Mục lục

Bài 5: CẤU TRÚC LẶP (tiếp)

Mục tiêu cần đạt được của bài dạy về kiến thức, kỹ năng

Về kiến thức:

+ Hiểu được khái niệm về cấu trúc lặp trong ngôn ngữ lập trình C.

Về kỹ năng:

+ Vận dụng các kiến thức về cấu trúc lặp để cài đặt các bài toán ứng dụng.

4.1. Cấu trúc for

Toán tử for dùng để xây dựng cấu trúc lặp có cú pháp như sau:

```
for (biểu thức 1; biểu thức 2; biểu thức 3)
        Lệnh hoặc khối lệnh;
```

Toán tử for gồm ba biểu thức và thân for. Thân for là một câu lệnh hoặc một khối lệnh viết sau từ khoá for. Bất kỳ biểu thức nào trong ba biểu thức trên có thể vắng mặt nhưng phải giữ dấu chấm phẩy (;).

Thông thường biểu thức 1 là toán tử gán để tạo giá trị ban đầu cho biến điều khiển, biểu thức 2 là một quan hệ logic biểu thị điều kiện để tiếp tục chu trình, biểu thức ba là một toán tử gán dùng để thay đổi giá trị biến điều khiển.

Hoạt động của toán tử for:

Toán tử for hoạt động theo các bước sau:

- Xác định biểu thức 1

- Xác định biểu thức 2

Tuỳ thuộc vào tính đúng sai của biểu thức 2 để máy lựa chọn một trong hai nhánh:

- Nếu biểu thức hai có giá trị 0 (sai), máy sẽ ra khỏi for và chuyển tới câu lệnh sau thân for.

- Nếu biểu thức hai có giá trị khác 0 (đúng), máy sẽ thực hiện các câu lệnh trong thân for.

- Tính biểu thức 3, sau đó quay lại bước 2 để bắt đầu một vòng mới của chu trình.

Chú ý: Nếu biểu thức 2 vắng mặt thì nó luôn được xem là đúng. Trong trường hợp này việc ra khỏi chu trình for cần phải được thực hiện nhờ các lệnh break, goto hoặc return viết trong thân chu trình.

Trong dấu ngoặc tròn sau từ khoá for gồm ba biểu thức phân cách nhau bởi dấu; . Trong mỗi biểu thức không những có thể viết một biểu thức mà có quyền viết một dãy biểu thức phân cách nhau bởi dấu phảy. Khi đó các biểu thức trong mỗi phần được xác định từ trái sang phải. Tính đúng sai của dãy biểu thức được tính là tính đúng sai của biểu thức cuối cùng trong dãy này.

Trong thân của for ta có thể dùng thêm các toán tử for khác, vì thế ta có thể xây dựng các toán tử for lồng nhau.

Khi gặp câu lệnh break trong thân for, máy ra sẽ ra khỏi toán tử for sâu nhất chứa câu lệnh này. Trong thân for cũng có thể sử dụng toán tử goto để nhảy đến một ví trí mong muốn bất kỳ.

Ví dụ 4.1: Nhập một dãy số rồi đảo ngược thứ tự của nó.

```
#include <stdio.h>

float x[] = {1.3, 2.5, 7.98, 56.9, 7.23};
int n = sizeof(x)/sizeof(float);

int main(){

	int i, j;
	float c;

	for (i = 0, j = n - 1; i < j; ++i, --j){
		c = x[i];
		x[i] = x[j];
		x[j] = c;
	}

	printf("\n Day so dao la \n \n ");

	for (i = 0; i < n; ++i)
		printf("%f", x[i]);

	return 0;
}
```

Ví dụ 4.2: Tính tích hai ma trận có kích thước M x N và N x P.

```
#include <stdio.h>

float x[3][2], y[2][4], z[3][4], c;

int main(){

	int i, j;
	printf("\n nhap gia tri cho ma tran X ");

	for (i = 0; i <= 2; ++i)
		for (j = 0; j <= 1; ++j){
			printf("\n x[%d][%d] = ", i, j);
			scanf("%f", &c);
			x[i][j] = c;
		}

	printf("\n nhap gia tri cho ma tran Y ");

	for (i = 0; i <= 1; ++i)
		for (j = 0; j <= 3; ++j){
			printf("\n y[%d][%d] = ", i, j);
			scanf("%f", &c);
			y[i][j] = c;
		}
	return 0;
}
```

4.2. Cấu trúc lặp while

Toán tử while dùng để xây dựng chu trình lặp dạng:

```
while (biểu thức)
		Lệnh hoặc khối lệnh;
```

Như vậy toán tử while gồm một biểu thức và thân chu trình. Thân chu trình có thể là một lệnh hoặc một khối lệnh.

Hoạt động của chu trình như sau:

Máy xác định giá trị của biểu thức, tuỳ thuộc giá trị của nó máy sẽ chọn cách thực hiện như sau:

Nếu biểu thức có giá trị 0 (biểu thức sai), máy sẽ ra khỏi chu trình và chuyển tới thực hiện câu lệnh tiếp sau chu trình trong chương trình.

