TIN HỌC ĐẠI CƯƠNG

Bài 4: Vòng lặp trong C++ (phần 1)

Nội dung chính

1. Tại sao cần viết chương trình con?
2. Vòng lặp
3. Ví dụ về vòng lặp dùng biến đếm
4. Ví dụ về vòng lặp dùng điều kiện
5. Biểu thức logic
6. Bài tập

Phần 1

Tại sao cần viết chương trình con?

Chương trình đơn giản

Yêu cầu: nhập số n và tính $\sqrt[2]{n}$, không dùng hàm có sẵn

```
#include <iostream>      // khai báo thư viện

using namespace std;     // khai báo tên miền chuẩn

int main() {             // bắt đầu hàm chính
    double n;            // biến để chứa số n
    cout << "N = ";      // in ra chuỗi "N = "
    cin >> n;            // nhập số và ghi vào n

    double x = 1;        // biến x (để chứa căn 2 của n)
```

Chương trình đơn giản

```
    x = (x + n/x) / 2;              // tính x
    x = (x + n/x) / 2;              // tính x
    x = (x + n/x) / 2;              // tính x
    x = (x + n/x) / 2;              // tính x
    x = (x + n/x) / 2;              // tính x
    x = (x + n/x) / 2;              // tính x
    x = (x + n/x) / 2;              // tính x
    x = (x + n/x) / 2;              // tính x
    x = (x + n/x) / 2;              // tính x

    cout << "SQRT(n) = " << x;     // in số x ra màn hình
}
```

Hạn chế của cách viết đơn giản

- Sự khó hiểu: chương trình gồm nhiều phần, mỗi phần có mục đích khác nhau, ta phải đọc kỹ phần ghi chú mới nắm được nội dung
  - Chẳng hạn như phần tính căn bậc 2 của n, nếu không có ghi chú thì khó có thể biết nó làm gì
- Sự cứng nhắc: chỉ tính được căn bậc 2 của biến n, nếu muốn tính căn bậc 2 của biến m thì phải viết lại từ đầu
  - Hai đoạn mã hầu như giống nhau, khác tên biến
  - Nếu muốn tính căn bậc 2 cho 100 biến thì sao?
  - Nếu lỡ viết sai sẽ phải sửa ở 100 chỗ giống nhau?

Tách thành các hàm

Yêu cầu: nhập số n và tính $\sqrt[2]{n}$, không dùng hàm có sẵn

```
#include <iostream>          // khai báo thư viện
using namespace std;         // khai báo tên miền chuẩn

double can2(double n) {      // tự định nghĩa hàm sqrt
    double x = 1;            // biến x (chứa căn 2 của n)
    x = (x + n/x) / 2;       // tính x
    ...
    x = (x + n/x) / 2;       // tính x
    return x;                // trả về kết quả tính được
}
```

Tách thành các hàm

```
int main() {                    // bắt đầu hàm chính
    double n;                   // biến để chứa số n
    cout << "N = ";             // in ra chuỗi "N = "
    cin >> n;                   // nhập số và ghi vào n

    // gọi hàm tính toán và in kết quả ra màn hình
    cout << "SQRT(n) = " << can2(n);
}
```

Nhận xét:

- Tên hàm tự nó cũng cung cấp thông tin về đoạn mã
- Không còn phụ thuộc vào tên biến, ta có thể gọi hàm can2 với bất kì biến nào mà ta cần
- Sửa sai ở một đoạn mã duy nhất

Phần 2

Vòng lặp

Vòng lặp

- Ba cấu trúc điều khiển cơ bản trong máy tính
  - Tuần tự ← Đã học trong bài trước
  - Lặp ← Chương 3 (bài này)
  - Lựa chọn ← Chương 4 (bài sau)
- Nhiều hành vi, thuật toán trong cuộc sống về bản chất đã có tính lặp
  - Đếm số học sinh trong lớp
  - Tập luyện thể thao
  - Tính tổng dãy số
  - Các phương pháp tính xấp xỉ
  - Các phương pháp thử sai

Vòng lặp

- Ví dụ ở phần 1 cho ta thấy việc tính căn bậc 2 bằng cách viết thật nhiều lệnh giống nhau

```
x = (x + n/x) / 2;
```

- Nhưng cách này có vẻ không ổn lắm!?
- Một số bài toán giản đơn có thể giải quyết bằng phương pháp tuần tự, tuy nhiên có nhiều bật cập nếu chỉ dùng tuần tự
  - Chương trình dài, nhàm chán, dễ nhầm lẫn
  - Không thể tổng quát hóa (viết bao nhiêu dòng giống nhau thì vừa?)

Vòng lặp

- Ví dụ khác: nhập điểm số và tính xem điểm trung bình của lớp K50.N12 môn Tin Đại Cương là bao nhiêu?
  - Khai báo 36 biến để lưu điểm của 36 sinh viên?
  - Viết 36 lệnh nhập dữ liệu?
  - Viết 36 lệnh cộng giá trị các biến với nhau?
- Cần phải có cách làm khác!!!
- Ngôn ngữ C/C++ có giải pháp khắc phục được các vấn đề này: các câu lệnh yêu cầu máy tính lặp lại một công việc cho đến khi đạt yêu cầu

Vòng lặp

- Hai kiểu lặp thông dụng trong cuộc sống
  - Lặp sử dụng điều kiện dừng
    - "Ăn cho đến khi no"
    - "Học cho đến khi thuộc"
    - Nhiều hành vi cuộc sống là lặp
  - Lặp sử dụng biến đếm
    - "Đếm số người trong một bàn tiệc"
    - "Chọn 10 bạn học giỏi nhất lớp"
    - Cũng một dạng điều kiện dừng đặc biệt
- Ứng với những kiểu lặp đó, C/C++ cung cấp các lệnh lặp while, do-while và for

Phần 3

Ví dụ về vòng lặp dùng biến đếm

Ví dụ 1

Yêu cầu: in ra màn hình các số từ 1 đến 100 mỗi số trên 1 dòng.

