4

CHƯƠNG TRÌNH CÓ YẾU TỐ LẶP LẠI

NỘI DUNG CHÍNH

Mục tiêu chính của chương này gồm có các nội dung sau

SỐ LẦN LẶP BIẾT TRƯỚC
Sử dụng cấu trúc **for**

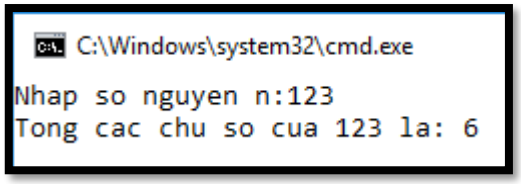
SỐ LẦN LẶP KHÔNG BIẾT TRƯỚC
Sử dụng cấu trúc **do**… **while** và cấu trúc **while**

Các mã nguồn được trình bày trong sách có thể được tải tại địa chỉ http://goo.gl/PWZhME

4.1. VÍ DỤ MẪU ÔN LẠI KIẾN THỨC

Lặp với số lần lặp không biết trước

Nhập vào một số nguyên. In ra màn hình tổng các chữ số của n.



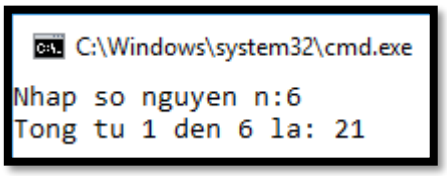
Mã nguồn

C	C++

#**include** "stdio.h"
#**include** "conio.h"
 	
void main()
{
int n;
printf("Nhap so nguyen n:");
scanf_s("%d", &n);
 	
int sum = 0;
int temp = n;
while (temp != 0)
{
int r = temp % 10;
sum += r;
temp /= 10;
}
 	
printf("Tong cac chu so la: %d", sum);
 	
_getch();
}
	#include <iostream>
	using namespace std;
	
	void main()
	{
	int n;
	cout << "Nhap so nguyen n:";
	cin >> n;
	
	int sum = 0;
	int temp = n;
	while (temp != 0)
	{
	int r = temp % 10;
	sum += r;
	temp /= 10;
	}
	
	cout << "Tong cac chu so la: " << sum;
	cin.get();
	}

Lặp với số lần lặp biết trước – Yêu cầu

Nhập vào một số nguyên n. In ra màn hình tổng 1 + 2 + ... + n



Mã nguồn

C	C++

#**include** "stdio.h"
#**include** "conio.h"
 	
void main()
{
int n;
printf("Nhap so nguyen n:");
scanf_s("%d", &n);
 	
int sum = 0;
for (**int** i = 1; i <= n; i++)
{
sum += i;
}
 	
printf("Tong tu 1 den %d la: %d", n, sum);
_getch();
}
	#**include** <iostream>
	using namespace std;
	
	void main()
	{
	int n;
	cout << "Nhap so nguyen n:";
	cin >> n;
	
	int sum = 0;
	for (**int** i = 1; i <= n; i++)
	{
	sum += i;
	}
	
	cout << "Tong tu 1 den " << n << " la: " << sum;
	cin.get();
	}

4.2. CÁC BÀI TẬP ÔN LẠI KIẾN THỨC

1. Chỉ thị break

Yêu cầu: Nhập vào một số nguyên n. Kiểm tra số này có phải chỉ toàn các chữ số chẵn thôi không.

Mã nguồn gợi ý:

```
bool allIsEven = true;
int temp = n; // Sao chép giá trị n tránh thao tác trực tiếp

do {
    int r = temp % 10; // Lấy chữ số cuối cùng
    if (r % 2 == 1)    // Là số lẻ
    {
        allIsEven = false; // Hạ cờ
        break;  // Chỉ cần có 1 chữ số lẻ là đủ để thoát khỏi vòng lặp
    }
    temp /= 10; // Bỏ đi chữ số cuối cùng
} while (temp != 0);

// Ra khỏi vòng lặp cờ allIsEven sẽ cho biết kết quả
```

2. Chỉ thị continue.

Yêu cầu: Nhập vào n số nguyên, chỉ tính. Tính tổng các chữ số lẻ của n.

Mã nguồn gợi ý:

```
int sumOdd = 0;
int temp = n; // Sao chép giá trị n tránh thao tác trực tiếp

do {
    int  r = temp % 10; // Lấy chữ số cuối cùng
    if (r % 2 == 1) {   // Là số lẻ
        sumOdd += r;
    }
    else continue;
} while (temp != 0);
```

3. Giải thuật tìm ước số chung lớn nhất của Euclide

Yêu cầu: Nhập vào hai số nguyên dương a và b, tìm ước chung lớn nhất của a và b.

