

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет "Запорізька політехніка"

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт із дисципліни

"Програмування"

для студентів спеціальності 123 «Комп'ютерна інженерія»

усіх форм навчання

"Робота з функціями"

Методичні вказівки до виконання лабораторних робіт із дисципліни "Програмування" для студентів спеціальності 123 "Комп'ютерна інженерія" усіх форм навчання. Робота з функціями / Укл.: Н.В. Луценко. – Запоріжжя: НУ "Запорізька політехніка", 2020. – 30 с.

Укладачі:

Н.В. Луценко, ст. викладач

Рецензент:

Р.К.Кудерметов, к.т.н., доцент

Відповідальний за випуск

Н.В. Луценко, ст. викладач

Затверджено
на засіданні кафедри КСМ
Протокол № 5 від 29.01.2020

Рекомендовано до видання
на засіданні НМК КНТ
Протокол № 6 від 30.01.2020

ЗМІСТ

ПОРЯДОК ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ	4
ЗАГАЛЬНІ ВІДОМОСТІ.....	5
ЛАБОРАТОРНІ РОБОТИ	14
ЛАБОРАТОРНА РОБОТА № 1 Елементарне введення у функції...	14
Контрольні питання до лабораторної роботи № 1	21
ЛАБОРАТОРНА РОБОТА № 2 Прийоми використання функцій...	22
Контрольні питання до лабораторної роботи № 2	29
РЕКОМЕНДОВАНА ЛІТЕРАТУРА	30

ПОРЯДОК ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ

Лабораторні роботи складаються з декількох програм, завдань до них і контрольних питань. Студент повинен розробити, відпрацювати і зберегти у робочому каталозі програму, що наведена, й програму, розроблену згідно заданого варіанта, та проаналізувати результати їхньої роботи. За результатами виконання кожної лабораторної роботи студент складає й захищає письмовий звіт, вимоги до якого викладені нижче.

Зміст звіту

1. Титульний лист, виконаний відповідно до СТП.
2. Мета роботи.
3. Відповіді на контрольні питання.

Поради та рекомендації

Запускаємо середовище програмування **Microsoft Visual C++ 2010 Express Edition**.

При цьому відкривається початкова сторінка, в якій потрібно створити пустий проект. Для цього обирається вкладка **File (Файл)** та пункт **New->Project (Создать->Проект)**. В вікні, що з'явилося, обирається **Visual C++**, а справа обирається значок **Empty project (Пустой проект)**. Нижче задається назва проекту та вказується папка, в якій буде зберігатися цей проект.

Для додавання в проект файлу з кодом програми використовується меню **Project (Проект)** та пункт **Add New Item (Добавить новый элемент)**. В вікні, що з'явилося, обирається значок **C++File.cpp (Файл.cpp)**, а нижче задається ім'я нового файлу. Далі натискається кнопка **Add (Добавить)**. Відкривається сторінка редактора, в якому пишеться код програми.

Для запуску програми обирається значок ► (**Start Debugging**).

Для відкриття написаної програми обирається вкладка **File (Файл)** та пункт **Open->Project (Открыть->Проект)**.

ЗАГАЛЬНІ ВІДОМОСТІ

Функції. Це елементи програми, що можуть використовуватися неодноразово для виконання деякої, заздалегідь визначеної задачі. Кожна програма містить тільки одну головну функцію з ім'ям **main()**, за допомогою якої здійснюється вхід у відкомпільовану програму. Крім головної функції в програмі може бути багато інших функцій.

Усім іменам функцій за умовчанням привласнюється клас пам'яті **extern**, тобто функції глобальні, мають зовнішній вигляд компонування, статичну тривалість існування (до кінця виконання програми) і доступні за певних умов в усіх модулях програми.

Будь-яка функція, окрім головної функції **main()**, має бути визначена або описана до її використання (виклику) в конкретному модулі.

Визначення функції має наступний синтаксис:

```
тип_функції ім'я_функції(список_формальних_параметрів)  
{ тіло функції }
```

де **тип_функції** – це тип значення, що повертається функцією. Якщо функція нічого не повертає, то тип функції позначається **void**;

список_формальних_параметрів (чи сигнатура функції) – це список специфікацій окремих параметрів з вказівкою типів і імен, що записані через кому, або **void** чи порожньо, якщо у функцію не передаються ніякі параметри;

тіло функції – це послідовність описів, визначень змінних та різних операторів, що поміщені у фігурні дужки.

Коли функція викликається, управління передається в початок тіла функції. Повернення в точку виклику функції виконується оператором **return випаз;** або **return;**. У тілі функції може бути декілька операторів **return**, якщо вихід з функції здійснюється в декількох точках.

Наприклад, визначення функції, в яку передаються три параметри типу **float** і яка знаходить суму цих чисел та повертає результат:

```
float sum(float a, float b, float c)  
{return a+b+c;}
```

В цьому прикладі **float** – тип значення, яке повертає функція, **sum** – ім'я функції, за яким у круглих дужках стоїть список формальних параметрів **a, b, c**, а у фігурних дужках – тіло функції.

Якщо функція не повертає ніякого значення, то тип її **void**, а оператор **return** може бути без виразу або бути відсутнім. Наприклад:
void func1() {cout<<"pi="<<3.141592;}

Звернення до функції (або *виклик функції*). Це вираз, в якому вказується ім'я функції та список фактичних параметрів в круглих дужках. Результат цього виразу – значення, що повертає функція, тип якого співпадає з типом функції:

ім'я_функції(список фактичних параметрів) ;

де **список_фактичних_параметрів** – це порожньо або **void**, (якщо в функцію нічого не передається), або фактичні параметри (константи або раніш визначені змінні, змінні з перетворенням типу, вирази).

При звернення до функції *формальні параметри* змінюються на *фактичні параметри* в строгій відповідності до типів параметрів по їх взаємному розташуванню в списках. Наприклад:

```
int max(int x, int y)
{if (x>y) return x; return y;}
void main()
{int a=10,res1,res2; float f=55.9;
 res1=max(a,20); // res1=20
 res2=max(int(f),a+30);} // res2=55
```

Всі параметри функції, крім покажчика і масиву, передаються за значенням. Тобто, для переданих (фактичних) параметрів в функції створюються копії цих параметрів, під які в стеку виділяється пам'ять. Дії, що виконуються над цими копіями в тілі функції, не впливають на значення фактичних параметрів поза функцією.

При передачі в функцію покажчиків (адрес фактичних параметрів) в функції створюються копії цих адрес, і в тілі функції становиться відомо, де знаходяться фактичні параметри, і, тому, їх значення можна змінити через розіменування покажчика.

