

TIN HỌC ĐẠI CƯƠNG

Bài 10: Kiểu dữ liệu string

Nội dung

1. Hằng số, tham chiếu và kiểu dữ liệu
2. Phạm vi và vòng đời của biến
3. Các kiểu dữ liệu tự tạo
4. Dãy kí tự (string)
 - Kiểu dữ liệu string
 - Khai báo và sử dụng string
 - Sử dụng chỉ mục với string
 - Các hàm làm việc với string
5. Vài ví dụ về xử lý string
6. Vài bài tập về xử lý string

Phần 1

Hằng số, tham chiếu và kiểu dữ liệu

Hằng số

- Hằng số = các giá trị cố định, không thay đổi trong toàn bộ chương trình
- Dùng trong biểu thức tương tự như một biến
- Khai báo hằng số:

```
const <kiểu> <tên hằng số> = <giá trị>;  
const bool b = false;           // hằng số logic  
const double pi = 3.14;         // hằng số số thực  
double x = 2 * 2 * pi;          // sử dụng hằng số
```
- *Hỏi: tại sao nên dùng hằng số mà không viết trực tiếp giá trị vào câu lệnh?*

Tham chiếu

- Tham chiếu: bí danh (alias) cho một biến
- Khai báo tham chiếu:
`<kiểu> & <tên biến> = <tên biến>;`
`int & n = m; // n là bí danh của m`
`double & x = y; // x là bí danh của y`
`x = 10; // y cũng bằng 10 luôn`
- Đặc điểm: tác động vào bí danh cũng giống như tác động trực tiếp vào biến
- *Hỏi: tại sao phải sử dụng bí danh của một biến mà không sử dụng trực tiếp biến đó?*

Kiểu dữ liệu cơ bản trong C/C++

- Logic: `bool`
- Kí tự: `char` (lưu giá trị mã hóa của các chữ)
- Số nguyên:
 - Có dấu: `char`, `short`, `int`, `long`, `long long`
 - Không dấu: thêm “`unsigned`” vào trước
- Số thực: `float`, `double`, `long double`
- Một số chú ý:
 - Xem chi tiết hơn ở phần 5.2 của giáo trình
 - C/C++ dùng lẫn lộn số nguyên và các kiểu khác
 - Kích cỡ của kiểu `int` tùy thuộc vào hệ điều hành

Phần 2

Phạm vi và vòng đời của biển

Phạm vi và vòng đời của biến

- Đây là hai khái niệm cơ bản giúp lập trình viên nắm bắt được nguyên tắc sử dụng biến trong khi viết chương trình
- “**phạm vi**” của biến = đoạn chương trình có thể sử dụng biến đó
 - Một số khái niệm liên quan: biến toàn cục, biến cục bộ, biến làm tham số của hàm, biến tĩnh,...
- “**vòng đời**” của biến = khoảng thời gian có thể sử dụng biến đó
 - Chú ý vòng đời của biến tĩnh (static)

Phần 3

Các kiểu dữ liệu tự tạo

Các kiểu dữ liệu tự tạo

- Kiểu dữ liệu: Hầu hết các kiểu dữ liệu trong máy tính đều phỏng theo các “loại” dữ liệu mà con người thường sử dụng
- Các ngôn ngữ lập trình cung cấp một số kiểu dữ liệu cơ bản (số nguyên, số thực, logic,...)
- Cho phép người dùng tổ hợp các dữ liệu cơ bản thành các loại phức tạp hơn. Ví dụ:
 - Phân số: tử số (số thực) + mẫu số (số thực)
 - Sinh viên: tên (chuỗi kí tự) + địa chỉ (chuỗi kí tự) + điểm trung bình học tập (số thực)

Tự tạo kiểu dữ liệu mới

- Ví dụ tự tạo kiểu dữ liệu phân số

```
struct PhanSo {  
    double tuso;  
    double mauso;  
};
```

- Ngoài khai báo dữ liệu, còn cần định nghĩa các phép toán, hàm,... sử dụng với kiểu dữ liệu đó
 - Tự tìm hiểu vì nằm ngoài phạm vi chương trình học
- Rất nhiều kiểu dữ liệu tự tạo được sử dụng phổ biến (string, vector, list,...)

Khai báo struct

- Cú pháp sử dụng struct:

```
struct <tên kiểu> {  
    <các dữ liệu thành phần>  
};
```

- Ví dụ:

```
struct ThoiGian { // kiểu dữ liệu ThoiGian  
    int ngay, thang, nam; // các thành phần con  
};  
ThoiGian homnay; // biến kiểu ThoiGian  
homnay.ngay = 21; // thành phần ngày = 21  
homnay.thang = 10; // thành phần tháng = 10  
homnay.nam = 2016; // thành phần năm = 2016
```

Phần 4

Dãy kí tự (string)

Kiểu dữ liệu “dãy kí tự”

- Nhiều phần mềm có nhu cầu xử lý dãy các kí tự, chẳng hạn như làm việc với tên của khách hàng, địa chỉ, email, chức vụ công tác,...
 - Xuất hiện nhu cầu xử lý các kí tự theo loạt
- Thời kì ban đầu, các lập trình viên tự tạo kiểu dữ liệu **string**, bản chất là dãy các kí tự, để xử lý các nhu cầu đó
- Khi việc sử dụng trở nên quá phổ biến, người ta chuẩn hóa đưa vào trong thư viện của C++
- “string” là kiểu dữ liệu tự tạo phổ biến nhất

