ЛАБОРАТОРНА РОБОТА № 12

Тема: Побудова підпрограм.

Мета: Навчитися створювати та використовувати підпрограми.

Порядок виконання роботи та методичні рекомендації до її виконання:

- створити новий консольний проект (Win32 Console Application) для виконання лабораторної роботи та зберегти його на власному мережевому диску;
- написати програмний код для виконання поставленого завдання згідно індивідуального варіанту;
- провести тестування програми з різним набором вхідних даних;
- побудувати блок-схему до написаної програми;
- оформити звіт до лабораторної роботи.

Теоретичні відомості

Функції є програмними блоками, які можуть викликатися з різних частин програми. При виклику в них передаються деякі змінні, константи, вирази, які є аргументами, які в самих процедурах і функціях сприймаються як *формальні параметри*. При цьому функції повертають значення певного типу, яке заміщає в викликаному виразі ім'я викликаної функції. Наприклад, оператор I = 5 * F (X); викликає функцію F з аргументом X, помножує повернене нею значення на 5 і присвоює результат змінній I.

Функція описується таким чином:

тип_повертаємого_значення ім'я_функції (список_параметрів)
{
оператори тіла функції;
}

Перший рядок цього *опису*, що містить тип значення, ім'я функції і список параметрів, називається заголовком функції. Тип повертаємого значення може бути будь-яким, крім масиву і функції. Можуть бути також функції, що не повертають жодного значення. У заголовку таких функцій тип значення, що повертається оголошується *void*.

Список параметрів, що укладається в дужки, являє собою розділений комами список виду

тип_параметра ідентифікатор_параметра

Наприклад , заголовок:

double FSum (double X1, double X2, int A)

оголошує функцію з ім'ям *FSum*, з трьома параметрами X1, X2 і A, з яких перші два мають тип *double*, а останній - *int*. Тип повертаємого результа - *double*. Імена параметрів X1, X2 і A - локальні, тобто вони мають значення тільки всередині даної функції і ніяк не пов'язані з іменами аргументів, переданих при виклику функції. Значення цих параметрів на початку виконання функції дорівнюють значенням аргументів на момент виклику функції. Виклик такої функції може мати вигляд:

FSum (Y, X2, 5);

Як правило (хоча формально не обов'язково), крім опису функції в текст програми включається також ***прототип функції*** - її попереднє оголошення. Прототип являє собою той же заголовок функції, але з точкою з комою ";" наприкінці. Крім того, в прототипі можна не вказувати імена параметрів.

При відсутності прототипів будь-яка використовуєма функція повинна бути описана до її першого виклику у тексті. Деколи при взаємозв’язаних викликах функцій одної з другої це взагалі неможливо.

Приклади прототипів:

double FSum(double X1,double X2, int A);

void SPrint(AnsiString S) ;

void F1(void);

Зазвичай функції приймають вказане число параметрів, однак можуть бути функції, які приймають різне число параметрів (їх кількість або типи невідомі завчасно). В цьому випадку в прототипі замість невідомого числа параметрів ставлять "...". Наприклад:

*int prf(char *format, . . .);*

Функція *prf* приймає один параметр *format* типу *char** і довільне число параметрів довільного типу.

Тепер розглянемо опис тіла функції. Тіло функції пишеться за тими же правилами, що і будь-який код програми, і може містити оголошення типів, констант, змінних і будь-які виконувані оператори. *Не можна оголошувати і описувати у тілі інші функції*. Таким чином, функції не можуть бути вкладені одна в одну. Всі оголошення у тілі функції носять локальний характер, оголошені змінні доступні тільки всередині даної функції.

Вихід з функції відбувається наступними шляхами. Якщо функція не повертає ніякого значення, то вихід з неї відбувається при досягненні фігурної дужки, що закриває її тіло, або при досягненні оператора return. Якщо функція має повертати деякі значення, то нормальний вихід з неї відбувається за допомогою оператора

return вираз;

де вираз має формувати значення, що повертається, і відповідати типу, оголошеному у заголовку функції.