Nguyên tắc: nếu x là ước số chung lớn nhất của a và b thì nó cũng là ước của (a-b) (giả sử a > b).

Mã nguồn gợi ý 1:

```
while (b != 0)
{
        if (a > b) a -= b;
        else b -= a;
}
// Ra khỏi vòng lặp a chính là ước chung lớn nhất
```

Mã nguồn gợi ý 2:

```
while (b != 0)
{
   t = b;
   b = a % b;
   a = t;
 }
// Ra khỏi vòng lặp a chính là ước chung lớn nhất
```

4. while (true)

Yêu cầu: Nhập liên tục số nguyên cho đến khi gặp số âm thì dừng. In ra màn hình tổng các số đã nhập

Mã nguồn gợi ý

```
while (true) {
   // Nhập n
   if (n < 0)
      break;
  else
     sum += n;
}
```

4.3. BÀI TẬP VẬN DỤNG

1. Viết chương trình nhập vào các số nguyên, dừng lại khi nhập số 0. Cho biết có bao nhiêu số chẵn, bao nhiêu số lẻ, bao nhiêu số âm, bao nhiêu số dương.

2. Nhập vào số nguyên dương n.

a. Cho biết n có phải là số nguyên tố (prime number) hay không. (Số nguyên tố được định nghĩa là số > 1, chỉ có 2 ước là 1 và chính nó)

b. Cho biết n có phải là số hoàn hảo (perfect number) hay không. Số hoàn hảo là số có tổng các ước nhỏ hơn bằng chính nó.

Ví dụ: **6** có 3 ước là 1, 2 và 3 có tổng là 6 => 6 là số hoàn hảo.

28 có các ước 1, 2, 4, 7, và 14 có tổng là 28 => 28 là số hoàn hảo.

c. Cho biết n có phải là số chính phương (perfect square) hay không.

+ Giải pháp dễ nhất: dùng sqrt(n), ép kiểu về int để có i, kiểm tra i*i có = n hay không.

+ Giải pháp rèn tư duy vòng lặp: i chạy từ 1 đến n / 2, nếu i * i = n thì là số chính phương.

+ Thủ tục đúng theo toán học:

Số chính phương được tạo thành bởi tổng các số lẻ liên tiếp

- CP1: 1 = 1
- CP2: 4 = 1 + 3
- CP3: 9 = 1 + 3 + 5
- CP4: 16 = 1 + 3 + 5 + 7

Vậy nếu trừ các số lẻ liên tiếp, nếu = 0 thì là số chính phương, nếu âm thì là số thường.

d. Cho biết n có phải số đối xứng hay không.

Ví dụ: 2442, 13531 là số đối xứng.

e. Xuất ra màn hình số đảo ngược của n.

Ví dụ: Nhập 5413, xuất ra màn hình số đảo là 3145.

3. Nhập vào n số nguyên. Cho biết số nhỏ nhất, số lớn nhất đã nhập, tổng tất cả các số, tích các số lẻ.

4. Nhập vào số nguyên n, k tính chỉnh hợp và tổ hợp.

$$A_n^k = \frac{n!}{(n-k)!}$$

Ví dụ:

Input	Output
K=3 N=5	60

$$C_n^k = \frac{n!}{k!(n-k)!}$$

Ví dụ:

Input	Output
K=3 C=5	10

5. **Các bài toán tính tổng**

Tính các tổng với $n \in N*$

a. $A=1+2+3+4+...+(n-1)+n$

b. $B=1^2+2^2+3^2+...+(n-1)^2+n^2$

c. $C=1^3+2^3+3^3+4^3+...+(n-1)^3+n^3$

d. $D=1.2+2.3+3.4+...+(n-2)(n-1)+(n-1)n$

e. $E=1.2.3+2.3.4+3.4.5+...+(n-3)(n-2)(n-1)+(n-2)(n-1)n$

f. $F=2+4+6+...+(2n-4)+(2n-2)+2n$

g. $G=1+3+5+...+(2n-3)+(2n-1)$ với $(n \geq 2)$

6. Các bài vẽ hình

a. Viết chương trình nhập vào số dòng và số cột, xuất ra màn hình hình chữ nhật đặc có n dòng và m cột.

b. Viết chương trình nhập vào số dòng và số cột, xuất ra màn hình hình chữ nhật rỗng có n dòng và m cột.

c. Viết chương trình nhập vào số dòng, xuất ra màn hình hình tam giác cân đặc và tam giác cân rỗng

2.5.1. Chọn đáp án đúng nhất

1.

2.

3.

2.5.2. Lựa chọn từ để điền vào chỗ trống còn thiếu

a. [1

a. errors

b. programming language

c. source code

d. hello world

e. compiling

-- END --