режимі;

getch(); – функція чекає натискування клавіші і повертає її код;

#include<conio.h> – відповідає за підключення функцій роботи з екраном в символьному режимі;

#include<graphics.h> – відповідає за підключення функцій роботи з екраном в графічному режимі.

Розглянемо на прикладі програми, що рисує прямокутник по центру екрана і коло в правому верхньому кутку, роботу з графікою. В нашій програмі значення координат точок, що задають положення прямокутника, задамо змінними x0, y0 для лівого верхнього кутка, і змінними xn, yn для правого нижнього, присвоївши їм відповідні значення при об'явленні.

Приклад 1

```
#include <conio.h>   //
#include<graphics.h>
void main()
{int x0= 300, y0=220, xn=340, yn=260;     //
```

```
int gd=DETECT, gm;                    //
initgraph(&gd, &gm, "\\tc\\bgi");     //

setbkcolor(1);                        // 5
setcolor(2);                          // 6

rectangle( x0, y0, xn, yn);           //
 circle( 550 , 40 , 20 );             //

getch();                              // 7
closegraph( );                        // 8
}
```

Напишемо програму на задачу 1 з попередньої лабораторної роботи, додавши побудову графіка. Нагадаємо умову.

Розглянемо задачу1.

В масиві Q з n цілих елементів знайти максимальний елемент (останній, якщо максимальний елемент не один) і обчислити суму елементів масиву, розташованих після нього. Заповнення масиву реалізувати за допомогою генератора випадкових чисел з даного інтервалу [-20, 20]. Забезпечити виведення графіка, побудованого на основі масиву. Вісь X – номер елемента масиву, вісь Y – його значеннння.

Розв’язання.

Визначимо підзадачі, розв’язання яких винесемо в окремі функції. Це знаходження максимуму і його розташування в масиві. Позначимо цю функцію іменем ***max*** ,а відповідні аргументи – це покажчик **mas** – покажчик на масив, **K** – розмір масиву, змінна **ind** – індекс максимального елемента. Тоді її заголовок функції має вигляд

int max(int *mas, int K, int *ind)

Також винесемо в окрему функцію введення даних в масив. Позначимо цю функцію іменем **vved** ,а відповідні аргументи – це покажчик **mas** – покажчик на масив, **K** – розмір масиву:

void vved(int *mas, int K)

І окремо запишемо функцію виведення даних з масиву на екран. Позначимо цю функцію іменем **prin**, а відповідні аргументи – це покажчик **mas** – покажчик на масив, **K** – розмір масиву:

void prin(int *mas, int K)

В окремій функції випишемо підключення графіки і відображення масиву в вигляді графіка. Визначимось з положеннями осей координат. Вертикальна вісь Y розташовується на рівні x=120, від y=10 до y= 460. Бажане положення графіка – це горизонтальна смуга від прямої y=40 до прямої y=440. На початку функції визначаємо максимальне mx і мінімальне mn значення елементів масиву, щоб визначитись з масштабом по осі Y.

Приклад 2.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include<graphics.h>
//-----------------------------------об'явлення функцій
int max(int *, int , int *);
int min(int *, int , int *);
void vved(int *, int );
void prin(int *mas, int K);
void graf(int *, int )
//----------------------------головна функція
void main()
{ int n, *pq, max, imax,sum=0;   //1
 int i;
printf("\n Input n= ",);          //2 визначення розміру
scanf("%d", &n);                  //  масиву

pq=new int[n];                    //3 виділення памяті
                                  // виклик функцій
vved(pq,n);                       // заповнення масиву
значеннями
prin(pq,n);                       // друк на екран
                                  //----виклик функції    max
printf("\n max=%d ", max(pq,n,&imax));
printf("\n  imax=%d",imax);
if(imax<n-1){                     // чи є елементи за мах
                                  // якщо є, рахуємо суму
for(i=imax+1;i<n;i++)
```

```
        sum=sum+*(pq+i);
        printf("\n sum=%d", sum);
              }                          // інакше друкуємо
                                         // повідомлення
else printf("\n За мах елементів немає");
getch();
graf(pq,n);
delete [] pq;
}
//-------------------------------описи функцій
//------------------------------- знаходимо мах
int max(int *mas, int K, int *ind)
{ int m,i;
m=*mas;  *ind=0;
for(i=0;i<K;i++)
if(m<=*(mas+i)){
                    m=*(mas+i);
                    *ind=i;
              }
return m;
}
//------------------------------- знаходимо мін
int min(int *mas, int K, int *ind)
{ int m,i;
m=*mas;  *ind=0;
for(i=0;i<K;i++)
if(m>*(mas+i)){
                    m=*(mas+i);
                    *ind=i;
              }
return m;
}
//----------------------------------- заповнення масиву
void vved(int *mas, int K)
{int i;
randomize();
for(i=0;i<K;i++)
        *(mas+i)=random(41)-20;
```