Прототип функції. Якщо функція визначена після функції **main()** або в іншому файлі, то до першого звернення до функції вона має бути описана. Опис функції називається прототипом. Прототип – це заголовок функції, наприкінці якого обов'язкова крапка з комою. Прототип потрібен для перевірки відповідності типів та кількості переданих у функцію параметрів і типів повернутих значень. У прототипі можна не вказувати імена формальних параметрів. Приклади прототипів:

```
float sum(float a, float b, float c);
int max(int, int);
```

Функції і масиви. При передачі масиву у функцію в якості параметру вказується ім'я масиву (показчик-константа), тобто передається адреса початку масиву. Тому усі дії усередині функції проводяться саме з елементами масиву-оригіналу. При цьому список формальних параметрів в заголовку функції має дві рівноправні форми:

```
int func1(int arr[]){...}
```

```
чи int func2(int *mass){...}
```

У обох випадках у функції створюється копія показчика відповідного типу, який діє усередині функції як звичайний показчик-змінна і може змінювати своє значення. Наприклад, **arr++**.

Доступ до елементів масиву усередині функції можна виконувати за допомогою індексів та за допомогою операції розіменування:

```
arr[0], arr[i], *(arr+0), *(arr+i).
```

Для роботи з усіма елементами масиву необхідно знати кількість елементів в масиві. Є два способи визначення кінця масиву:

- 1) в якості одного з параметрів у функцію передається розмір масиву;
- 2) останній елемент масиву задається будь-яким унікальним значенням, яке аналізують у функції.

Приклад. Функція обчислює суму елементів масиву.

```
int sum(int x[],int N) // N визначає кількість елементів в масиві
{int S=0;
for(int i=0;i<N;i++) S+=x[i];
return S;}

int main()
{int a[5]={1,3,5,10,2},c[3]={2,1,6}, res1,res2;
res1=sum(a,5); // виклик функції
res2=sum(c,3); // виклик функції
cout<<"res1="<<res1<<"\nres2="<<res2; // 21 9
return 0;}
```

Приклад. Функція обчислює кількість символів в слові.

```
int len(char *c) // визначення функції
{int i=0;
while(c[i++]!='\0'); // ознака кінця рядка символ '\0'
return i-1; } // довжина рядка без урахування символу '\0'

int main()
{char *str1="TERMINATOR", str2[6]="Рядок";
cout<<"length1="<<len(str1)<<"\nlength2="<<len(str2);
return 0;}
```

Функції з параметрами за умовчанням. Це функції, що мають в специфікації формальних параметрів значення за умовчанням. При зверненні до такої функції параметри за умовчанням можна не вказувати. Але якщо параметр має значення за умовчанням, то всі параметри, що йдуть за ним, повинні мати теж значення за умовчанням. Наприклад:

```
int def(int a,int b=3,int c=4)
{return a*b+c;}
int main()
{int am=2,bm=5,cm=6,dm1,dm2,dm3;
 dm1=def(am);           // a=2 b=3 c=4 dm1=2*3+4=10
 dm2=def(am,bm);        // a=2 b=5 c=4 dm2=2*5+4=14
 dm3=def(am,bm,cm);      // a=2 b=5 c=6 dm3=2*5+6=16
 cout<<"dm1="<<dm1<<" dm2="<<dm2<<" dm3="<<dm3<<endl;
 return 0;}
```

Функції із змінним числом параметрів. Це функції, в яких кількість і типи параметрів визначаються тільки під час їх виклику.

Синтаксис визначення прототипу таких функцій :

тип ім'я(специфікація явних параметрів, ...);

Остання кома необов'язкова. Три крапки повідомляють, що подальший контроль відповідності кількості параметрів і їх типів при обробці виклику не треба проводити. Перехід в списку від параметра до параметра здійснюється за допомогою покажчика, тип якого є типом змінної. При збільшенні покажчика на одиницю здійснюється перехід до наступної змінної такого ж типу. У C++ функції із змінним числом параметрів не працюють з типами **char**, **unsigned char**, **float**.

У таких функціях існують два способи визначення кількості параметрів: 1) передача у функцію в списку фактичних параметрів інформації про їх кількість; 2) додавання параметра-індикатора в кінець списку фактичних параметрів.

Приклади прототипів:

```
double sumd(int i,double d...); // i – кількість елементів в списку
double mult(double arg, ...);  // аналізується значення 0.0
```

Приклад. У функцію передається в списку явних параметрів кількість елементів, які підсумовуються.

```
double sumd(int i,double d...)//1-й параметр - число змінних
{double *ptr=&d; //покажчик на 2-й параметр типу double
```

```

double sum=0.0;
for(;i;i--) // 1-й параметр - кількість змінних в списку
sum+=*(ptr++);
return sum;}
void main()
{double d1=5.7, d2=1.2, s1,s2;
s1=sumd(2,d1,d2); //сума 2-х чисел
s2=sumd(4, d1, 1.25, d2, 2.5); //сума 4-х чисел
cout<<"s1="<<s1<<" s2="<<s2<<" s3="<<s3;
}

```

Рекурсивна функція – це така функція, яка містить в собі виклик цієї ж функції. Рекурсивний підхід слід уникати, якщо є очевидне ітераційне рішення.

Приклад. Обчислення факторіалу додатного цілого числа

```

int fact(int k)
{if(k<0) return 0;
if(k==0) return 1;
return k* fact(k-1);}
void main()
{int n=5;
cout<<n<<"!="<<fact(n);} // 5!=120

```

Функція, що підставляється (inline -функція) – це функція, тіло якої підставляється при кожному її виклику. Така функція визначається за допомогою слова **inline**.

Inline-функція обробляється компілятором, як така, що вбудовується, тобто замість кожного виклику цієї функції компілятор спробує підставити тіло функції. При багатократних викликах код програми може значно збільшитися, але швидкість роботи програми підвищиться за рахунок відсутності викликів цієї функції.

Для **inline-функції** прийнято: 1) вона не має бути рекурсивною; 2) функція не повинна містити операторів **for**, **while**, **do**, **switch**, **goto**.

У цих випадках компілятор видає попередження, і функція трактуватиметься як звичайна.