Наприклад:

```
double FSum(double X1, double X2, int A)
{
return  A * (X1 + X2);
```

}

Список параметрів, переданий у функції, як було показано вище, складається з імен параметрів і вказівок на їх тип. Наприклад, в заголовку

double FSum (double X1, double X2, int A);

вказано три параметри X1, X2, A та визначено їх типи. Виклик такої процедури може мати вигляд:

FSum (Y, X2, 5);

Це тільки один із способів передачі параметрів в процедуру, званий передачею за значенням. Працює він так. У момент виклику функції в пам'яті створюються тимчасові змінні з іменами X1, X2, A, і в них копіюються значення аргументів Y, X2 і константи 5 . На цьому зв'язок між аргументами і змінними X1, X2, A розривається. Ви можете змінювати всередині процедури значення X1, X2 і A, але це ніяк не відіб'ється на значеннях аргументів. Аргументи при цьому надійно захищені від ненавмисної зміни своїх значень викликаною функцією. Це запобігає випадковим побічним ефектам, які так сильно заважають іноді створенню коректного і надійного програмного забезпечення.

Недоліком передачі параметрів за значенням є неможливість з функцій змінювати значення деяких аргументів, що в багатьох випадках дуже небажано.

Можливий і інший спосіб передачі параметрів - виклик за посиланням. У разі виклику за посиланням оператор виклику дає функції,що викликає, можливість прямого доступу до переданих даних, а також можливість зміни цих даних. Виклик за посиланням добрий у сенсі продуктивності, тому що він виключає накладні витрати на копіювання великих обсягів даних; в той же час він може послабити захищеність, тому що функція,що викликається, може зіпсувати дані, які передаються в неї.

Щоб показати, що параметр функції переданий за посиланням, після типу параметра в прототипі функції ставиться символ амперсанда (&); таке ж позначення використовується в списку типів параметрів у заголовку функції. У виклику такої функції досить вказати ім'я змінної і вона буде передана за посиланням. Реально у функцію передається не сама змінна, а її адреса, отримана операцією адресації (&).

Наприклад:

void square (int &); // Прототип функції обчислення квадрата
void square (int &a) // Заголовок функції
{
*a * = a; // Зміна значення параметра*
}

При передачі масиву в функцію як параметра заголовок функції містить тип та ім'я масиву з подальшими порожніми квадратними дужками. Наприклад, якщо функція *F* повинна приймати масив як параметр, її прототип може мати вигляд:

void F (int Ar [ ]);

Звернення до такої функції може бути записано так:

const int Amax = 10;
int A [Amax];
F (A);

У більшості випадків тільки імені масиву мало, щоб провести в функції обробку його елементів. Усередині функції потрібно знати розмір масиву, щоб можна було організувати його циклічну обробку. Тому зазвичай у функцію передається не тільки масив, але і його розмір. При цьому заголовок функції може мати вигляд:

void F(int Ar[ ], int N);

а виклик функції:

F (A, Amax);

Зразок виконання завдання

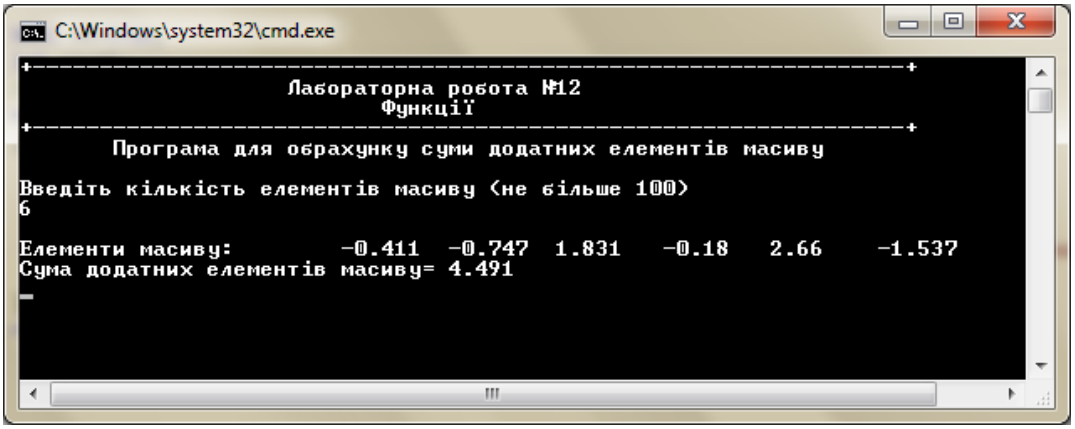
Написати програму для обрахунку суми додатних елементів масиву за допомогою підпрограм.

