TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TP.HCM
KHOA CÔNG NGHỆ THÔNG TIN
MÔN: **NHẬP MÔN LẬP TRÌNH**

HƯỚNG DẪN THỰC HÀNH

TUẦN 05

HÀM (NÂNG CAO)

TP.HCM, ngày 09 tháng 09 năm 2017

MỤC LỤC

1 Khái niệm

Trong phần này ta sẽ tìm hiểu các vấn đề nâng cao liên quan tới hàm trong ngôn ngữ lập trình. Các vấn đề như hàm trùng tên, hàm có tham số mặc định, và hàm có tham số là hàm sẽ được phân tích. Ngoài ra ta cần học cách tách các tập tin ra thành các tập tin khai báo và định nghĩa, sau đó kết hợp lại trong tập tin chứa hàm main.

2 Hàm trùng tên

Trong ngôn ngữ C, ta có thể khai báo và định nghĩa nhiều hàm trùng tên (nhưng phải khác về kiểu trả về hoặc danh sách tham số đầu vào). Trong thực tế có nhiều tình huống cần phải sử dụng hàm trùng tên. Ví dụ xét ngữ cảnh tính độ lớn của một vector. Ta biết rằng có thể vector một chiều, hai chiều hay ba chiều…Như vậy trong các hệ tọa độ khác nhau thì cách tính sẽ khác nhau nhưng ý nghĩa vẫn là tính độ lớn vector. Ta sẽ xây dựng các hàm trùng tên như sau

Dòng	
1	#include <stdio.h>
2	#include <math.h>
3	
4	double DoLonVector(double);
5	double DoLonVector(double, double);
6	double DoLonVector(double, double, double);
7	
8	double DoLonVector(double x){
9	return fabs(x);
10	}
11	double DoLonVector(double x, double y){
12	return sqrt(x*x + y*y);
13	}
14	double DoLonVector(double x, double y, double z){
15	return sqrt(x*x + y*y + z*z);
16	}
17	
18	void main(){
19	double x = 2.1, y = -1.9, z = 3.7;
20	double kq1 = DoLonVector(x);
21	double kq2 = DoLonVector(x, y);
22	double kq3 = DoLonVector(x, y, z);
23	printf("kq1 = %lf", kq1);

```
24    printf("kq2 = %lf", kq2);
25    printf("kq3 = %lf", kq3);
26  }
```

Trong ví dụ này ta thấy có ba hàm trùng tên 'DoLonVector', chỉ khác nhau danh sách tham số đầu vào. Nếu ngôn ngữ C không hỗ trợ hàm trùng tên thì ta cần ba hàm với ba tên khác nhau và khi gọi sử dụng trong hàm main ta cần phải nhớ nhiều tên hàm hơn gây bất tiện khi sử dụng hàm.

3 Hàm có tham số mặc định

Tính năng này của ngôn ngữ cũng giúp hỗ trợ đoạn mã chúng ta trở nên tinh gọn hơn. Xét ngữ cảnh làm tròn số thực của các ví dụ trước. Trong ngữ cảnh này ta sẽ có hai hàm làm tròn số thực, một hàm làm tròn không lấy số sau dấu thập phân và một hàm làm tròn tới các số sau dấu chấm thập phân.

```
Dòng
1   #include <stdio.h>
2   #include <math.h>
3
4   double LamTron(double, int n = 0);
5
6   double LamTron(double x, int n){
7     double kq, tile = pow(10, n);
8     x *= tile;
9     if(x > 0) kq = floor(x + 0.5)/tile;
10    else kq = -floor(-x + 0.5)/tile;
11    return kq;
12  }
13
14  void main(){
15    double x = 2.71828;
16    double y = LamTron(x);
17    printf("Ket qua sau khi lam tron: %lf\n", y);
18    y = LamTron(x, 2);
19    printf("Ket qua sau khi lam tron: %lf\n", y);
20  }
```

Trong ví dụ này ta thấy mặc dù việc gọi hàm ở dòng 16 và 18 có vẻ khác nhau. Tuy nhiên thực chất nó gọi tới cùng một hàm. Như vậy nếu không hỗ trợ tính năng này ta cần khai báo tới hai hàm cùng tên là 'LamTron', trong khi việc tính toán hầu như không có sự khác biệt.

4 Hàm có tham số là hàm

Giả sử ta có yêu cầu là in danh sách các số ≤ n cho trước (n do người dùng nhập vào) thỏa mãn điều kiện nào đó. Ví dụ viết hàm in các số nguyên tố ≤ n cho trước hay in các số chẵn ≤ n cho trước… Ta thấy việc in danh sách các số là công việc như nhau cho dù là chẵn lẻ hay nguyên tố. Tuy nhiên việc kiểm tra điều kiện là khác nhau cho nguyên tố, chẵn hay lẻ. Như vậy ta tách việc kiểm tra ra thành các hàm riêng lẻ sau đó kết hợp vào trong hàm in danh sách các số.