Наприклад.

```

inline float mod(float x, float y)
{return sgrr(x*x+y*y);}
void main()
{float f1=4.5, f2=-2.5, f3=-1.5, res1, res2;

```

```

    res1=mod(f1,f2); // 5.1478
//res1=sqrt(f1*f1+f2*f2);
    res2=mod(f1, f3); }// 4.743
//res2=sqrt(f1*f1+f3*f3); }

```

Передача параметрів у функцію main(). В функцію `main()` як і в будь-яку іншу функцію можуть передаватися параметри. Для цього функцію `main()` треба описати у формі:

[тип] main(int argc, char **argv) {...}

чи **[тип] main(int argc, char **argv, char **envp) {...}**

В першому випадку функція `main()` буде мати доступ до слів командного рядка, а в другому - ще і до рядків середовища програми, що визначає деякі особливості операційної системи, файлової системи, шляху пошуку (`path`) і ін.

Параметр **[тип]** може бути `int` чи `void`.

Параметр `int argc` визначає кількість слів у командному рядку.

Параметр `char **argv` – покажчик на масив слів з командного рядка.

Розмір масиву дорівнює `argc`. Нульове слово, на яке вказує `argv[0]`, є специфікацією програми, що запускається на виконання.

Параметр `char **envp` – покажчик на масив слів середовища програми.

Перевантажені функції. Це функції, що мають одне спільне ім'я, але відрізняються за сигнатурами (за типами та за кількістю переданих в функцію параметрів). Для забезпечення перевантаження необхідно для кожного імені визначити, скільки різних функцій пов'язано з ним. Розпізнавання таких функцій при виклику виконується тільки за сигнатурою (списком параметрів функції).

Наприклад, функції з ім'ям `sum` обчислюють суму двох або трьох параметрів типу `int`, чи двох параметрів типу `float`.

```

int sum(int x,int y) {return x+y;}
int sum(int x,int y, int z) {return x+y+z;}
float sum(float x, float y) {return x+y;}
void main()
{int a=2,b=5,c=6,d1,d2; float f1=1.5,f2=4.6,f3;
  d1=sum(a,b);           // 2+5=7
  d2=sum(a,b,c);         // 2+5+6=13
  f3=sum(f1,f2);         // 1.5+4.6=6.1
  cout<<"d1="<<d1<<" d2="<<d2<<" f3="<< f3<<endl;
}

```

Шаблони функцій. Для автоматизації перевантаження функцій різного виду в C++ введені шаблони функцій. Визначається шаблон сімейства функцій так:

```
template <class type1, class type 2, i т.д.>
```

де **type1**, **type2** і т.д. – формальні параметри шаблону, якими можна позначати тип функції, типи формальних та локальних параметрів в визначенні функції. Наприклад:

```
template <class A> //визначення шаблону функції  
A max1(A x, A y)  
{if(x>y) return x; return y; }  
int main()  
{int a1=-20,a2=10,a3;  
  a3=max1(a1,a2);          // A приймає значення int  
  float f1=-10.5,f2=-45.8,f3;  
  f3=max1(f1,f2);          // A приймає значення float  
  system("pause"); return 0;}
```

Параметри шаблонів мають наступні властивості:

1. Імена параметрів шаблону повинні бути унікальними;
2. Список параметрів шаблону не може бути порожнім;
3. Оголошення кожного параметра повинне починатися із слова **class**;
4. Ім'я параметра шаблону має властивості і права типу, область видимості усе визначення функції;
5. Усі параметри шаблону функції, що використовуються в визначенні функції, повинні обов'язково бути використані в сигнатурі функції.

Покажчики на функції. Ім'я функції, за визначенням, є покажчик, значенням якого є адреса розміщення функції в пам'яті. Якщо значення цієї адреси присвоїти іншому покажчику, то цей покажчик можна використовувати для виклику функції.

Покажчик на функцію визначається спеціальним чином:

```
тип_функції (*ім'я_покажчика) (специфікація_параметрів);
```

де **тип_функції** – тип значення, що повертається функцією;

специфікація_параметрів – типи формальних параметрів;

(*ім'я_покажчика) – ім'я покажчика на функцію в круглих дужках.

У визначенні покажчика на функцію тип значення, що повертається, і сигнатура функції повинні співпадати з типами і сигнатурами тих функцій, адреси яких передбачається присвоїти покажчику.

Приклад.

```
float mul(float x, float y){return x*y;}  
float sum(float x, float y){return x+y;}
```

```

void main()
{float f1=3.0, f2=9.0, f3;
 float(*ptr)(float, float); //покажчик на функцію
 ptr=sum; //покажчику присвоїли ім'я функції sum
 cout<<" sum="<<(*ptr)(f1,f2); // sum=12.0
 ptr=mul; //покажчику присвоїли ім'я функції mul
 cout<<"mul="<<(*ptr)(f1,f2)); }// mul=27.0

```

Покажчики на функцію можуть утворювати масив, при цьому тип і сигнатура в усіх функціях мають бути однакові.

Приклад. Визначений масив з трьох покажчиків на функції, що мають тип `int` з аргументом типу `int`

```

int f1(int x){return x+x;};
int f2(int x){return x*x;};
int f3(int x){return x%10;}; //побитовое отрицание
void main()
{int(*p[3])(int);
 p[0]=f1; p[1]=f2; p[2]=f3;
 int a=(*p[0])(4); // a=8
 int b=(*p[1])(a); // b=64
 int c=(*p[2])(b); // c=4
 cout<<"a="<<a<<" b="<<b<<" c="<<c; }

```

Можна для завдання типу покажчика на функцію застосовувати специфікатор `typedef`:

```

//визначення типу покажчика на функцію
typedef тип_функции(*ім'я_типу_покажчика)(спец_ія_парам);
// визначення покажчика на функцію, масиву покажчиків на функцію
ім'я_типу_покажчика ім'я_покажчика_ф-ції;
ім'я_типа_покажчика ім'я_покажчика_ф-ції[n];

```

Приклад

```

void t1(){cout<<"Hello, ";}
void t2(){cout<<"dear ";}
void t3(){cout<<"friend!\n ";}
typedef void (*TYPE)(); // ім'я_типу_покажчика
TYPE mpf[]={t1,t2,t3}; // масив покажчиків на функцію
void main()
{for(int i=0;i<3;i++)
 (*mpf[i])(); }

```

Покажчики на функції можуть використовуватися як аргумент функції. В цьому випадку у функцію можна передати покажчики на різні функції.

