Комп’ютерний практикум №6

ОрганІзацІЯ пІдпрограм

*Мета роботи*  - набути навичок складання і використання підпрограм користувача.

* 1. Теоретичні відомостi

У програмуванні типовою є ситуація, коли в різних місцях програми необхідно виконати одну й ту ж саму послідовність дій (обчислити певну послідовність виразів, вивести одну й ту ж саму інформацію, тощо). Ці дії можуть бути досить складними і задаватися великими фрагментами програми. Для забезпечення компактності і підвищення наочності програм, усі логічно завершені її дії рекомендується оформлювати у вигляді підпрограм.

*Підпрограма*- іменований, спеціальним чином оформлений, логічно завершений програмний фрагмент, який в процесі роботи програми може виконуватись багаторазово при звертанні до нього з різних місць програми.

Щоб підпрограма кожного разу при звертанні до неї виконувалась з новими даними, ці дані передаються у неї як аргументи. Параметри підпрограм використовуються також і для повернення результатів роботи підпрограми.

Усі підпрограми користувача, що використовуються в програмі, повинні бути попередньо описані. Опис будь-якої підпрограми складається із заголовка, що містить ім'я підпрограми та, за необхідності, список *формальних* параметрів, і програмного блока (тіла), що описує дії, які виконує підпрограма.

*Формальні* параметри являють собою змінні, через які в підпрограму передаються вхідні дані і повертаються результати. Безпосередньо в термінах цих змінних і описуюються дії в тілі підпрограми, що робить їх загальними для різних вхідних даних.

Структура підпрограми, зазвичай, повторює структуру програми, але задані в ній оголошення є *локальними* - доступними тільки всередині підпрограми. Для задання дій в підпрограмі, можуть використовуватись як формальні параметри і локальні змінні, так і глобальні дані, що описані в основній програмі.

При активізації (виклику) підпрограми керування передається першому її оператору. Коли виконання підпрограми завершується, відбувається повернення керування в основну програму, яка продовжує своє виконання з наступного за оператором виклику оператора.

Для виклику підпрограми необхідно вказати її ім’я і, якщо потрібно, список *фактичних* параметрів, які є тими конкретними даними (кожного разу різними), що передаються у підпрограму, і змінними, яким повертаються результати роботи підпрограми.

Список фактичних параметрів в операторі виклику підпрограми повинен відповідати списку формальних параметрів у заголовку підпрограми. Ця відповідність повинна бути взаємно-однозначною по кількості, порядку і типу параметрів у результаті їхнього перебору зліва направо.

Сучасний підхід до розробки програм заохочує оформлення у вигляді підпрограми кожного логічно завершеного програмного фрагмента, навіть, якщо він виконується тільки один раз.

У C/C++ підпрограми реалізуються у формі *функцій*, які завжди повертають результат і тільки один.

Формат опису функцій у С/С++:

[ тип\_результата ] ім'я ([ список\_формальних\_параметрів] )

тіло\_функції.

Опис формальних параметрів у заголовку функції являє собою опис відповідних змінних:

тип ім'я.

Типи формальних параметрів функції можуть описуватися як в заголовку функції, так і після нього, але до тіла функції. Наприклад,

float fun(p1, p2)

float p1; float p2;

{ return p1>= p2 ? p1 : p2;

}

За відсутності специфікатора типу у деяких версіях передбачається, що функція повертає ціле значення (типу int).

Визначення функції має таку саму структуру, як і головна функція.

Повернення значення із функції виконується операторомreturn наступного формату:

return [ вираз ].

Наприклад,

float Distance( float x1, float y1, float x2, float y2)

{ return sqrt(pow((x1–x2),2)+pow((y1–y2),2));

}

Функція може не повертати жодного значення. Тоді тип її результату задається ключовим словом void. У цьому випадку керування із функції повертається або по досягненні кінця тіла функції (символа «}»), або при виконанні пустого оператора return*.*Наприклад,

void print()

{ cout<< “Друк рядка” << endl;

}

*Прототип функції* – це попереднє оголошення функції за допомогою тільки її заголовка. Воно передує традиційному визначенню функції.