Dòng	
1	#include <stdio.h>
2	
3	bool KiemTraSoChan(int);
4	bool KiemTraSoNguyenTo(int);
5	void InDanhSach(bool F(int x), int n);
6	
7	bool KiemTraSoChan(int n){
8	if(n % 2 == 0) return true;
9	return false;
10	}
11	bool KiemTraSoNguyenTo(int n){
12	if(n == 0 || n == 1) return false;
13	for(int i = 2; i < n; i++){
14	if(n % i == 0) return false;
15	}
16	return true;
17	}
18	
19	void InDanhSach(bool F(int x), int n){
20	for(int i = 0; i < n; i++){
21	if(F(i) == true){
22	printf("%d ", i);
23	}
24	}
25	}
26	
27	void main(){
28	int n;
29	printf("Nhap n: ");
30	scanf("%d", &n);
31	InDanhSach(KiemTraSoChan, n);
32	printf("\n");

33	InDanhSach(KiemTraSoNguyenTo, n);
34	}

Trong đoạn mã trên, ta thấy hàm InDanhSach đã làm đúng vai trò của nó đó là gọi hàm printf để in. Nhiệm vụ kiểm tra điều kiện sẽ được tách ra cho các hàm con. Như vậy cùng hàm InDanhSach nhưng có thể sử dụng cho nhiều mục đích khác nhau.

5 Tổ chức hàm với nhiều tập tin mã nguồn

Ta đã được làm quen với việc tách một chương trình lớn ra thành nhiều tác vụ con (hàm con). Ví dụ như việc tính tổ hợp C_n^k, cần hàm con tính giai thừa. Giờ ta tìm hiểu cách tách biệt tập tin khai báo hàm ra khỏi tập tin định nghĩa hàm. Trong C++, tập tin khai báo hàm có phần mở rộng là '.h' và tập tin định nghĩa hàm có phần mở rộng là '.cpp'. Việc tách biệt này giúp đoạn mã linh hoạt cũng như có thể đóng gói thành các module riêng lẻ cho nhiều dự án cùng sử dụng.

Xét ngữ cảnh cần viết một module giải các phương trình cơ bản: bậc nhất một ẩn, bậc hai một ẩn, bậc bốn trùng phương một ẩn, bậc ba một ẩn… Ta sẽ thực hiện khai báo các hàm cần thiết trong một tập tin '.h', ví dụ đặt tên là 'Equations.h'

Dòng	
1	//Equations.h
2	#ifndef _EQUATIONS_H_
3	#define _EQUATIONS_H_
4	const int NoSolution = 0, Undetermined = -1; // Cu phap khai bao hang so C++
5	int EquaDeg1(double, double, double&);//Bac nhat mot an
6	int EquaDeg2(double, double, double, double&, double&);//Bac hai mot an
7	//…Cac Phuong trinh khac
8	#endif

Tiếp theo ứng với tập tin 'Equations.h' sẽ có một tập tin tên 'Equations.cpp' để định nghĩa các hàm vừa khai báo.

Dòng	
1	//Equations.cpp
2	#include <math.h>
3	#include "Equations.h"
4	
5	int EquaDeg1(double a, double b, double& x){
6	//…
7	//return …
8	}
9	

```
10  int EquaDeg2(double a, double b, double c, double &x1, double &x2){
11    //…
12    //return …
13  }
14  //Dinh nghia cac ham khac
```

Tiếp theo giả sử ta có nhu cầu viết các hàm nhập/xuất riêng cho các loại phương trình, ta cũng cần có một cặp '.h/.cpp' như trên. Ta đặt tên là 'EquationIO.h' và 'EquationIO.cpp'.

```
Dòng
1  //EquationIO.h
2  #ifndef _EQUATIONIO_H_
3  #define _EQUATIONIO_H_
4  void PrintSolution(int, double, double, double, double);
5  void DisplayEquation(double, double);
6  void DisplayEquation(double, double, double, bool type4 = false);
7  void EquationInput(double&, double&);
8  void EquationInput(double&, double&, double&);
9  #endif
```

Khai báo hàm dòng số 4 có tham số đầu tiên là số lượng nghiệm cần in ra. Các tham số còn lại lần lượt là các biến chứa các nghiệm nếu có. Khai báo hàm dòng số 5 sẽ giúp ta in ra màn hình dòng 'ax + b = 0', trong đó a và b chính là các số mà người dùng nhập vào, tương tự là khai báo hàm dòng số 6 dành cho phương trình 'ax^2 + bx + c = 0' hay 'ax^4 + bx^2 + c = 0'. Hai khai báo hàm dòng 7 và 8 giúp nhập các biến a, b và c cho phương trình cần giải. Tiếp theo là tập tin 'EquationIO.cpp'