Приклад. Функція підсумовує додатні елементи масиву, а потім – від’ємні.

```
float dodat(float x){if(x<0) return 0; return x;}
float vid(float y){if(y>0) return 0; return y;}
typedef float PF(float);
float summa(PF f, float a[],int N)
{float s=0.0;
 for(int i=0;i<N;i++) s+=(*f)(a[i]);
 return s;}
void main()
{float res1, res2, A[4]={1.5,-2.1,3.4,-6.2};
 res1=summa(dodat,A,4); //1.5+3.4=4.9
 cout<<"res1="<<res1<<endl;
 res2=summa(vid,A,4); //-2.1-6.2=-8.3
 cout<<"res2="<<res2<<endl;}
```

Багатофайловий проект. Частіше програми створюються з декількох файлів, що містять згруповані за призначенням різні функції. При цьому в одному з файлів міститься головна функція **main()**, в інших - другі необхідні функції, дані й бібліотеки функцій.

Найпростіший спосіб збирання файлів у єдину програму, це створити програму з декількох файлів, використовуючи препроцесорну директиву **#include**, за допомогою якої в файл з головною функцією підключаються усі необхідні файли.

Інший спосіб об’єднати файли за допомогою проекту. При об’єднанні декількох файлів в один проект слід пам’ятати:

- 1) всі функції мають клас пам’яті **extern**, і вони доступні у всіх файлах одного проекту. Але для використання функцій, що визначені в інших файлах чи нижче першого звертання до неї, необхідно їх описати (вставити в код програми прототип цієї функції);
- 2) об’єкти (змінні, масиви, структури і т.д.), що визначені в файлах як глобальні (поза функцій), доступні у всіх файлах одного проекту. Але для використання їх в програмі (функції) раніш визначення, ці об’єкти необхідно описати поза функцій з специфікатором **extern**.

Кожен об’єкт може бути визначений тільки один раз, але може бути неодноразово описаний.

Можна помістити прототипи функцій, оголошення об’єктів в окремий файл з розширенням **.h**, та включати цей файл в початок інших файлів за допомогою препроцесорної директиви **#include**.

ЛАБОРАТОРНІ РОБОТИ

ЛАБОРАТОРНА РОБОТА № 1

Елементарне введення у функції

Мета роботи – вивчити елемент мови C(C++) – функції, навчитися створювати і використовувати їх.

Програма 1_1. В програмі показана робота найпростіших функцій.

```
#include <stdio.h>
#include <stdlib.h>
#include<iostream>
using namespace std;
int mul(int arg1,int arg2) // визначення функції mul
{return arg1*arg2; }
float func(float x,float y) // визначення функції func
{if(x!=0.0||y!=0.0)
    return 1/(x*x+y*y);
    else {puts("It is error!"); return -1;} }
double epsilon() // визначення функції epsilon
{double eps=1.0,eps1;
    do{eps*=0.5;
        eps1=eps+1.0;
        }while(eps1>1.0);
    return eps;}
int main()
{int a=2,b=3,c=5,d=6,k1,k2;
    puts("\nfunction 1:");
    k1=mul(a,b);    // 1-й виклик функції mul
    cout<<"a="<<a<<" b="<<b<<" k1="<<k1<<endl;
    k2=mul(c,d);    // 2-й виклик функції mul
    cout<<"c="<<c<<" d="<<d<<" k2="<<k2<<endl;
    float f1=0.5, f2=0.2, f3=0.0,r1,r2;
    puts("\nfunction 2:");
    r1=func(f1,f2); // 1-й виклик функції func
    cout<<"f1="<<f1<<" f2="<<f2<<" r1="<<r1<<endl;
    r2=func(f2,f3); // 2-й виклик функції func
    cout<<"f2="<<f2<<" f3="<<f3<<" r2="<<r2<<endl;
    puts("\nfunction 3:");
    double E=epsilon(); // виклик функції epsilon
    cout<<"E="<<E<<endl;
    system("pause"); return 0;}
```

Завдання до програми 1_1. Розробіть задані згідно варіанту функції та зробіть в головній програмі по два виклики цих функцій з різними параметрами. Для першого виклику кожної з функцій фактичні параметри проініціалізуйте, для другого – введіть з клавіатури.

1. Функція типу **int**, що повертає більше з трьох цілих чисел, та функція типу **float**, що повертає суму двох вхідних параметрів, якщо другий параметр додатний, інакше повертає різницю цих параметрів.
2. Функція типу **int**, що повертає суму остач від ділення першого та другого параметрів на 5, якщо суму більша 4, інакше повертає число 0, та функція типу **double**, що повертає мінімальне з двох вхідних параметрів.
3. Функція типу **float**, що повертає абсолютне значення різниці першого та другого аргументів, та функція типу **int**, що повертає мінімальне з трьох цілих чисел.
4. Функція типу **long**, що повертає добуток двох чисел типу **long**, якщо друге число менше 100, інакше повертає число 100, та функція типу **float**, що повертає середнє арифметичне трьох чисел.
5. Функція типу **int**, що повертає суму трьох вхідних параметрів, якщо значення першого параметру непарне, інакше повертає число 50, та функція типу **float**, що повертає різницю квадратів двох вхідних параметрів.

Програма 1_2. В програмі функція визначає суму елементів масиву. При цьому у функцію передається покажчик на масив та розмір масиву. Функція визначається наприкінці файлу. Компілятору повідомляється про подальше визначенні функції за допомогою прототипу функції.

```
#include <stdio.h>
#include <iostream>
#include <stdlib.h>
#include <time.h>
using namespace std;

float summa(float[], int); // визначення прототипу

int main()
{float res1,mas1[]={3.36,2.7,7.51,1.5};
  int i,n;
  n=sizeof(mas1)/sizeof(mas1[0]);
```

```

puts("  Massiv 1");
for(i=0;i<n;i++)
    cout<< mas1[i]<<"  ";
res1=summa(mas1,n); // 1-й виклик функції
cout<<"\nsuma1="<<res1<<endl;
float res2,mas2[10];
srand((unsigned)time(NULL)); //запуск генератора випадк. чисел
puts("  Massiv 2");
for(i=0;i<10;i++)
{mas2[i]=rand()%101*0.1;
    cout<< mas2[i]<<"  ";
}
res2=summa(mas2,10); // 2-й виклик функції
cout<<"\nsuma2="<<res2<<endl;
system("pause");
return 0;
}

float suma(float arr[],int size) // визначення функції
{float sum=0.0;
    for(int i=0;i<size;i++)
        sum+=arr[i];
    return sum;}

```

Завдання до програми 1_2. Розробіть задану згідно варіанту функцію та зробіть в головній програмі три виклики функції з різними параметрами. При першому виклику значення елементів масиву проініціалізуйте, при другому – задайте випадковими числами, при третьому – введіть з клавіатури. Всі масиви повинні мати різний розмір.