Синтаксис прототипу функції:

тип\_результата ім’я(тип\_1, тип\_2, …, тип\_n).

Компілятор ігнорує ідентифікатори параметрів у прототипі функції, тому їх можна не вказувати. Тип результату, кількість, тип і порядок параметрів у прототипі функції повинні відповідати типу результата, кількості, типу і порядку параметрів у визначенні функції. Наприклад,

float Distance(float, float, float, float).

Вкладеність функцій у С/С++ не допускається, але прототипи функцій можуть розміщуватися всередині інших функцій.

Декларація функції користувача у програмі С/С++ здійснюється або у вигляді визначення функції, яке записують раніше оголошення головної функції main(), або у вигляді *прототипу* функції.

Виклик функції у С/С++ може здійснюватися як окремим оператором виклику, так і бути операндом виразу.

У мові С/С++ підтримуються такі способи передачі параметрів у функції:

* за значенням,
* за адресою з аргументами-покажчиками,
* за адресою з аргументами-посиланнями.

Предача параметрів функції за значенням передбачає створення копії значення фактичного параметра і передачі її відповідному формальному параметру. Зміна копії у тілі функції не впливає на значення оригіналу в операторі виклику.

Після ініціалізації формального параметра значенням фактичного параметра, останній є недоступним з функції - вона не в змозі ні змінити його, ані використати яким-небудь іншим способом.

Відповідні параметри називають *параметрами-значеннями*, оскільки кожен такий параметр в тілі функції представляє значення відповідного фактичного параметра.

Декларування такого способу передачі параметрів у С/С++ здійснюється наступним чином:

параметр-значення = тип ім'я {, ім'я}.

Наприклад,

float func(int par1, int par2).

Як фактичні параметри-значення можуть використовуватися константи, змінні, вирази.

Наприклад,

int func(int);

int main ( )

{ int n = 4;

n= func(n);

cout << “n = “ << n << endl;

system “pause“;

}

int func(int num);

{ return (num + 5) \*num;

}

Якщо передача параметрів відбувається за значеннями, то у тілі функції не можна змінити значення змінних, що є фактичними параметрами.

Однак, формальні параметри функції визначають не лише дані, які передаються на обробку до функції, а також дані, які повертаються функцією у вигляді результатів. Такі параметри в тілі функції повинні представлятися не значеннями, а деякими змінними, що існують поза тілом функції. Щоб функція могла присвоїти значення таким змінним, необхідно забезпечити безпосередній доступ до них із функції.

Такі можливості притаманні так званим *параметрам-змінним*. При виклику функції такому формальному параметру передається не значення фактичного, а адреса аргумента в сегменті даних, що дає змогу функції напряму модифікувати значення цього аргумента.

На відміну від формального параметра-значення, фактичним параметром для якого може бути будь-який вираз відповідного типу, для параметра-змінної фактичним параметром може бути лише змінна.

Якщо деякий параметр функції представляє результат її виконання, то він обов'язково повинен задаватися як формальний параметр-змінна.

У мові С/С++ для реалізації даного способу передачі параметрів треба описати відповідний формальний параметр як покажчик на деяку змінну. У цьому випадку формальному параметру передається адреса аргумента, що надає функції можливість прямого доступу до даних, які передаються, а також можливість зміни цих даних, використовуючи операцію розіменування. Наприклад,

void prim (int \*);

main ( )

{ int n = 4;

prim (&n);

cout << “n = “ << n << endl;

system “pause“;

}

void prim (int \*nPtr);

{ \*nPtr = (\*nPtr + 5) \* \*nPtr;

}

Недоліком даного способу передачі параметрів є недостатня захищеність даних, оскільки функція має доступ до них і може зіпсувати дані, що передаються до неї.

Параметри-посилання є різновидом параметрів-змінних. При такій передачі відбувається пряме звертання до параметра, але через його псевдонім (інше ім’я).