Програмний код:

```
#include <iostream>
#include <conio.h>
#include <windows.h>
#include <ctime>
using namespace std;
//Оголошення прототипів функцій
int GetElementCount();
double SumOfPositive(double *Array, int arrayCount);
void ShowArray(double *Array, int arrayCount);
void GenerateArray(double *Array, int arrayCount);
//Оголошення та визначення функції
float rnd(int min, int max)
{
    return min + rand() % (1000*(max-min)) / 1000.0f;
}
//Основна програма
int main(int argc, char* argv[])
{
    setlocale(LC_ALL, "Ukrainian");
    cout<< "+------------------------------------------------------------------  +"<<endl;
    cout<< "                    Лабораторна робота №12"<< endl;
    cout<< "                           Функції"<< endl;
    cout<< "+------------------------------------------------------------------  +"<<endl;

    //Оголошення змінних
    const int n=100;  //зарезервована к-сть елементів масиву
    int count;        //фактична к-сть елементів масиву
    double A[n];      //масив дійсних чисел
    double sum=0;     //сума додатніх елементів масиву
    cout<< "       Програма для обрахунку суми додатних елементів масиву"<< endl;
```

```
    count=GetElementCount();
    GenerateArray(A,count);
    ShowArray(A,count);
    sum=SumOfPositive(A, count);

    cout<<"\nСума додатних елементів масиву=\t"<< sum<<endl;
    getch();
    return 0;
}
    ///Обчислення суми додатніх елементів масиву
    double SumOfPositive(double *Array, int arrayCount)
    {
        double sum=0;
        for(int i=0; i<arrayCount;i++)
        {
            if(Array[i]>0) sum+=Array[i];
        }
        return sum;
    }
    ///Генерування значень елементів масиву
    void GenerateArray(double *Array, int arrayCount)
    {
        srand(time(0));
        for(int i=0; i<arrayCount;i++)
            Array[i]=rnd(-2,5);
    }

    ///Вивід елементів масиву у консоль
    void ShowArray(double *Array, int arrayCount)
    {
        cout<<"\nЕлементи масиву:\t";
        for(int i=0;i<arrayCount;i++)
               cout<<Array[i]<<"\t";
    }
    int GetElementCount()
    {
        bool ok=true;
        int count;
        //Ввід кількості елементів масиву, що будуть використовуватись
        while(ok)
        {
                cout<< "\nВведіть кількість елементів масиву (не більше 100)"<< endl;
                cin>>count;
                   if ((count>0)&&(count<=100))ok=false;
                   else cout<< "\n Кількість елементів має бути цілим додатнім числом, що менше 100."
                       <<"\nСпробуйте ще раз."<< endl;
        }
        return count;
    }
```

Результати виконання:



Варіанти індивідуальних завдань

Примітка: виконання всіх завдань передбачає використання функцій у програмному коді.

1. Масив R розміром n містить значення радіусів кіл. Обчислити довжину кола для кожного з n кіл та вивести результати на екран.

2. Напишіть функцію для обрахунку площі трапеції. Значення висоти і довжин основ трапеції введіть з консолі.

3. Напишіть функцію для обрахунку площі паралелограма. Значення висоти і довжини сторони, що перпендикулярна до висоти, введіть з консолі..

4. Напишіть функцію для обрахунку площі кола. Значення радіусу введіть з консолі..

5. У масиві розміром m містяться значення довжини ребра правильних тетраедрів. Обчисліть площу поверхні кожного тетраедра.

$S=\sqrt{3} * a*a$.

6. Напишіть функцію, що буде визначати чи є заданий текст числом.(Текст вводиться з консолі).

7. Напишіть функцію для обрахунку площі та периметра квадрата. Передбачте можливість вводу довжини сторони квадрата користувачем з консолі.

8. Напишіть функцію для обрахунку суми діагональних елементів матриці розміром n*n.

9. Напишіть функцію для обрахунку периметра прямокутника. Передбачте можливість вводу довжин сторін прямокутника з консолі

10. Напишіть програму, яка визначає реальну заробітну плату, якщо відомі такі показники: оклад, податок на доходи фізичних осіб-13%, збір до Пенсійного фонду-2%, збір до фонду соціального страхування -1%. Передбачити можливість вводу розміру окладу користувачем.