1. Функція, в яку передається масив типу **double**, що повертає середнє арифметичне елементів масиву.
2. Функція, в яку передається масив типу **int**, що повертає суму додатних парних елементів масиву.
3. Функція, в яку передається масив типу **float**, що повертає добуток додатних елементів масиву, що менші 10.0.
4. Функція, в яку передається масив типу **long**, що повертає суму від'ємних непарних елементів масиву.
5. Функція, в яку передається масив типу **float**, що повертає добуток від'ємних елементів масиву, що більші -10.0.

Програма 1_3. В програмі функція визначається індекс максимального елементу масиву. При цьому у функцію передається покажчик на масив та розмір масиву.

```
#include <stdio.h>
#include <iostream>
#include <stdlib.h>
#include <time.h>
using namespace std;

int max_ind(float[], int); // визначення прототипу

int main()
{float m1[6]={-3.36,132.7,770.51,31.5,-23.8,67.45};
 int i,ind1,ind2;
 puts(" Massiv 1");
 for(i=0;i<6;i++)
     cout<< m1[i]<<" ";
 ind1= max_ind(m1,6); // 1-й виклик функції
 cout<<"\nindex_max1="<<ind1<<" max1="<<m1[ind1]<<endl;
 float m2[5];
 puts(" Vvedite massiv 2");
 for(i=0;i<5;i++)
     cin>> m2[i];
 puts(" Massiv 2");
 for(i=0;i<5;i++)
     cout<< m2[i]<<" ";
 ind2= max_ind(m2,5); // 2-й виклик функції
 cout<<"\nindex_max2="<<ind2<<" max2="<<m2[ind2]<<endl;
 system("pause"); return 0;}

int max_ind(float arr[],int size) // визначення функції
{int index=0;
 for(int i=0;i<size;i++)
     if(arr[i]>arr[index]) index=i;
 return index;}
```

Завдання до програми 1_3. Розробіть задану згідно варіанту функцію та зробіть в головній програмі два виклики функції з різними параметрами. При першому виклику значення елементів масиву проініціалізуйте, при другому – введіть з клавіатури. Всі масиви повинні мати різний розмір.

1. Функція, в яку передається масив типу **double**, що повертає кількість додатних елементів масиву, що менші **10.5**.

2. Функція, в яку передається масив типу **int**, що повертає кількість додатних парних елементів масиву.
3. Функція, в яку передається масив типу **float**, що кількість від'ємних елементів масиву, що більші -20.5 .
4. Функція, в яку передається масив типу **long**, що повертає кількість від'ємних непарних елементів масиву.
5. Функція, в яку передається масив типу **short**, що повертає індекс мінімального елементу цього масиву.

Програма 1_4. В програмі визначається функція з параметрами за умовчанням.

```
#include <stdlib.h>
#include <time.h>
#include <iostream>
using namespace std;
int bigger(int var,int n=10) // визначення функції
{if(var<n) return n;
 return var;}
int main()
{int a=8, b, a1, b1;;
 cout<<"max=10\n";
 b=bigger(a); // виклик функції, значення n=10
 cout<<"a="<<a<<"   b="<<b<<endl;
 srand( (unsigned) time(NULL) ); //запуск генератора випадк.чисел
 cout<<"max=50\n";
 a1=rand()%101;
 b1=bigger(a1,50); // виклик функції, значення n=50
 cout<<"a1="<<a1<<"   b1="<<b1<<endl;
 system("pause"); return 0;}
```

Завдання до програми 1_4. Розробіть задану згідно варіанту функцію з трьома вхідними параметрами типу **float** **x**, **a**, **b**, два з яких (**a**,**b**) мають значення за умовчанням. Продемонструйте роботу цієї функції, зробивши в головній програмі три виклики функції з різною кількістю параметрів. Параметри для виклику проініціалізуйте. Запишіть коментарі до отриманих результатів.

1. Функція, якщо **a** додатне, то повертає значення $x*a$, інакше повертає значення $x*b$.
2. Функція, якщо **a** від'ємне, то повертає значення $x*x-a$, інакше значення $x*x-b$.

3. Функція, якщо a більше b , то повертає значення $x+a$, інакше значення $x+b$.
4. Функція, якщо x більше a , то повертає значення виразу $(x+a*b)$, інакше значення $(x-a*b)$.
5. Функція, якщо x менше b , яка повертає значення виразу $(x*a-b)$, інакше значення $(x*b-a)$.

Програма 1.5 Показана робота функцій зі змінним числом параметрів. В першій функції за допомогою першого параметра передається до неї значення кількості параметрів, що йдуть за першим. Ця функція повертає суму другого і наступних параметрів. Друга функція обчислює добуток аргументів, а ознакою закінчення списку параметрів є визначене значення параметра (0.0).

```
#include <stdlib.h>
#include <iostream>
using namespace std;
int varg1(int i,...) // визначення функції varg1
{int *num_arg=&i;
 int sum=0;
 for(;i;i--)
  sum+=*(++num_arg);
 return sum;}
double varg2(double arg,...) // визначення функції varg2
{double mul=1.0;
 double *ptr=&arg;
 if(*ptr==0.0) return 0;
 for(*ptr!=999.0;ptr++) mul*=*ptr;
 return mul;}
int main()
{int a=11,b=22,c=33,sum;
 double mul,x=2.25,y=4.4,z=5.0,end=999.0;
 sum=varg1(2,a,b); // виклик функції
 cout<<"sum(a,b)="<< sum<<endl;
 int j=3;
 sum=varg1(j,a,b,c); // виклик функції
 cout<<"sum(a,b,c)="<< sum<<endl;
 sum=varg1(6,a,b,c,100,200,700); // виклик функції
 cout<<"sum(...)"<< sum<<endl;
 mul=varg2(x,y,z,end); // виклик функції
 cout<<"mul(x,y,z)="<< mul<<endl;
 mul=varg2(x,y,z,1.5,4.2,end); // виклик функції
```

```
cout<<"mul(...)"<< mul<<endl;
system("pause"); return 0;}
```

Завдання до програми 1.5 Розробіть задану згідно варіанту функцію зі змінним числом параметрів та продемонструйте роботу цієї функції, зробивши в головній програмі кілька викликів функції з різними значеннями. Всі параметри мають тип **int**.