```
}
//------------------------------------ друк на екран
void prin(int *mas, int K)
{int i;
for(i=0;i<K;i++)                    //4 заповнення
      printf("\n q[%d]=%d ", i, *(mas+i));
}
//------------------------------------ виведення графіка на екран
void graf(int *pa, int K)
{int i, x0,y0, xn, yn,in;
 int gd=DETECT, gm;
 int mx =20, mn =-10;
 mx = max(pa, K, &in);
 mn = min(pa, K, &in);
 initgraph(&gd, &gm, "\\tc\\bgi");
 xn=500/K;  x0=120;
if(mx>0 && mn<0) { yn=400/(mx-mn);
              y0=40+mx*yn; }
              else if (mx > 0) { yn=400/(mx-mn);
              y0=40+mx*yn;}
                    else { yn=400/(-mn);
                    y0=40; }
setbkcolor(1);
setcolor(2);
line(x0, 10, x0, 460);   //   Y
line(10, y0, 630, y0);   //   X
setcolor(3);
line(10, y0-10*yn, 630, y0-10*yn);
outtextxy(x0-4, y0-10*yn, "10");
setcolor(14);
moveto( x0, y0-*(pa) *yn);
for(i=1; i<K; i++)
      lineto( x0+ i*xn, y0- *(pa +i)* yn);
print(pa, K);
getch();
closegraph();
}
```

В програмі формальним параметрам **mas, K у** функціях **vved(), prin(), max()**, при виклику відповідають фактичні параметри **pq, n**, а в функції **graf()** формальним параметрам **pa,** і **K** при виклику відповідають фактичні параметри **pq, n.**

6 Варіанти для самостійного виконання

Рівень 1

1 Побудувати програму, яка виводить на екрані в графічному режимі рисунок, де зліва зображений будинок (хоча б з одним вікном), а справа вгорі три кулі одного радіуса - дві на одному рівні, а третя над ними.

2 Побудувати програму, яка виводить на екрані в графічному режимі рисунок, де зліва зображена ялинка (з трьох трикутників), а справа вгорі – дві кулі різного діаметра на одному рівні.

3 Побудувати програму, яка виводить на екрані в графічному режимі рисунок, де справа зображена вантажівка, а зліва вгорі - дві кулі різного діаметра на одному рівні.

4 Побудувати програму, яка виводить на екрані в графічному режимі рисунок, де справа зображений вагон, а зліва вгорі - три кулі - дві на одному рівні, а третя над ними.

5 Побудувати програму, яка виводить на екрані в графічному режимі рисунок, де по центру зображений вагон, а вгорі над ним - три кулі - і на одному рівні.

Рівень 2

До задач, що були в попередньому розділі, додати функцію яка будує графік.

