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

HƯỚNG DẪN THỰC HÀNH

TUẦN 03

CÂU LỆNH LẶP

TP.HCM, ngày 09 tháng 09 năm 2017

MỤC LỤC

1 Khái niệm

Trong thực tế, có nhiều vấn đề đòi hỏi việc lặp đi lặp lại để tính toán một kết quả nào đó. Trong ngôn ngữ lập trình có các cấu trúc lặp để hỗ trợ bao gồm: while, do-while và for. Ta sẽ lần lượt đi qua các ví dụ minh họa cách sử dụng các vòng lặp này.

2 Vòng lặp while

Phần này trình bày về vòng lặp while. Bên dưới là cấu trúc chung của vòng lặp này:

```
while(<điều kiện lặp>)
{
  //Khối lệnh A
}
//Các lệnh kế tiếp…
```

Tiếp theo ta xét ví dụ tính S(n) = 1 + 2 + … + n. Ta thấy tùy thuộc vào giá trị n mà S(n) sẽ có các giá trị khác nhau. Bên dưới là minh họa việc dùng vòng lặp while để tính S(n).

Dòng	
1	#include <stdio.h>
2	void main(){
3	int n, i = 1, S = 0;
4	printf("Nhap n: ");
5	scanf("%d", &n);
6	while(i <= n)
7	{
8	S = S + i;
9	i = i + 1;
10	}
11	printf("S co gia tri: %d\n", S);
12	}

Trong ví dụ này ta thấy ngoài biến n và S, ta cần một biến chạy *i*. Trong khối lệnh while ta cần tăng biến *i* này lên một đơn vị

3 Vòng lặp do-while

Trong phần này, ta sẽ tìm hiểu cấu trúc lặp do-while. Khác với while, do-while sẽ thực hiện câu lệnh trước sau đó mới kiểm tra điều kiện. Tính chất này sẽ thuận lợi trong việc xử lý một số bài toán mà đòi hỏi việc thực hiện trước khi kiểm tra điều kiện.

```
do
```

```
{
  //Khối lệnh A
}while(<điều kiện lặp>);
//Các lệnh kế tiếp…
```

Xét ví dụ tìm k sao cho n = 1 + 2 + … + k. Dĩ nhiên có trường hợp không tồn tại k thỏa điều kiện trên. Tuy nhiên ta cũng có thể tìm k lớn nhất có thể sao cho 1 + 2 + … + k ≤ n.

Dòng	
1	#include <stdio.h>
2	
3	void main(){
4	int sum = 0, k = 1;
5	do{
6	sum += k;
7	k++;
8	}while(sum < n);
9	k--;
10	if(sum > n){
11	sum -= k;
12	k--;
13	}
14	}

Trong đoạn mã trên, ta lưu ý câu lệnh số 9. Trong câu lệnh này, ta cần giảm k vì khi ra khỏi vòng lặp do-while, có nghĩa là sum ≥ n. Vì vậy ta phải giảm k đi một đơn vị để **quay về giá trị** k thực sự mà sum đã cộng dồn vào trước đó. Câu lệnh if sau đó để kiểm tra trường hợp sum > n (trường hợp sum = n thì ta không cần xét vì thỏa mãn yêu cầu). Ta cần giảm sum đi một lượng k và giảm k đi một đơn vị để lấy giá trị k tương ứng.

4 Vòng lặp for

Trong phần này ta sẽ làm quen với cấu trúc for. Có nhiều cách thể hiện cấu trúc này, tuy nhiên để đơn giản ta sẽ sử dụng cấu trúc thông dụng nhất.

```
//Khối lệnh A
for(<biến chạy>;<điều kiện lặp>;<tăng biến chạy>)
{
  //Khối lệnh B
}
//Các lệnh kế tiếp…
```

Ta xét ví dụ tính S(n) lúc nãy bằng cấu trúc lặp for:

Dòng	
1	#include <stdio.h>
2	void main(){
3	int n, S = 0;
4	printf(“Nhap n: ”);
5	scanf(“%d”, &n);
6	for(int i = 0; i <= n; i++)
7	{
8	S = S + i;
9	}
10	printf(“S co gia tri: %d\n”, S);
11	}