1. Функція обчислює добуток першого параметра і суми параметрів, що йдуть за другим: $F = \text{arg1} * (\text{arg3} + \text{arg4} + \dots)$, загальна кількість параметрів передається за допомогою другого параметру.
2. Функція обчислює результат від ділення першого параметра на суму параметрів, що йдуть за першим: $F = \text{arg1} / (\text{arg2} + \text{arg3} + \dots)$, ознакою закінчення списку параметрів є визначене значення параметра.
3. Функція обчислює суму додатних парних параметрів, загальна кількість параметрів передається за допомогою першого параметру.
4. Функція обчислює добуток додатних непарних параметрів, ознакою закінчення списку параметрів є визначене значення параметра.
5. Функція обчислює суму другого параметра і добуток додатних параметрів, що йдуть за другим: $F = \text{arg2} + (\text{arg3} * \text{arg4} * \dots)$, загальна кількість параметрів передається за допомогою першого параметру.

Програма 1.6 В програмі демонструється передача аргументів у функцію **main (int argc, char **argv, char **envp)**. Для роботи програми її необхідно запустити з командного рядка, наприклад, у середовищі **FAR**, **NC**, **VC** і ін. Для роботи програми в командному рядку треба набрати перше слово ім'я програми з розширенням **.exe**). друге слово **Ivanov**, а третє – ціле число.

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
using namespace std;
int main(int argc, char **argv, char **envp)
{for(int i=0;i<argc;i++)
  cout<<"argv["<<i<<"]="<<argv[i]<<endl;
  system("pause");
  char str[]="Ivanov"; int x;
  if(strcmp(argv[1],str)!=0) // порівняння 2-х рядків
    {cout<<"Error"; system("pause"); return 1;}
```

```

scanf(argv[2], "%d", &x); //перетворення рядка в ціле число
cout<<"x="<<x<<" x^2="<<x*x<<" x^3="<<x*x*x<<endl;
system("pause");
while(*envp) puts(*envp++);
system("pause");
return 0; }

```

Завдання до програми 1.6 Розробіть програму з використанням передачі аргументів в головну функцію з командного рядка. В функцію `int main(int argc, char**argv)` з командного рядка передаються задані згідно варіанту пароль, число та інші повідомлення. Якщо пароль правильний, то на екран виводиться вся інформація з командного рядка, прізвище і ім'я студента та таблиця множення на задане число, інакше виводиться повідомлення про неправильний пароль.

1. Місяць народження студента.
2. Назва міста, в якому проживає студент.
3. Ім'я улюбленої кінозірки.
4. Назва улюбленої спортивної ігри.
5. Улюблена пора року.

Контрольні питання до лабораторної роботи № 1

1. Приведіть основні складові визначення функції.
2. Які параметри називають фактичними, а які формальними?
3. Що таке прототип функції? Для чого та коли він використовується?
4. Для чого використовується оператор `return` у функції? Скільки таких операторів може мати функція?
5. В чому різниця передачі параметрів за значенням та за покажчиком?
6. Як передати в функцію масив? Приведіть два способи визначення кінця масиву. Привести приклади функцій.
7. Дайте визначення та приведіть приклад *прототипу* функції з параметрами за умовчанням.
8. Дайте визначення та приведіть приклад *прототипу* функції з змінним числом параметрів.
9. Як та які параметри передаються до функції `main()`?
10. Дайте визначення та приведіть приклади прототипів рекурсивних та підставляємих функцій.

ЛАБОРАТОРНА РОБОТА № 2

Прийоми використання функцій

Мета роботи – познайомитися і зрозуміти деякі сучасні засоби мови C++, такі як перевантаження функцій, шаблони функцій і покажчики на функції, вивчити прийоми створення програм, що складаються з декількох файлів.

Програма 2.1. В програмі є три функції з однаковими іменами, при компіляції компілятор розпізнає перевантажені функції по їх сигнатурах (у даному випадку за типами параметрів).

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
using namespace std;
char func1(char arr[],int n)//визначення 1 перевантаженої функції
{char sum=0;
  for(int i=0;i<n;i++)
    sum+=arr[i];
  return sum; }
int func1(int arr[],int n)//визначення 2 перевантаженої функції
{int sum=0;
  for(int i=0;i<n;i++)
    sum+=arr[i];
  return sum;}
float func1(float arr[],int n)//визначення 3 перевантаж. функції
{float sum=0.0;
  for(int i=0;i<n;i++)
    sum+=arr[i];
  return sum;}
int main()
{char resch,ch[5]={4,20,2,5,10};
  int resint,set[8]={2,3,5,7,9,11,25,1};
  float resfl,mass[6]={2.1,3.1,4.1,5.1,6.1,7.1};
  resch=func1(ch,5);    // виклик 1 перевантаженої функції
  cout<<"Sum of elements of array char="<<resch<<endl;
  resint=func1(set,8);  // виклик 2 перевантаженої функції
  cout<<"Sum of elements of array int ="<<resint <<endl;
  resfl=func1(mass,6);  //виклик 3 перевантаженої функції
  cout<<"Sum of elements of array float="<<resfl <<endl;
  system("pause");    return 0;}
```

Завдання до програми 2.1 Розробіть задані згідно варіанту перевантажені функції та продемонструйте роботу цих функцій, зробивши кілька викликів функції з різними значеннями. Змінні та масиви проініціалізуйте. Масиви повинні бути різного розміру.

1. Функції типу **int**, **long**, **float**, що повертають суму тільки додатних значень трьох чисел.
2. Функції, що повертають кількість додатних чисел в масивах типу **int**, **float**, **double**.
3. Функція типу **float**, що повертають середнє арифметичне двох, трьох або чотирьох чисел.
4. Функції типу **int**, **float**, **double**, що повертають мінімальний елемент масиву.
5. Функції типу **int**, **long**, **float**, що повертають максимальне з трьох чисел.

Програма 2.2. В програмі створюється шаблон функції для оцінки дисперсії результатів вимірів еталонної величини, заданих як масив цілих і дійсних чисел. При компіляції програми формуються дві функції, відповідно для масиву типу **int** й масиву типу **double**.