1 В масиві T з n дійсних елементів знайти максимальний елемент (останній, якщо максимальний елемент не один) і обчислити добуток додатних елементів масиву, розташованих після нього. Заповнення масиву реалізувати за допомогою генератора випадкових чисел з даного інтервалу [-10, 20].

2 В масиві S з k дійсних елементів знайти мінімальний елемент (останній, якщо мінімальний елемент не один) і обчислити добуток від’ємних елементів масиву, розташованих після нього. Заповнення масиву реалізувати за допомогою генератора випадкових чисел з інтервалу [-10, 15].

3 В масиві D з m цілих елементів знайти максимальний елемент і обчислити добуток від'ємних елементів масиву, розташованих до нього. Заповнення масиву реалізувати за допомогою генератора випадкових чисел з інтервалу [-20, 10].

4 В масиві H з p цілих елементів знайти мінімальний елемент і обчислити добуток додатних елементів масиву, розташованих до нього. Заповнення масиву реалізувати за допомогою генератора випадкових чисел з інтервалу [-15, 10].

5 В масиві F з x цілих елементів знайти максимальний і мінімальний елемент і обчислити суму елементів масиву, розташованих поміж ним. Заповнення масиву реалізувати за допомогою генератора випадкових чисел з даного інтервалу [-10, 20].

Рівень 3

До задач, що були в рівні 2, додати функцію, яка будує стовпчикову діаграму.

5 ПРОЕКТУВАННЯ ДІАЛОГОВИХ ДОДАТКІВ WINDOWS У СЕРЕДОВИЩІ VC++ 6.0

1 Мета роботи – вивчення методики проектування програм та розробки інтерфейсу користувача з використанням елементів керування Edit Box, List Box.

2 Навчальний матеріал та порядок виконання

2.1 Створення діалогового вікна

Програма в середовищі Windows завжди буде мати велику кількість **діалогових вікон**, які призначені для обміну або виведення інформації.

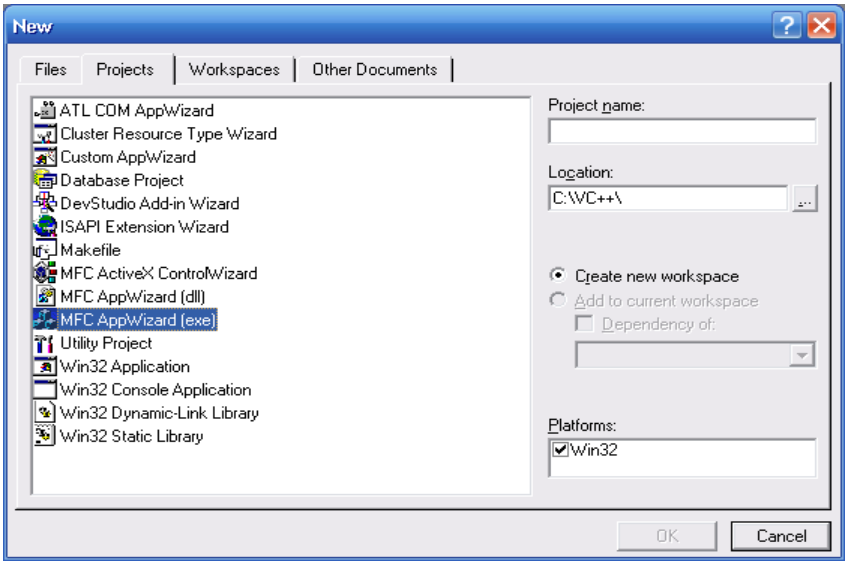
Опис кожного **діалогового вікна** складається з двох частин, які розташовані в двох різних файлах:

- опис ресурсів **діалогового вікна** (зберігається в спеціальному файлі з розширенням .rc);
- опис класу діалогового вікна, який розташований в файлі реалізації (розширення).

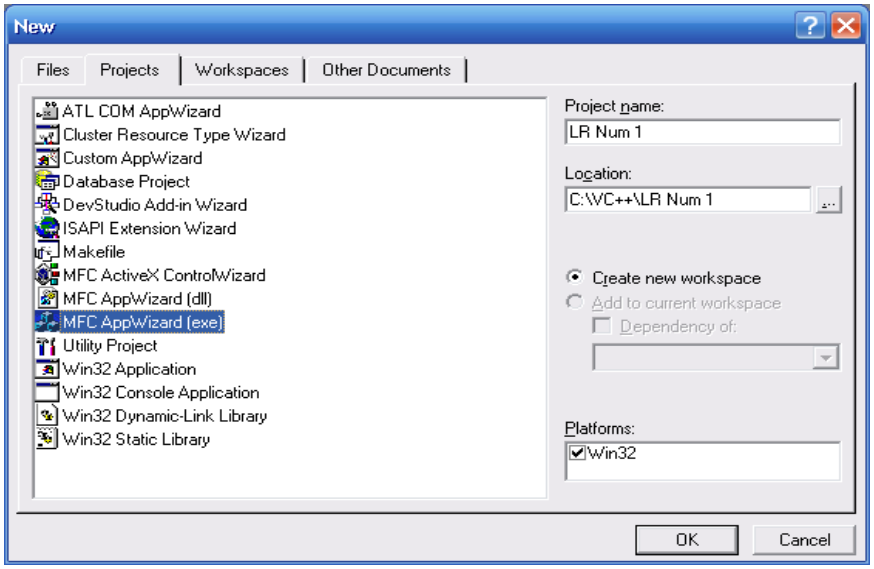
Кожному з цих файлів відповідає свій файл заголовка.

Створіть заготовку діалогового додатка за допомогою майстра AppWizard. Для цього виконайте такі дії:

1 Запустіть середовище VC++. Оберіть команду **File, New.** З'явиться діалогове вікно **New** з активною вкладкою **Projects.**

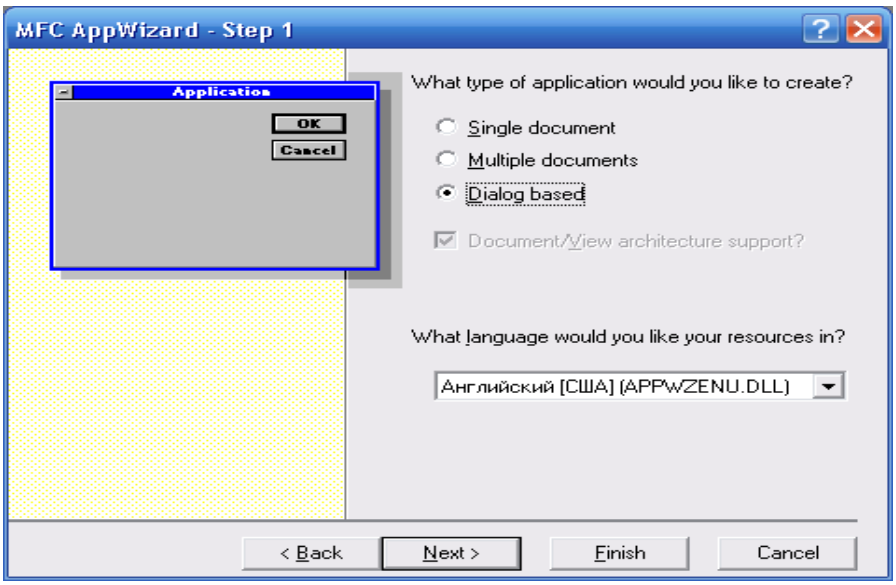