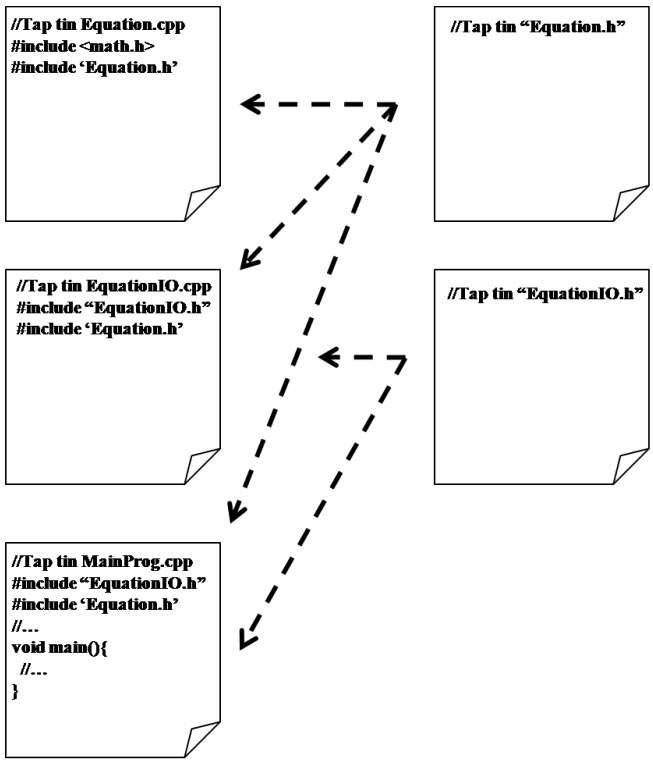
```
Dòng
1  //EquationIO.cpp
3  #include "EquationIO.h"
4  #include "Equation.h"
5  void PrintSolution(int nSol, double x1, double x2, double x3, double x4){
6    //…
7  }
8
9  void DisplayEquation(double a, double b){
10   printf("%lf x + %lf = 0", a, b);
11 }
12 //Dinh nghia cac ham khac…
```

Trong tập tin 'EquationIO.cpp' ta thấy có include 'Equations.h'. Như vậy khi tách các tập tin .h, thì các tập tin khác có thể sử dụng lại các tập tin .h này. Từ đó có thể tái sử dụng đoạn mã tránh lặp lại. Cuối cùng là tập tin MainProg.cpp ta sẽ lần lượt include các tập tin .h trên để sử dụng

Dòng	
1	//MainProg.cpp
2	#include “EquationIO.h”
3	#include “Equation.h”
4	
5	void main(){
6	double a, b, x;
7	EquationInput(a, b);
8	nSol = EquaDeg1(a, b, x);
9	PrintSolution(nSol, x, 0, 0, 0);
10	}

Lưu ý hàm PrintSolution với ba tham số cuối là 0 hàm ý là không tồn tại. Đoạn mã có thể sai nếu phương trình có nghiệm là zero. Đoạn mã mục tiêu chỉ muốn minh họa việc tách tập tin không có ý giải quyết trọn vẹn tính đúng đắn của việc giải phương trình. Nếu muốn xét cẩn thận tất cả các trường hợp ta cần thiết kế hàm PrintSolution theo một hướng khác.

Bên dưới là sơ đồ kết nối các tập tin trong chương trình giải phương trình của ta.



6 Bài tập

Sinh viên xây dựng tập tin gồm các hàm xử lý ngày tháng thực hiện các thao tác sau:

6.1 Kiểm tra năm nhuận

6.2 Trả ra ngày hôm sau với ngày tháng năm người dùng nhập vào

6.3 Trả ra ngày hôm trước ứng với ngày tháng năm người dùng nhập vào

6.4 Kiểm tra ngày tháng năm hợp lệ

6.5 Trả ra số ngày ứng với tháng và năm nhập vào

6.6 Trả ra khoảng cách (tính theo ngày) giữa hai khoảng thời gian người dùng nhập vào (người dùng vẫn nhập vào ngày tháng năm)

6.7 Trả ra khoảng cách (tính theo tháng) giữa hai khoảng thời gian người dùng nhập vào (người dùng vẫn nhập vào ngày tháng năm)

6.8 Trả ra khoảng cách (tính theo năm) giữa hai khoảng thời gian người dùng nhập vào (người dùng vẫn nhập vào ngày tháng năm)

Sau khi xây dựng xong, tự viết hàm main trong tập tin MainProg.cpp để minh họa các hàm vừa viết.