Nếu biểu thức có giá trị khác 0 (biểu thức đúng), máy sẽ thực hiện lệnh hoặc khối lệnh trong thân của while. Khi máy thực hiện xong khối lệnh này nó lại thực hiện xác định lại giá trị biểu thức rồi làm tiếp các bước như trên.

Chú ý:

Trong các dấu ngoặc () sau while chẳng những có thể đặt một biểu thức mà còn có thể đặt một dãy biểu thức phân cách nhau bởi dấu phảy. Tính đúng sai của dãy biểu thức được hiểu là tính đúng sai của biểu thức cuối cùng trong dãy.

Bên trong thân của một toán tử while lại có thể sử dụng các toán tử while khác. bằng cách đó ta đi xây dựng được các chu trình lồng nhau.

Khi gặp câu lệnh break trong thân while, máy sẽ ra khỏi toán tử while sâu nhất chứa câu lệnh này.

Trong thân while có thể sử dụng toán tử goto để nhảy ra khỏi chu trình đến một vị trí mong muốn bất kỳ. Ta cũng có thể sử dụng toán tử return trong thân while để ra khỏi một hàm nào đó.

Ví dụ 4.3: Chương trình tính tích vô hướng của hai véc tơ x và y:

```
#include <stdio.h>

float x[] = {2, 3.4, 4.6, 21};
float y[] = {24, 12.3, 56.8, 32.9};

int main(){

	float s = 0;
	int i = -1;
	while (++i < 4)
		s += x[i] * y[i];
	printf("\n Tich vo huong hai vec to x va y:%f", s);
	return 0;
}
```

4.3. Cấu trúc lặp do-while

Khác với các toán tử while và for, việc kiểm tra điều kiện kết thúc đặt ở đầu chu trình, trong chu trình do while việc kiểm tra điều kiện kết thúc đặt cuối chu trình. Như vậy thân của chu trình bao giờ cũng được thực hiện ít nhất một lần.

Chu trình do while có dạng sau:

```
do
        Lệnh hoặc khối lệnh;
    while (biểu thức);
```

Lệnh hoặc khối lệnh là thân của chu trình có thể là một lệnh riêng lẻ hoặc là một khối lệnh.

Hoạt động của chu trình như sau:

Máy thực hiện các lệnh trong thân chu trình.

Khi thực hiện xong tất cả các lệnh trong thân của chu trình, máy sẽ xác định giá trị của biểu thức sau từ khoá while rồi quyết định thực hiện như sau:

Nếu biểu thức đúng (khác 0) máy sẽ thực hiện lặp lại khối lệnh của chu trình lần thứ hai rồi thực hiện kiểm tra lại biểu thức như trên.

Nếu biểu thức sai (bằng 0) máy sẽ kết thúc chu trình và chuyển tới thực hiện lệnh đứng sau toán tử while.

Chú ý: Những điều lưu ý với toán tử while ở trên hoàn toàn đúng với do while.

Ví dụ 4.4: Đoạn chương trình xác định phần tử âm đầu tiên trong các phần tử của mảng x.

```
#include <stdio.h>

float x[5], c;
int main(){

    int i = 0, n = 4;
    printf("\n nhap gia tri cho ma tran x ");

    for (i = 0; i <= n; ++i){
        printf("\n x[%d] = ", i);
        scanf("%f", &c);
        x[i] = c;
```

```
	}

	do
		++i;
	while (x[i] >= 0 && i <= n);

	if (i <= n)
		printf("\n PT am dau tien = x[%d] = %f", i, x[i]);
	else
		printf("\n Mang khong có phan tu am ");

	return 0;
}
```

4.4. Câu lệnh break, continue

Câu lệnh break

Câu lệnh break cho phép ra khỏi các chu trình với các toán tử for, while và switch. Khi có nhiều chu trình lồng nhau, câu lệnh break sẽ đưa máy ra khỏi chu trình bên trong nhất chứa nó không cần điều kiện gì. Mọi câu lệnh break có thể thay bằng câu lệnh goto với nhãn thích hợp.

Ví dụ 4.5: Biết số nguyên dương n sẽ là số nguyên tố nếu nó không chia hết cho các số nguyên trong khoảng từ 2 đến căn bậc hai của n. Viết đoạn chương trình đọc vào số nguyên dương n, xem n có là số nguyên tố.

```
#include <stdio.h>
#include <math.h>

unsigned int n;

int main(){

	int i, nt = 1;
	printf("\n cho n = ");
	scanf("%d", &n);
	for (i = 2; i <= sqrt(n); ++i)
		if ((n % i) == 0){
			nt = 0;
			break;
		}
	if (nt)
```

```
            printf("\n %d la so nguyen to", n);
        else
            printf("\n %d khong la so nguyen to", n);

        return 0;
}
```

4.5. Câu lệnh continue

Trái với câu lệnh break, lệnh continue dùng để bắt đầu một vòng mới của chu trình chứa nó. Trong while và do while, lệnh continue chuyển điều khiển về thực hiện ngay phần kiểm tra, còn trong for điều khiển được chuyển về bước khởi đầu lại (tức là bước: tính biểu thức 3, sau đó quay lại bước 2 để bắt đầu một vòng mới của chu trình).