2 Оберіть пункт **MFC AppWizard (exe)** і введіть в текстове вікно **Project name** ім'я проекта **«LR Num 1».**



3 Натисніть кнопку **ОК.** З'явиться діалогове вікно **MFC AppWizard – Step 1** (майстер створення заготовки проекту, який використовує бібліотеку **MFC** – крок 1).

4 Встановіть перемикач **Dialog Based**.



5 Натисніть кнопку Next> (Далі). З'явиться діалогове вікно MFC AppWizard – Step 2 of 4 (другий крок з чотирьох майстра AppWizard);

- до групи **What features would you like to include? (**які властивості матиме вікно, яке створюється?**)** віднесені такі прапорці:

• **About box** (діалогове вікно властивостей) – визначає можливість виклику з діалогового вікна, яке створюється, іншого діалогового вікна, в якому буде розміщена інформація про версію програми та її розробників і яке буде викликатись за замовчуванням;

• **Context-sensitive Help** (контекстна довідка) – визначає необхідність підключення до додатка довідкової системи. За замовчуванням в додатку відсутня довідкова система;

• **3D controls** (об'ємні елементи керування) – визначає зовнішній вигляд елементів керування діалогового вікна. Об'ємні елементи керування використовуються за замовчуванням;

- до групи **What other support would you like to include?** (які додаткові властивості матиме вікно, яке створюється?) увійшли такі прапорці:

• **Automation** (автоматизація) – визначає можливість цього додатка передати керування іншому додатку завдяки механізму **Automation.** За замовчуванням така можливість не передбачена;

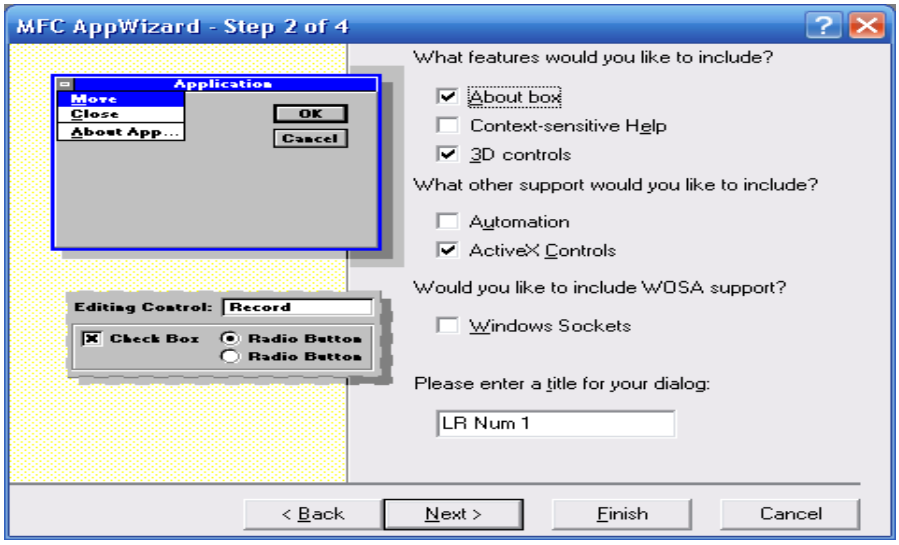
• **ActiveX Controls** (елементи керування ActiveX) – визначає можливість використання у вікні, яке створюється,

елементів керування ActiveX. Ця можливість передбачена за замовчуванням;

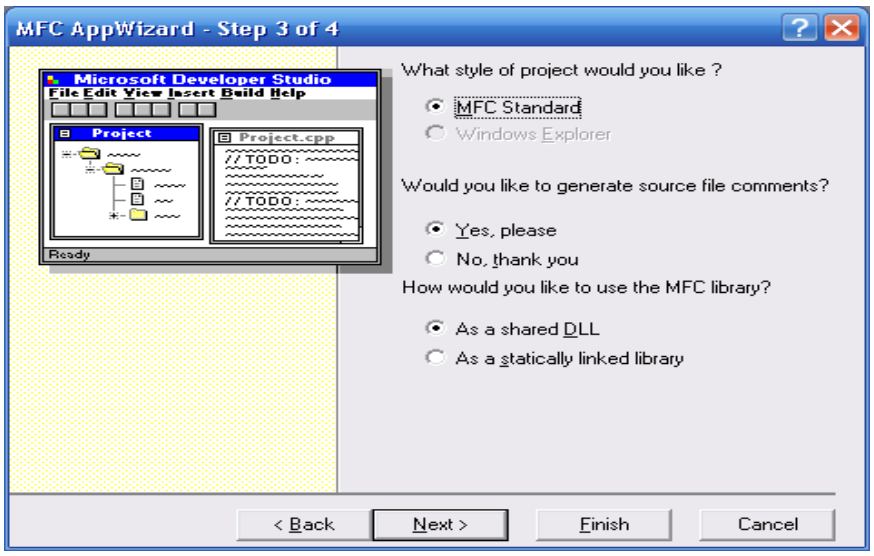
- у групі **Would you like to include WOSA support?** (чи є необхідність додати до проекту підтримку Internet?) є лише один прапорець:

Windows Sockets, призначення якого зрозуміле без перекладу. За замовчуванням підтримка Internet відсутня.

У текстовому полі **Please enter a title for your dialog** (введіть заголовок діалогового вікна) розміщується запропонований майстром **AppWizard** заголовок діалогового вікна, що створюється. Користувач може ввести до цього поля будь-який заголовок.



6 Залиште всі налаштування без змін та натисніть кнопку **Next>** . З’явиться діалогове вікно **MFC AppWizard – Step 3 of 4** (третій крок з чотирьох майстра MFC AppWizard).