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
using namespace std;
template <class A> //визначення шаблону функції
double sigma(int m,A arr[],A e) //визначення функції
{A tmp,sum=0;
  for(int i=0;i<m;i++)
    {tmp=(arr[i]-e);
      sum+=tmp*tmp; }
  return (double)sum/m;}
int main()
{int iarr[]={20,21,21,20,18,19,19,20,21,21,22,21};
  double darr[6]={21.3,21.4,20.9,20.8,20.9,21.1};
  double sg1, sg2;
  int sz=sizeof(iarr)/sizeof(iarr[0]);
  sg1=sigma(sz,iarr,21); //виклик функції, A замінюється на int
  cout<<"sg1="<<sg1<<endl;
  sg2=sigma(6,darr,21.5); //виклик функції, A замінюється на double
  cout<<"sg2="<<sg2<<endl;
  system("pause"); return 0;}
```

Завдання до програми 2.2 Розробіть згідно варіанту шаблон для функцій та продемонструйте роботу цих функцій, зробивши в головній програмі кілька викликів функції з різними значеннями. Масиви повинні мати різний розмір.

1. Функції типу **int**, **float**, **double**, що повертають максимальний елемент масиву.
2. Функції типу **int**, **float**, **long**, що повертають мінімальне з трьох чисел.
3. Функції, що повертають кількість від'ємних чисел в масивах типу **int**, **float**, **double**.
4. Функції типу **int**, **long**, **char**, що повертають суму тільки парних значень з чотирьох чисел.
5. Функції, що повертають індекс мінімального елемента в масивах типу **int**, **float**, **double**.

Програма 2.3. В цій програмі реалізовано приклад застосування шаблону функції. Функція здійснює обмін значень вмісту двох покажчиків. Компілятор створить відповідну функцію в залежності від переданих до неї параметрів.

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
using namespace std;
template <class T>
void swap(T *arg1, T *arg2) //визначення функції
{ T tmp=*arg1;
  *arg1=*arg2; *arg2=tmp; }
int main()
{ char c1='A', c2='B';
  int i1=10, i2=20;
  puts("before swap()");
  printf("&c1=%p c1=%c\n", &c1, c1);
  printf("&c2=%p c2=%c\n", &c2, c2);
  printf("&i1=%p i1=%d\n", &i1, i1);
  printf("&i2=%p i2=%d\n", &i2, i2);
  swap(&c1, &c2); //виклик функції
  swap(&i1, &i2); //виклик функції
  puts("after swap()");
  printf("&c1=%p c1=%c\n", &c1, c1);
  printf("&c2=%p c2=%c\n", &c2, c2); }
```

```
printf("&i1=%p i1=%d\n",&i1,i1);
printf("&i2=%p i2=%d\n",&i2,i2);
system("pause"); return 0;}
```

Завдання до програми 2.3 Доповніть програму, застосувавши наведений в програмі шаблон функції для обміну значень між двома об'єктами, заданими згідно варіанту. Виведіть на екран адреси переданих покажчиків, вміст цих покажчиків та значення об'єктів.

1. Визначте та проініціалізуйте дві змінні типу **float**, поміняйте значення цих змінних.
2. Визначте та проініціалізуйте дві змінні типу **double**, поміняйте значення цих змінних.
3. Визначте та проініціалізуйте два покажчика на рядки, поміняйте значення цих покажчиків.
4. Визначте та проініціалізуйте дві змінні типу **float**, визначте два покажчики на ці змінні, поміняйте значення цих покажчиків.
5. Визначте і проініціалізуйте два масиви типу **int**, визначте покажчики на ці масиви, поміняйте значення цих покажчиків.

Програма 2.4. В цій програмі при розробці "універсальної" функції сумування використовується в якості параметру покажчик на функцію. При цьому застосовується тип покажчика на функцію. Функція сумує спочатку куби елементів масиву, а потім - їх квадрати.

```
#include <stdlib.h>
#include<iostream>
using namespace std;
float cube(float y) {return y*y*y;}
float sqr(float e) {return e*e;}
float pol(float y) {if(y>0) return y; return 0;}
typedef float (*PF)(float); //визначення типу покажчика на функцію
float somma(PF pf1,float a[],int N) //визначення функції
{float sum=0.0;
//обчислення суми квадратів (кубів) елементів масиву
//за допомогою покажчика на задану функцію
for(int i=0;i<N;i++)
sum+=(*pf1)(a[i]);
return sum;}

int main()
{float res1, res2, res3, A[3]={1.5,2.0,-3.0};
res1=somma(cube,A,3); // виклик функції
cout<<"res1="<<res1<<endl;
```

```
res2=summa(sqr,A,3); //виклик функції з покажчиком на функцію
cout<<"res2="<<res2<<endl;
res3=summa(pol,A,3); // виклик функції
cout<<"res3="<<res3<<endl;
system("pause"); return 0;}
```

Завдання до програми 2.4 Розробіть програму використання покажчика на функцію, як параметра функції, згідно заданому варіанту. Зробіть в головній програмі кілька викликів функції з різними значеннями. Всі параметри, крім розміру масивів, мають тип **float**.

1. Визначте дві функції, що обчислюють максимальне та мінімальне з трьох чисел. Потім визначте функцію, що помножує суму заданих трьох чисел на максимальне (мінімальне) число, яке обчислюється за допомогою покажчика на задану функцію.
2. Визначте дві функції, що обчислюють середнє арифметичне та середнє геометричне з двох чисел. Потім визначте функцію, що ділить добуток заданих двох чисел на середнє арифметичне (геометричне), яке обчислюється за допомогою покажчика на задану функцію
3. Визначте дві функції, що обчислюють максимальний та мінімальний елемент масиву. Потім визначте функцію, що обчислює суму елементів масиву без обліку максимального (мінімального) числа, яке обчислюється за допомогою покажчика на задану функцію.
4. Визначте дві функції, що обчислюють суму та добуток трьох чисел. Потім визначте функцію, що помножає максимальне з трьох чисел на суму чи добуток, які обчислюються за допомогою покажчика на задану функцію.
5. Визначте дві функції, що обчислюють суму (добуток) елементів масиву. Потім визначте функцію, що обчислює максимальний елемент масиву і ділит на суму чи добуток, які обчислюються за допомогою покажчика на задану функцію.

Програма 2.5. Створіть із декількох файлів програму способом багатофайлової компіляції. Спочатку проект складається з трьох файлів із вихідними текстами функцій і одного заголовного файлу. Після доробки продемонструйте викладачеві уміння створювати багатофайловий проект і програму.

```
//файл complex.h
struct CMP {float real; float imag;};
CMP sum(CMP, CMP);
CMP sub(CMP, CMP);
```

```
//файл sum.cpp
#include "complex.h"
CMP sum(CMP com1, CMP com2)
{CMP tmpc;
  tmpc.real=com1.real+com2.real;
  tmpc.imag=com1.imag+com2.imag;
  return tmpc;}

//файл sub.cpp
#include "complex.h"
CMP sub(CMP com1, CMP com2)
{CMP tmpc;
  tmpc.real=com1.real-com2.real;
  tmpc.imag=com1.imag-com2.imag;
  return tmpc;}

//файл test.cpp
#include <stdlib.h>
#include<iostream>
using namespace std;
#include "complex.h"
int main()
{CMP c1,c2,r1,r2;
  c1.real=4.1;  c1.imag=1.4;
  c2.real=1.5;  c2.imag=3.2;
  r1=sum(c1,c2);
  cout<<showpos<<r1.real<<r1.imag<<"i"<<endl;
  r2=sub(c1,c2);
  cout<<showpos<<r2.real<<r2.imag<<"i"<<endl;
  system("pause");  return 0;}
```

Завдання до програми 2. Створіть багатофайловий проект із приведених вище файлів і випробуйте його. (Ім'я проекту наберіть на англійській клавіатурі).