Chú ý: Lệnh continue chỉ áp dụng cho chu trình chứ không áp dụng cho switch.

Ví dụ 4.6: Viết chương trình in ra các giá trị từ 1 đến 10 (không in giá trị 4)

```
#include <stdio.h>
int main()
{
  for (int j = 0; j <= 10; j++)
  {
    if (j==4)
    {
        // Bỏ qua các câu lệnh cuối vòng lặp nếu j = 4
        continue;
     }

     printf("%d ", j);
  }
  return 0;
}
```

Bài tập cuối chương

Bài 1. Nhập vào hai số nguyên a, b. In ra màn hình giá trị lớn nhất

Bài 2. Viết chương trình nhập vào tháng, in ra tháng đó có bao nhiêu ngày.

Bài 3. Viết chương trình nhập vào số nguyên dương n. Kiểm tra xem n có phải là số chính phương hay không? (số chính phương là số khi lấy căn bậc 2 có kết quả là nguyên)

Bài 4. Viết chương trình xét xem một tam giác có là tam giác đều hay không khi biết chiều dài ba cạnh của tam giác.

Hướng dẫn: Nhập ba cạnh của tam giác vào ba biến a, b, c. Nếu a = b và b = c thì tam giác là tam giác đều và ngược lại tam giác không là tam giác đều

Bài 5. Viết chương trình tính tổng S, với n nguyên dương được nhập vào từ bàn phím: $S = 1 + 2+, \dots, N$

Bài 6. In ra màn hình giá trị từ 10 đến 20 trừ đi số 13 và số 17, sử dụng câu lệnh continue

Bài 7. Viết chương trình nhập số nguyên dương n. Liệt kê n số nguyên tố đầu tiên.

Bài 8. Viết chương trình nhập vào hai số nguyên dương a và b. Tìm ước số chung lớn nhất và bội số chung nhỏ nhất của a và b.

Bài 9. Viết chương trình nhập vào một số nguyên n gồm tối đa 10 chữ số (4 bytes). In ra màn hình giá trị nhị phân của số trên.

Hướng dẫn: chia lấy dư cho 2 và xuất theo thứ tự ngược lại.

Bài 10. Viết chương trình đếm số ước số của số nguyên dương N, sử dụng for; while; do ... while

Ví dụ: N=12 số ước số của 12 là 6

Bài 11. Viết chương trình in ra các số lẻ nhỏ hơn hoặc bằng số nguyên dương n (Với n được nhập). Yêu cầu nhập lại nếu n <= 0; sử dụng for; while; do ... while

Hướng dẫn: Sử dụng kiến thức số lẻ đầu tiên bằng 1. Số lẻ sau bằng số trước cộng với 2. Cho biến i có giá trị ban đầu bằng 1. Dùng vòng lặp while do với điều kiện i <n và công việc bên trong là in i và tăng i lên 2.

Bài 12. Viết chương trình tính n! với n! được định nghĩa như sau:

- n! = 1 với n = 0

- n! = 1.2.3...n (Tích của n số từ 1 đến n).

Yêu cầu: Sử dụng vòng lặp với số lần chưa biết trước:

Hướng dẫn: Có thể viết lại: n! = n. (n-1)... 3.2.1. Lặp gt = gt*n; n = n-1 với điều kiện n > 0.

Bài 13. Viết chương trình cho phép tính tổng của nhiều số (Chưa biết bao nhiêu số). Nhập số 0 để kết thúc quá trình nhập (sử dụng for; while; do ... while)

Bài 14. Viết chương trình tìm các số nguyên gồm 3 chữ số sao cho tích của 3 chữ số bằng tổng 3 chữ số. Ví dụ: 1*2*3 = 1+2+3

Tài liệu tham khảo

A. Tài liệu học tập chính

[1] Bộ môn Khoa học máy tính (2024), Bài giảng Kỹ thuật lập trình, Khoa Công nghệ thông tin - Trường Đại học Công nghệ Thông tin và Truyền thông.

[2] Sazzad M.S. Imran, and Md Atiqur Rahman Ahad (2024), Learn Programming with C: An Easy Step-by-Step Self-Practice Book for Learning C, 1st Edition, CRC Press.

B. Tài liệu tham khảo

[3] GS. Phạm Văn Ất, ThS. Đỗ Văn Tuấn, ThS. Nguyễn Hiếu Cường, Lê Trường Thông (2023), Kỹ thuật lập trình C cơ sở và nâng cao, Nhà xuất bản Bách Khoa Hà Nội.

[4] Slobodan Dmitrović (2024), Modern C for Absolute Beginners: A Friendly Introduction to the C Programming Language, 2nd Edition, Apress.

[5] Jeff Szuhay (2020), Learn C Programming: A beginner's guide to learning C programming the easy and disciplined way, 1st Edition, Packt Publishing.

C. Phần mềm (nếu có)

[6] Dev C++, URL: https://github.com/Embarcadero/Dev-Cpp/releases

[7] Visual Studio Code, URL: https://code.visualstudio.com/