• Група перемикачів **What style of project would you like?** (вибір стилю проекта) має такі положення:

MFC Standard (стандартний додаток MFC). Встановлено за замовчуванням;

Windows Explorer (стиль браузеру Internet). Можливість переключення до цього положення вимкнена в даному випадку.

• Група перемикачів **Would you like to generate sourse file comments?** (чи додавати коментарі до тексту вихідних файлів?) має такі положення:

Yes, please (так, будь-ласка). Встановлено за замовчуванням.

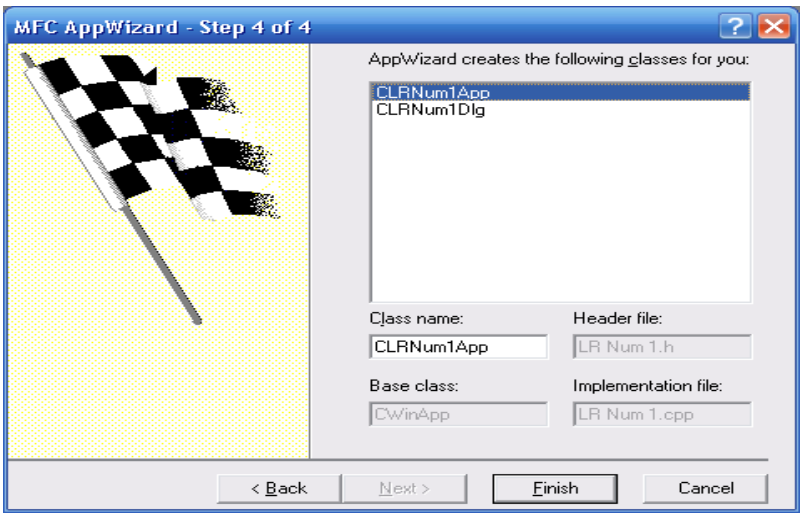
No, thank you (дякую, не треба)

• Група перемикачів **How would you like to use the MFC library**? (як використовувати бібліотеку MFC?) має такі положення:

As a shared DLL (як загальновживані бібліотеки динамічної компоновки). Цей вибір зменшує розмір файла, що виконується, та зменшує об’єм оперативної пам’яті, яка буде використана.

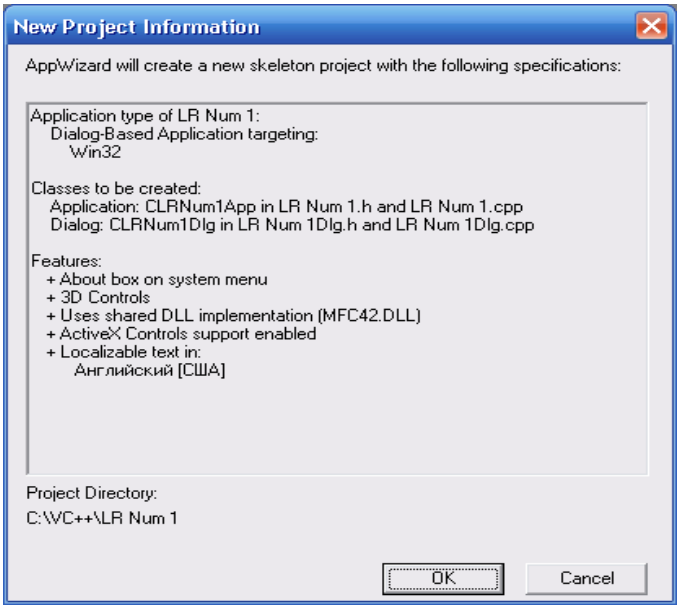
As a statically linked library (як статичні бібліотеки). Цей вибір дасть змогу програмі виконуватись на комп’ютері, де не встановлені необхідні бібліотеки динамічної компоновки.

7 Залиште налаштування без змін та натисніть кнопку **Next>**. З’явиться діалогове вікно **MFC AppWizard – Step 4 of 4** (четвертий крок з чотирьох).

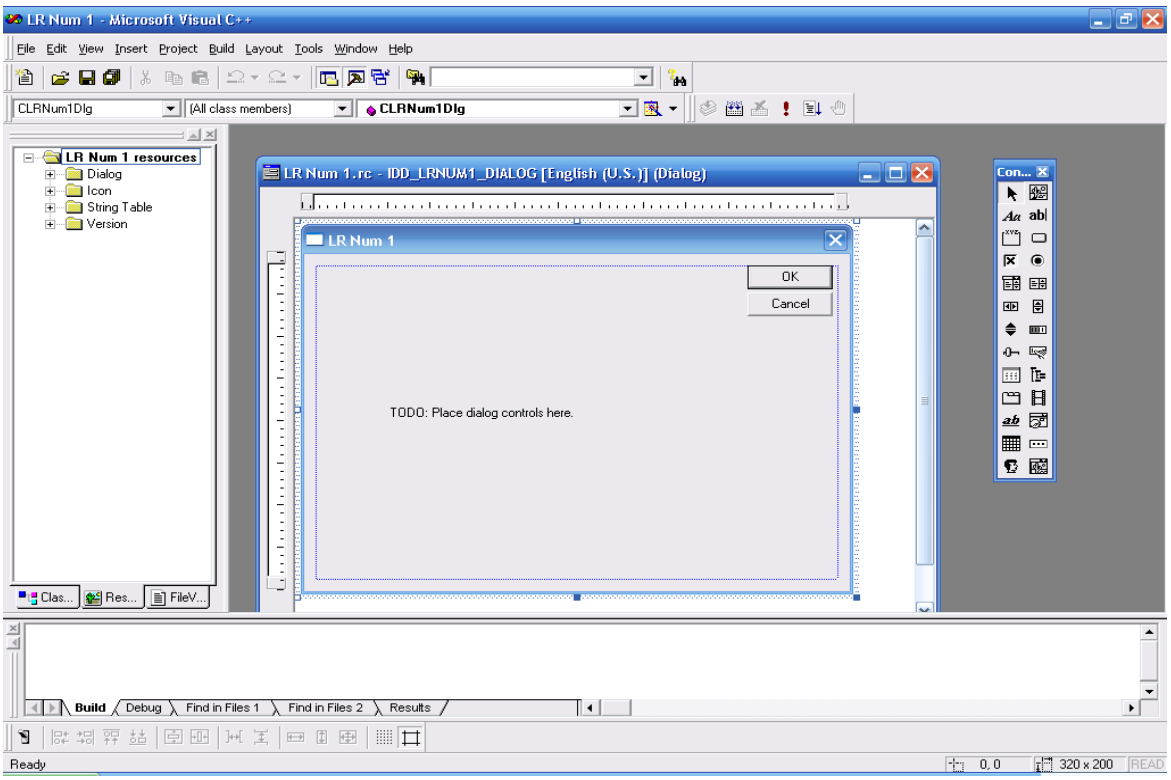


В цьому вікні перераховані класи, які створюються майстром. Якщо активувати клас у верхньому вікні у відповідних текстових полях, які розташовані нижче, з'явиться ім'я цього класу, ім'я базового класу, ім'я файлу заголовка, в якому знаходиться заголовок цього класу, та ім'я файлу реалізації, в якому знаходиться опис методів цього класу. Будь-яка інформація на білому фоні може бути змінена, на сірому – ні.

8 Натисніть кнопку **Finish.** З'явиться діалогове вікно **New Project Information**, в якому будуть перераховані значення всіх встановлених параметрів. Натисніть кнопку **ОК**.



Система генерує необхідні файли і відкриває макет діалогового вікна.



Праворуч від діалогового вікна розташована панель **Controls** (елементи керування), яка дозволяє автоматизувати створення діалогового вікна.

9 Видаліть з заготовки діалогового вікна статичний текст, оберіть на панелі інструментів **Controls** елемент керування **List Box** і розмістіть його на заготовці діалогового вікна.