Розробіть ще три файли для роботи з комплексними числами. Перший файл повинний містити функцію введення з клавіатури комплексних чисел, другий – функцію згідно варіанту, третій – функцію виведення на екран комплексних чисел. Додайте усі ці файли в новий проект та перевірте всі операції з комплексними числами.

1. Визначте функцію множення двох комплексних чисел.
2. Визначте функцію знаходження модуля комплексного числа.
3. Визначте функцію знаходження аргумента комплексного числа.
4. Визначте функцію ділення дійсного числа на комплексне число.
5. Визначте функцію ділення двох комплексних чисел.

Для довідки:

Комплексні числа: $c_1 = a_1 + ib_1$, $c_2 = a_2 + ib_2$,

Модуль комплексного числа: $|c_1| = \sqrt{a_1^2 + b_1^2}$

Аргумент комплексного числа: $\arg(c_1) = \arctg \frac{b_1}{a_1}$

Множення комплексних чисел: $c_1 \cdot c_2 = (a_1 a_2 - b_1 b_2) + i(a_1 b_2 + a_2 b_1)$

Ділення дійсного числа d на комплексне: $\frac{d}{c_2} = \frac{d}{a_2^2 + b_2^2} (a_2 - ib_2)$.

Ділення комплексних чисел: $\frac{c_1}{c_2} = \frac{a_1 a_2 + b_1 b_2}{a_2^2 + b_2^2} + i \frac{a_2 b_1 - a_1 b_2}{a_2^2 + b_2^2}$.

Програма 2.6. Створіть із декількох файлів програму, використовуючи спосіб багатобайтової компіляції.

Програма повинна обчислювати визначений інтеграл від функції на заданому інтервалі методом трапецій.

Метод трапецій: $\int_a^b f(x) dx \approx \frac{h}{2} (y_0 + 2y_1 + \dots + 2y_{n-1} + y_n)$,

де $h = (b-a)/n$; $y_0 = f(a)$, $y_1 = f(a+h)$, ..., $y_n = f(b)$.

Функція для інтегрування і підінтегральна функція повинні бути оформлені у вигляді окремих модулів (файлів).

Прототипи підінтегральної функції й функції для інтегрування повинні мати вигляд:

```
double func1(double x); // x – аргумент функції
double func2(double x); // x – аргумент функції
double func3(double x); // x – аргумент функції
typedef double (*PF)(double); // визначення типу покажчика на функцію
double integ(PF pf, double a, double b, int n);
```

де a, b – границі інтервалу $[a, b]$; n – кількість ділянок, на які розбивається інтервал $[a, b]$; pf – покажчик на підінтегральну функцію.

Обчисліть інтеграли:

від функції $f(x) = x^2$ на інтервалі $[1.5, 4.0]$ при $n = 500$;

від функції $f(x) = 3x^2 - 0.5x + 1$ на інтервалі $[1.0, 2.0]$ при $n = 400$;

від функції $f(x) = 8x^2 - 2x$ на інтервалі $[2.0, 4.0]$ при $n = 400$.

Контрольні питання до лабораторної роботи № 2

1. Що таке перевантаження функцій? Для чого воно застосовується?
2. Наведіть приклади прототипів перевантажених функцій із сигнатурами, що відрізняються за кількістю параметрів та за типом параметрів.
3. Чи є помилки при визначенні прототипів наступних перевантажених функцій? Поясніть.

```
float func1(int, float);  
float func1(int, int);  
int func1(int, float);
```
4. Дайте визначення шаблону функції та параметру шаблону.
5. Приведіть правила визначення і використання шаблонів функції.
6. Чи є помилки в наступному запису? Поясніть.

```
template <class D, C>  
D func2(int, C);
```
7. Що таке покажчик на функцію? Приведіть основні правила його визначення.
8. Визначте покажчики на функції, що мають наступні прототипи:

```
float F1(int n, float arr[]);  
float *F2(int n, char *st);
```
9. Як у модулі (файлі) одержати доступ до змінної, визначеної в іншому модулі (файлі)?
10. Як у модулі (файлі) одержати доступ до функції, визначеної в іншому модулі (файлі)?

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Подбельский В.В. Язык Си++: Учеб. пособие. – М.: Финансы и статистика, 2007. – 560 с.
2. Подбельский В.В. Практикум по программированию на языке Си: Учебн. пособие. – М.: Финансы и статистика, 2004. – 576 с.
3. Подбельский В.В., Фомин С.С. Программирование на языке Си: Учебн. пособие. – 2-е изд., – М.: Финансы и статистика, 2009. – 600с.
4. Павловская Т.А. C/C++. Программирование на языке високого уровня. – СПб.: Питер, 2009. – 464 с
5. Крячков А.В., Сухинина И.В., Томшин В.К. Программирование на С и С++. Практикум: Учебное пособие для вузов / А.В. Крячков, И.В. Сухинина, В.К. Томшин. – 2-е изд., исправ. – М.: Горячая линия.-Телеком, 2000. – 344с.
6. Крупник А. Изучаем Си. – СПб.: Питер, – 2001. – 256с.
7. Стивен Прата. Язык программирования С++. Лекции и упражнения. – 6-е изд. / Пер. с англ. – М.: Вильямс, 2012. – 478 с.
8. C/C++. Структурное программирование. Практикум / Т.А. Павловская, Ю.А. Щупак. – СПб.: Питер, 2002. – 240с.
9. Культин Н.Б. С /C++ в задачах и примерах. – СПб.: БХВ: Петербург, 2001. – 288с.
10. Х.М. Дейтел, П.Дж. Дейтел. Как программировать на С++. – 5-е изд. / Пер. с англ. – М.: ООО "Бином-Пресс", 2008.– 1456 с.
11. Шилдт, Герберт. Полный справочник по С++. – 4-е изд. / Пер. с англ. – М.: Вильямс, 2008. – 800 с.
12. Брайан Керниган, Деннис Ритчи. Язык программирования С. – 2-ое изд. / Пер. с англ. – М.: Вильямс, 2009. – 532 с.