Trong đoạn mã trên, tại dòng lệnh số 6, biến i sẽ được định nghĩa lúc đầu là 0, sau đó i sẽ được so sánh với n, nếu thỏa điều kiện thì câu lệnh số 8 sẽ thực hiện. Sau khi làm xong câu lệnh số 8, thì biến i sẽ tăng lên một đơn vị, sau đó tiếp tục so sánh điều kiện ‘i ≤ n’. Nếu thỏa điều kiện thì câu lệnh số 8 lại tiếp tục được thực hiện, ngược lại thì sẽ thoát ra khỏi vòng lặp và đi tới câu lệnh số 10.

5 Câu lệnh break

Ta dùng câu lệnh break trong trường hợp muốn thoát ra khỏi vòng lặp ngay tức thì, cho dù điều kiện của vòng lặp vẫn thỏa. Giả sử ta xét ví dụ sau, ta sẽ cho người dùng nhập giá trị số thực và lần lượt cộng dồn vào các giá trị đó. Nhưng khi người dùng nhập số 0 thì ta sẽ thoát chương trình và không cộng dồn giá trị 0.

Dòng	
1	#include <stdio.h>
2	
3	void main(){
4	float s = 0, x;
5	while(1){
6	printf(“Nhap gia tri can them vao s: ”);
7	scanf(“%f”, &x);
8	if(x == 0)
9	break;
10	s = s + x;
11	}
16	printf(“Gia tri s = %f\n”, s);
17	}

Ở ví dụ này ta thấy việc dừng chương trình hoàn toàn phụ thuộc vào người dùng.

6 Câu lệnh continue

Câu lệnh continue được dùng để bỏ qua một số lần lặp nào đó. Giả sử ta cần tính S(n) = 1 + 3 + 5 + … + k, với điều kiện k lẻ và k ≤ n. Ta sẽ dùng câu lệnh này để bỏ qua trường hợp biến chạy i mang giá trị chẵn.

Dòng	
1	#include <stdio.h>
2	void main(){
3	int n, S = 0;
4	printf("Nhap n: ");
5	scanf("%d", &n);
6	for(int i = 0; i <= n; i++)
7	{
8	if(i % 2 == 0) continue;
9	S = S + i;
10	}
11	printf("S co gia tri: %d\n", S);
12	}

7 Bài tập

Sinh viên làm các bài tập sau:

7.1 Cấu trúc For

7.1.1 Viết chương trình tính n! với n nguyên dương nhập từ bàn phím.

7.1.2 Viết chương trình tính S = $1^2 + 2^2 + 3^2 + \ldots + n^2$ với n nguyên dương nhập từ bàn phím

7.1.3 Viết chương trình tính S = 1/2 + 2/3 + 3/4 + 4/5 + 5/6 + n/(n+1) với n nhập từ bàn phím

7.1.4 Viết chương trình tính S = 1 + (1*2)/(1+2)+ (1*2*3)/(1+2+3) + … + (1*2*…*n)/(1+2+…+n) với n nhập từ bàn phím

7.1.5 Viết chương trình kiểm tra một số nguyên n có phải số nguyên tố.

7.2 Cấu trúc Do-while

7.2.1 Viết chương trình nhập vào một số nguyên không âm. Xuất ra số nguyên đó ra màn hình.

7.2.2 Viết chương trình nhập vào một năm (có kiểm tra điều kiện năm phải nhỏ hơn năm hiện tại). Xuất ra màn hình năm vừa nhập

7.2.3 Viết chương trình nhập vào một ngày sinh (có kiểm tra tính hợp lệ của ngày nhập). Xuất ra màn hình ngày mới nhập.

7.3 Cấu trúc while

7.3.1 Viết chương trình kiểm tra số n có phải là số chính phương hay không

7.3.2 Viết chương trình kiểm tra một số có phải dạng 3^k hay không.

7.3.3 Viết chương trình kiểm tra một số có phải là số hoàn thiện hay không. Số hoàn thiện là số có tổng các số nguyên dương nhỏ hơn nó bằng nó.