Наприклад,

void prim (int &);

main ( )

{ int n = 4;

prim (n);

cout << “n = “ << n << endl;

system “pause“;

}

void prim (int &nRef);

{ nRef \*= nRef ;

}

* 1. Приклад програми

Приклад**.** За відомих координат вершин трикутника обчислити довжини його сторін.

#include <math.h>

#include <iostream>

using namespace std;

float x1,Y1,x2,y2,x3,y3; *//координати вершин трикутника*

float d1,d2,d3; *//довжини сторін трикутника*

//================= *прототипи функцій* ======================

float Distance(float,float,float ,float);

void Init(); *//введення даних*

void Solution(); *//розрахунок довжин сторін трикутника*

void Browse();  *//виведення результатів*

//================= *головна* *функція* =========================

int \_tmain(int argc, \_TCHAR\* argv[])

{ cout<<"calculate lengths of triangle side"<<endl;

Init();

Solution();

Browse();

system("pause");

}

//=============== *визначення* *функцій* =======================

float Distance(float a1,float b1,float a2,float b2)

{ return sqrt(pow((a1-a2),2)+pow((b1-b2),2));

}

void Init()

{ cout<<"enter triangle apexes coordinates"<<endl;

cout<<"coordinates x1,y1 "; cin>>x1>>Y1;

cout<<"coordinates x2,y2 "; cin>>x2>>y2;

cout<<"coordinates x3,y3 "; cin>>x3>>y3;

}

void Solution()

{ d1=Distance(x1,Y1,x2,y2);

d2=Distance(x2,y2,x3,y3);

d3=Distance(x1,Y1,x3,y3);

}

void Browse()

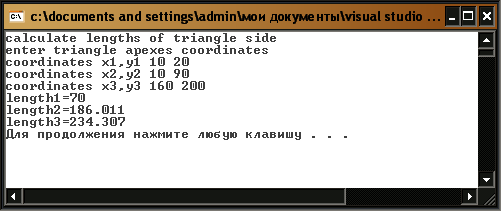
{ cout<<"length1="<<d1<<endl;

cout<<"length2="<<d2<<endl;

cout<<"length3="<<d3<<endl;

}

Відеокопія результату:



* 1. Варіанти завдань

1. Для заданих цілих чисел *a*, *b*, *с*, *d* обчислити

,

де  **.

1. Три трикутники задані своїми сторонами *А*, *В* та *С*. За допомогою формули Герона обчислити площі трикутників та визначити, площа якого з них є найбільшою. Формула Герона: , де *Р* - периметр трикутника.
2. Задані дійсні числа *a*, *b*, *с*. Обчислити

.

1. Для заданих цілих *m*, *k* (*k* >0), використовуючи ітераційну формулу

 , 

для знаходження кореня  при заданому натуральному *n* обчислити

.

1. Для заданого цілого *х*, використовуючи розкладання функції *e-x* в ряд Тейлора

e-x*=* ,

обчислити із заданою точністю *ε* значення

 .

1. Для заданого дійсного *а* >0, використовуючи рекурентну формулу Ньютона

, *y*0 = *a*

для знаходження кореня ** при заданому натуральному *n*, обчислити

.

1. Задані дійсні числа *s*, *t*. Обчислити значення

,

де .

1. Задане дійсне число *y*. Отримати

,

де .

1. Задані дійсні числа *a*, *b*. Отримати значення

; ; .

1. Для заданого дійсного *х*, використовуючи розкладання функції *Cos*(*x*) в ряд Тейлора

Cos(*x*) *=* ,

обчислитиіз заданою точністю *ε* значення

*y =* Cos( *a*) *+* Cos *2*( *b*) *+ Cos* ( *a+b*).

1. Для заданого дійсного числа *а* отримати

,

де .

1. Нехай *n* трикутників задані двома сторонами і кутом між ними. Визначити трикутник, що має максимальну площу.
2. Задані дійсні числа *s*, *t*. Визначити значення виразу

,

де .