10 Натисніть лівою кнопкою миші на **List Box.** З'явиться діалогове вікно **List Box Properties.**

11 Уведіть в текстове поле **ID:** ідентифікатор ресурсу IDC_LIST.

12 Розмістіть над ним статичний текст «МАСИВ».

13 Оберіть на панелі інструментів **Controls** елемент керування **Edit Box** і розмістіть його на заготовці діалогового вікна.

14 Розмістіть над ним статичний текст «ЕЛЕМЕНТ МАСИВУ».

15 Оберіть на панелі інструментів **Controls** елемент керування **Button** і розмістіть його на заготовці діалогового вікна.

16 Уведіть в текстове поле **ID:** ідентифікатор ресурсу IDC_ADD, а в текстове поле **Caption** – «Додати».

17 Повторіть операції описані в пп. 15 і 16, використавши ідентифікатор ресурсу IDC_DELETE і заголовок «Видалити».

18 Натиснувши правою кнопкою миші на заготовці діалогового вікна, викличте контекстне меню, оберіть пункт ClassWizard і відкрийте вкладку **Member Variables.** У списку **Control IDs:** виділіть ідентифікатор ресурсу IDC_EDIT1 і натисніть кнопку **Add Variable.** З'явиться діалогове вікно **Add Member Variable.**

19 Уведіть в текстове поле **Member Variables name:** ідентифікатор m_Edit і натисніть кнопку ОК.

20 У списку **Control IDs:** виділіть ідентифікатор ресурсу IDC_LIST і натисніть кнопку **Add Variable.** З'явиться діалогове вікно **Add Member Variable.**

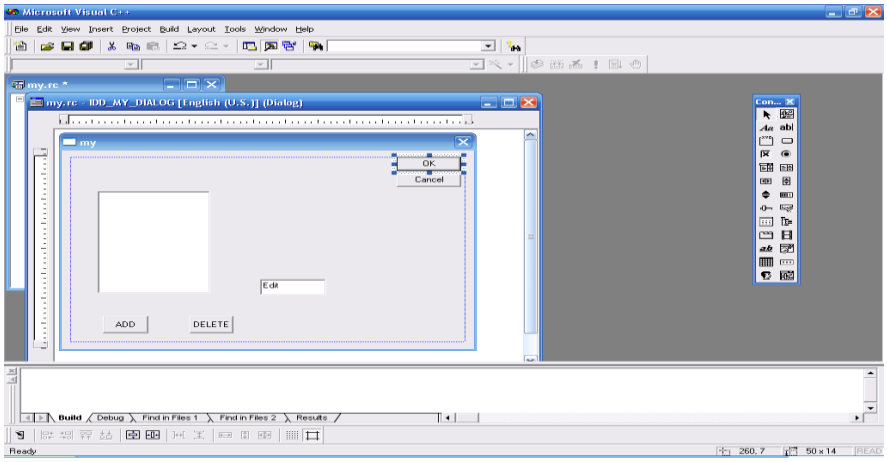
21 Уведіть в текстове поле **Member Variables name:** ідентифікатор m_List, а в списку **Category** оберіть опцію **Control** і натисніть кнопку ОК.

22 Відкрийте вкладку **Message Maps** діалогового вікна **MFC ClassWizard.**

23 У вікні списку **Object IDs:** виділіть ідентифікатор ресурсу IDC_ADD, у списку **Messages:** виділіть повідомлення «BN_CLICKED» і натисніть кнопку **Add Function.** У діалоговому вікні, що з'явилось, залиште все без змін і натисніть кнопку ОК.

24 Виконайте аналогічну операцію для ідентифікатора ресурсу IDC_DELETE.

25 У вікні списку **Object IDs:** виділіть ідентифікатор ресурсу IDC_List, у списку **Messages:** виділіть повідомлення «LBN_DBLCLK» і натисніть кнопку **Add Function.** У діалоговому вікні, що з'явилось, залиште все без змін і натисніть кнопку ОК. Вікно заготовки діалогового додатка буде мати вигляд



26 Натисніть кнопку Edit Code. Відкриється вікно редагування файла ListAppDlg.cpp, а курсор буде встановлений в тексті функції OnDblclkList.

27 Внесіть зміни у файл ListAppDlg.cpp:

```
void CMyDlg::OnAdd()
{
   UpdateData();
   m_List.AddString(m_Edit);
}
void CMyDlg::OnDelete()
{
   if(m_List.GetCurSel()!=LB_ERR)
       m_List.DeleteString(m_List.GetCurSel());
   UpdateData();
}
void CMyDlg::OnDblclkList1()
{
   CString Temp;
   if(m_List.GetCurSel()!=LB_ERR)
   {m_List.GetText(m_List.GetCurSel(),Temp);
   SetDlgItemText(IDC_EDIT1, Temp);}
```

28 Оберіть команду Build, Execute ProgressApp.exe і дайте ствердну відповідь на питання про необхідність створення файла, що виконується. Почнеться компіляція програми.

29 Уведіть в текстове поле ЕЛЕМЕНТ МАСИВУ двозначне ціле число і натисніть кнопку ДОДАТИ. Число з’явиться у списку.

30 Введіть таким чином три елементи масиву.

31 Двічі натисніть лівою кнопкою миші на другому рядку списку. У текстовому полі ЕЛЕМЕНТ МАСИВУ з’явиться число з другого рядка списку.

32 Залишивши рядок виділеним натисніть кнопку ВИДАЛИТИ. Цей рядок буде видалено зі списку.

33 Оберіть на панелі інструментів **Controls** елемент керування **Button** і розмістіть його на заготовці діалогового вікна.

34 Уведіть в текстове поле **ID:** ідентифікатор ресурсу IDC_max, а в текстове поле **Caption** – «MAX».

35 Оберіть на панелі інструментів **Controls** елемент керування **Edit Box** і розмістіть його на заготовці діалогового вікна.

36 Розмістіть над ним статичний текст «МАКСИМАЛЬНИЙ ЕЛЕМЕНТ МАСИВУ».

37 Відкрийте вкладку **Message Maps** діалогового вікна **MFC ClassWizard.**

38 У вікні списку **Object IDs:** виділіть ідентифікатор ресурсу IDC_max, у списку **Messages:** виділіть повідомлення «BN_CLICKED» і натисніть кнопку **Add Function.** У діалоговому вікні, що з'явилось залиште все без змін і натисніть кнопку ОК.

39 Натисніть кнопку Edit Code. Відкриється вікно редагування файла ListAppDlg.cpp, а курсор буде встановлений в тексті функції Onmax.

40 Внесіть зміни у файл ListAppDlg.cpp:

```
void CMyDlg::Onmax()
{
int mas[3],n, i,max;
char k[10];
CString s;
for(i=0;i<3;i++){
   m_List.GetText(i,s);
   int n=atoi(s);
mas[i]=n;
}
max=mas[0];
for(i=0;i<3;i++)
if(max<mas[i])max=mas[i];
itoa(max,k,10);
SetDlgItemText(IDC_max, k);
FILE *file;
   if ((file=fopen("C:\\OUT.txt","w"))==NULL) {
        printf("Невозможно открыть файл\n");
        exit(1);
   };
for(i=0;i<3;i++){
   fprintf(file,"%d\n",mas[i]);
};
```

```
fclose(file);
}
```

41 Оберіть команду Build, Execute ProgressApp.exe і дайте ствердну відповідь на питання про необхідність створення файла, що виконується. Почнеться компіляція програми.

42 Повторіть операції з пп. 29-30 і натисніть кнопку MAX. У вікні МАКСИМАЛЬНИЙ ЕЛЕМЕНТ МАСИВУ з'явиться найбільше із введених чисел.

3 Варіанти індивідуальних завдань

Найкраще значення параметра системи автоматизованого управління відповідає максимальному чи мінімальному значенню деякої функції на заданому інтервалі. Скласти схеми алгоритмів розв'язання задач.

Рівень 1

1 Знайти MIN значення функції:

$$Y = A \cdot E^{-B \cdot X} \cdot \sin(W \cdot X + F), \qquad x \in [0, T], \quad Hx = H.$$

2 Визначити MAX та MIN значення функції:

$$Y = A \cdot E^{-B} + C \cdot E^{-D}, \qquad B \in [W, F], \quad Hb = H.$$

Якщо A>0 , визначити MAX, якщо A<0 – MIN.

3 Визначити MAX значення функції:

$$Y = \cos(C \cdot Z) \cdot (X + Y), \qquad C \in [0, \pi], \quad Hc = \pi/6.$$

4 Визначити різницю між MAX та MIN значеннями функції:

$$Y = \frac{A \cdot X^{-C}}{B + K}, \quad C \in [-3,5], \quad Hc = 0{,}5.$$

5 Визначити , чи позитивне MAX значення функції:

$$Y = X \cdot \sin(-B \cdot X), \quad X \in [\pi/12, \pi], \quad Hx = \pi/12.$$

Рівень 2

1 Визначити значення аргументу, при якому досягається MAX значення функції:

$$Y = \frac{A \cdot X^A + B \cdot X}{C}, \quad X \in [M, K], \quad Hx = H.$$

2 Визначити значення аргументу, при якому досягається MIN значення функції:

$$Y = \frac{A \cdot E^B}{B + A}, \quad A \in [F, Q], \quad Ha = H.$$

3 Визначити MIN значення функції:

$$Y = A \cdot X^{-C \cdot X}, \quad X = \frac{A + C}{R}, \quad \begin{matrix} A \in [K, L], & Ha = 1 \\ C \in [M, K], & Hc = 1. \end{matrix}$$

4 Визначити MAX значення функції:

$$Y = \sqrt{|B|} \cdot X^{-D \cdot C}, \quad C = 2 \cdot \sqrt{|B|}, \quad \begin{matrix} B \in [M, K], & Hb = H \\ D \in [-2,3], & Hd = 0{,}5. \end{matrix}$$

5 Визначити різницю між аргументами, при яких функція досягає MIN і MAX:

$$Y = \frac{A \cdot X + C \cdot X^D}{2 + \sin(B \cdot X)}, \quad X \in [M, K], \quad Hx = H.$$

Рівень 3

1 Визначити, чи є 4-те значення функції найбільшим:

$$Y = B \cdot E^{-SIN(F)}, \quad F \in [0;0{,}1], \quad Hf = 0{,}01.$$

2 Визначити найбільше від’ємне значення функції:

$$Y = \frac{\cos(A \cdot X)}{K}, \quad X \in [R, Q], \quad Hx = H.$$

3 Визначити, яким за рахунком є MAX значення функції:

$$Y = \frac{\sqrt{|F \cdot K|}}{X^A}, \quad B \in [R, F], \quad Hb = H.$$

4 Визначити найменше додатне значення функції:

$$Y = \ln(1 + E^{C+Z}), \quad C \in [2,10], \quad Hc = 1.$$

5 Визначити, яким за рахунком є найбільше від'ємне значення функції:

$$Y = (X + C) \cdot \sin(B \cdot X), \quad X \in [-\pi/3, \pi/2], \quad H = \pi/12.$$

СПИСОК ЛІТЕРАТУРИ

1 Бантюков, С. Є. Використання інтегрованого середовища BORLAND C++ для розв'язання інженерно – технічних задач [Текст] : консп. лекцій з дисц. «Комп'ютерна техніка та програмування»/ С. Є. Бантюков. – Харків: УкрДАЗТ, 2006. –Ч. 1, 2.

2 Павловская, Т. А. C/C++. Программирование на языке высокого уровня [Текст] / Т. А. Павловская. – СПб., 2003. – 461 с.

3 Болотов, О. Б. Основи алгоритмізації обчислювальних процесів [Текст] : метод. вказівки до лаб. робіт з дисц. «Комп'ютерна техніка і організація обчислювальних робіт» / О. Б. Болотов. – Харків: УкрДАЗТ, 2010. – 54 с.

4 Основи алгоритмізації базових обчислювальних процесів [Текст] : навч. посібник / О. В. Головко, В. С. Меркулов, В. О. Гончаров, І. Г. Бізюк, В. М. Бутенко. – Харків: УкрДАЗТ, 2008.– 163 с.

5 Бутенко, В. М. Методичні вказівки до виконання лабораторних робіт з дисципліни «Автоматизація технологічних процесів» Schnider Electric [Текст] / В. М. Бутенко. – Харків: УкрДАЗТ, 2015. – 54 с.

6 Бутенко, В. М. Методичні вказівки до виконання лабораторних робіт з дисципліни «Технічні засоби автоматизації» [Текст] / В. М. Бутенко. – Харків: УкрДУЗТ, 2016. – Ч. 1. – 40 с.

7 Завдання і методичні вказівки до розрахунково-графічної та контрольної робіт з дисциплін «Програмування» та «Інформатика» для студентів факультету АТЗ [Текст] / В. М. Бутенко, О. В. Головко, М. О. Колісник, С. О.Бантюкова. – Харків : УкрДАЗТ, 2016. – 74 с.

8 Меркулов, В. С. Методичні вказівки з варіантами завдань для виконання контрольних робіт з дисципліни «Обчислювальна техніка, програмування, моделювання систем» [Текст] / В. С. Меркулов, В. М. Бутенко, О. В. Чаленко. – Харків: УкрДАЗТ, 2013. Ч. 2. – 51 с.

9 Меркулов, В. С. Методичні вказівки з варіантами завдань для виконання контрольних робіт з дисципліни «Обчислювальна техніка, програмування, моделювання систем» [Текст] / В. С. Меркулов, В. М. Бутенко, О. В. Чаленко – Харків: УкрДАЗТ, 2013. Ч. 3. – 51 с.