Cách làm: dùng số i làm biến đếm, cho i chạy từ 1 đến 100, mỗi lần chạy thì in i ra màn hình.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       for (int i = 1; i <= 100; i = i+1)
6           cout << i << endl;
7   }
```

Ví dụ 2

Yêu cầu (mở rộng của bài trước): in ra các số từ 1 đến n mỗi số trên 1 dòng.

Cách làm: nhập n, dùng i làm biến đếm, i chạy từ 1 đến n, mỗi lần chạy thì in i ra màn hình.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       int n;
6
7       cout << "N = "; cin >> n;
8
9       for (int i = 1; i <= n; i = i+1)
10          cout << i << endl;
11  }
```

Ví dụ 3

Yêu cầu: tính tổng các số từ 1 đến n

Cách làm: nhập n, cho biến i chạy từ 1 đến n, mỗi lần chạy cộng dồn i vào biến tong.

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int n, tong = 0;
6
7      cout << "N = "; cin >> n;
8
9      for (int i = 1; i <= n; i = i+1)
10
11         tong = tong + i;
12
13     cout << "Tong = " << tong << endl;
14 }
```

Ví dụ 4

Yêu cầu: nhập n và tính n!

Cách làm: nhập n cho biến i chạy từ 1 đến n, mỗi lần chạy nhân dồn i vào biến tich

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       int n, tich = 1;
6
7       cout << "N = "; cin >> n;
8
9       for (int i = 1; i <= n; i = i+1)
10          tich = tich * i;
11
12      cout << "n! = " << tich << endl;
13  }
```

Phần 4

Ví dụ về vòng lặp dùng điều kiện

Ví dụ: nhập liệu và tính tổng

```
// thực hiện liên tiếp việc nhập và tính tổng
// cộng dồn và biến tong, kết thúc lặp nếu nhập vào số 0

#include <iostream>
using namespace std;
int main() {
    int tong = 0, n;
    do {
        cout << "Nhap mot so: "; cin >> n;
        tong = tong + n;
    } while (n != 0);
    cout << "Tong cac so vua nhap = " << tong;
}
```

Phần 5

Biểu thức logic

Biểu thức logic

- Các biểu thức logic là cơ sở để xây dựng điều kiện dừng lặp
- Giá trị logic có 2 loại: false (sai) và true (đúng)
  - Số nguyên có thể dùng lẫn lộn với kiểu logic, trong đó giá trị 0 tương đương với false và ngược lại
- Các phép toán logic:
  - Phép NOT (phép "đảo" - !)
  - Phép AND (phép "và" - &&)
  - Phép OR (phép "hoặc" - ||)
  - Phép XOR (phép "hoặc nghịch đảo" - ^)

Biểu thức logic

- Các phép so sánh: có kết quả kiểu logic
  - Bằng nhau: ==
  - Khác nhau: !=
  - Lớn hơn: >
  - Lớn hơn hoặc bằng: >=
  - Nhỏ hơn: <
  - Nhỏ hơn hoặc bằng: <=
- Nên dùng cặp ngoặc để làm rõ thứ tự tính toán
  - ☐ `(a + 5 < 0) || (a >= b) && (a != c)`
  - ☑ `((a + 5) < 0) || ((a >= b) && (a != c))`

Phép toán AND

- Tiếng Anh: AND
- Tiếng Việt: VÀ
- Trong ngôn ngữ C/C++: &&

"chỉ đúng khi cả 2 vế đều đúng"

- Ví dụ:

```
(a > b) && (a > c)
((x % 2) == 0) && ((x % 5) == 0)
```

Phép toán OR

- Tiếng Anh: OR
- Tiếng Việt: HOẶC
- Trong ngôn ngữ C/C++: ||

"chỉ sai nếu cả 2 vế đều sai"

- Ví dụ:

```
(a == 1) || (a == 3)
(a > (b+c)) || (b > (a+c)) || (c > (a+b))
```

Phép toán XOR

- Tiếng Anh: XOR
- Tiếng Việt: HOẶC NGHỊCH ĐẢO
- Trong ngôn ngữ C/C++: ^

"sai nếu 2 vế có giá trị giống nhau"

- Ví dụ:

```
(a > 10) ^ (b > 10)
(a > b) ^ (a <= b)
```

Bảng chân lý của các phép logic

x	y	x && y	x || y	x ^ y
True	True	True	True	False
True	False	False	True	True
False	True	False	True	True
False	False	False	False	False

Phần 6

Bài tập

Bài tập

1. Tính giá trị của các biểu thức logic sau
   1. (100 >= 2) && (2 < 3)
   2. (a > b) || (a < b)
   3. (a + b) != (b + a)
   4. ((a % 2) != 1) || ((a % 2) != 0)
2. Hãy chỉ ra khi nào những biểu thức logic sau là sai
   1. ((a+b) > c) && ((a+c) > b) && ((b+c) > a)
   2. (a <= b) && (a <= c)
   3. (a * b) < 0
   4. (a == b) ^ (a != b)