1. Знайти усі досконалі числа в діапазоні від 1 до *n*. Число називається досконалим, якщо воно дорівнює сумі усіх своїх дільників за виключенням себе самого.
2. Для заданого цілого *х*, використовуючи розкладання функції *ex* в ряд Тейлора

*ex =* ,

обчислити із заданою точністю *ε* значення

.

1. Визначити чого більше: всіх можливих трьохзначних чисел, що записуються цифрами 1, 2, 3, 4, 5; всіх двозначних чисел, що записуються цифрами 2, 4, 6, 8; всіх чотирьохзначних чисел, що записуються цифрами 1, 3, 7, 8, 9? Кількість *k*–значних чисел із *n* цифр визначається так:



1. Для *х*, що змінюється від 0 до 3 з кроком 0.5, використовуючи розкладання функції *Ln*(*x*) в ряд Тейлора

Ln(*x*)=,

обчислити із заданою точністю *ε* значення

 .

1. Задане натуральне число *n*. З’ясувати, чи є серед чисел *n*, *n* + 1, …, 2⋅*n* числа-близнюки, тобто прості числа, різниця між якими дорівнює двом. Вивести усі пари таких близнюків.
2. Задані ціле число *n*, дійсне число *x*. Обчислити з точністю *ε* значення виразу

.

1. Задані невід‘ємні цілі числа *n* і *m*. Обчислити функцію Акермана, користуючись наступними рекурентними співвідношеннями:



1. Задані натуральні числа *n*, *і*, *m*. Знайти їх найбільший спільний дільник (НСД).
2. Для *х*, що змінюється від -2 до 2 з кроком 0.5, використовуючи розкладання функції *Sin*(*x*) в ряд Тейлора

Sin(*x*)*=*,

обчислити із заданою точністю *ε* значення

 .

1. Використовуючи формулу трапецій

 , 

для наближеного обчислення інтеграла при заданих цілих *a*, *b*, *n* знайти

.

1. Для заданого *n* обчислити і вивести на екран трикутник Паскаля:



1. Для заданих дійсних *а*, *b* (*а* >0, *b* >0)використовуючи формулу , *z*0 = *x*  для знаходження кореня ** при заданому натуральному *n*, обчислити .
2. Для заданого дійсного *х*, використовуючи розкладання функції *Arctg(x)* в ряд Тейлора

Arctg (*x*) *=* ,

обчислити із заданою точністю *ε* значення

.

1. Для заданих цілих *a*, *b*, *n*, використовуючи формулу прямокутників для наближеного обчислення інтеграла

,

де *h =* (*b - a*) */ n*; *xi = a + i⋅ h - h* / 2, *i =*1 *.. n*, знайти

.

1. Із заданою точністю *ε* > 0 обчислити площу заштрихованої фігури



1. Для заданих дійсних *а*, *b* (*а* >0, *b* >0)використовуючи формулу , *z*0 = *x*  для знаходження кореня ** при заданому натуральному *n*, обчислити .
2. Обчислити , де



* 1. Контрольні питання
  2. Що таке підпрограма та які переваги дає використання підпрограм?
  3. Які процеси відбуваються в оперативній пам’яті під час виклику підпрограми?
  4. Що таке точка входу до підпрограми та точка виходу з неї?
  5. Дайте визначення функції.
  6. Наведіть синтаксис оголошення функції у С/С++.
  7. Чи допускається в програмах на С/С++ вкладеність підпрограм ?
  8. Що таке прототип функції ?
  9. Що таке формальні параметри підпрограм ?
  10. Що таке фактичні параметри підпрограм ?
  11. У чому полягає відмінність формальних параметрів від аргументів функції?
  12. Які різновиди формальних параметрів існують у мовах С/С++?
  13. Структура звіту з лабораторної роботи

Звіт оформлюється у вигляді документу та складається із наступних частин:

1. Номер роботи та автор
2. Умова задачі
3. Блок-схема алгоритму вирішення задачі, виконана у Visio
4. Текст усіх файлів проекту
5. Копії екранних форм результатів роботи
6. Висновок, що містить обґрунтування коректності розробленого проекту