Khai báo và sử dụng string

- Muốn sử dụng, cần: `#include <string>`

- Cách khai báo biến:

```
string str;           // int x;  
string w("Hello");   // int m(100);  
string s = "Hello";  // int n = 10;
```

- Chú ý:

- Một chữ (char) được viết trong cặp nháy đơn ('a')
- Một giá trị string viết trong cặp nháy kép ("Ok", "How are you?", "x", "", ...)
- Cần viết một string có chứa dấu nháy kép thì sao?

Khai báo và sử dụng string

- Phép toán ghép chuỗi: 2 string có thể ghép với nhau bằng phép cộng (+)

```
string ho = "Nguyen";  
string ten = "Ai Quoc";  
string hoten = ho + " " + ten;
```

- In string ra màn hình qua `cout`

```
cout << hoten << endl;
```

- Nhập dữ liệu bằng 2 cách: `getline` hoặc `cin`

```
cin >> hoten;           // nhập 1 đoạn  
getline(cin, hoten);   // nhập 1 dòng
```


Sử dụng chỉ mục với string

Dữ liệu	H	A	P	P	Y		N	E	...
Chỉ mục	0	1	2	3	4	5	6	7	...

Chuỗi `s = "HAPPY NEW YEAR";`

`s[0] → 'H'` `s[1] → 'A'` `s[2] → 'P'` ...

Cách dùng:

```
cout << s[1] << endl;           // in ra 'A'
s[4] = 'I';                     // đổi 'Y' thành 'I'
for (int i = 0; i < 10; i++)    // in 10 chữ đầu tiên
    cout << s[i] << endl;
```

Các hàm làm việc với string

- Các string sử dụng các phép so sánh để so giá trị với nhau (so sánh theo giá trị từ điển)
- Thư viện <string> có rất nhiều hàm xử lý xâu, tuy nhiên, cần luyện tập (viết mã) nhiều để có thể sử dụng thông thạo các hàm này.
- Các hàm cơ bản: xem bảng 5.1 giáo trình
- Các hàm được cung cấp là các hàm thành phần của kiểu string, cách dùng hơi đặc biệt:

```
cout << s.length(); // in ra độ dài của s
int p = s.find("PP"); // tìm xâu "PP" trong s
```

Các hàm làm việc với string

- Một số hàm thông dụng của kiểu string:
 - Hàm `length()` hoặc `size()`: trả về chiều dài của string (số kí tự có trong string)
 - Hàm `push_back(c)`: thêm kí tự `c` vào cuối string
 - Hàm `insert(v, str)`: chèn chuỗi `str` vào vị trí `v` trong string hiện tại
 - Hàm `erase(v, k)`: xóa `k` kí tự bắt đầu từ vị trí `v`
 - Hàm `find(str)`: tìm vị trí xuất hiện đầu tiên của `str` trong string hiện tại
 - Hàm `substr(v, k)`: tạo ra chuỗi mới là chuỗi con của string bằng cách lấy `k` kí tự từ vị trí `v`

Phần 5

Vài ví dụ về xử lý string

Ví dụ 1

```
// Nhập vào một chuỗi s từ bàn phím  
// In ra: số kí tự của chuỗi, chữ cái đầu tiên và cuối cùng
```

```
#include <iostream>  
#include <string>  
using namespace std;  
  
int main() {  
    string s;  
    cout << "Moi nhap mot chuoai: "; getline(cin, s);  
    cout << "Do dai cua chuoai: " << s.length() << endl;  
    cout << "Chu cai dau: " << s[0] << endl;  
    cout << "Chu cai cuoi: " << s[s.length()-1] << endl;  
}
```

Ví dụ 2

```
// Nhập vào một chuỗi s từ bàn phím  
// In ra: các chữ số nằm trong chuỗi  
// Gợi ý: dùng hàm isdigit của thư viện ctype để kiểm tra
```

```
#include <iostream>  
#include <string>  
#include <ctype>  
using namespace std;
```

```
int main() {  
    string s;  
    cout << "Moi nhap mot chuoi: "; getline(cin, s);  
    cout << "Cac chu so: ";  
    for (int i = 0; i < s.length(); i++)  
        if (isdigit(s[i])) cout << s[i];  
}
```

Ví dụ 3

```
// Nhập vào từ bàn phím một chuỗi gồm nhiều từ  
// (mỗi từ cách nhau bởi kí tự trống)  
// Yêu cầu: xóa đi từ đầu tiên trong chuỗi
```

```
#include <iostream>  
#include <string>  
using namespace std;
```

```
int main() {  
    string s;  
    cout << "Moi nhap mot chuoai: "; getline(cin, s);  
    int v = s.find(' ');  
    s.erase(0, v);  
    cout << "Chuoai sau khi xoa: " << s;  
}
```

Ví dụ 4

```
// Nhập vào từ bàn phím một chuỗi  
// Yêu cầu: đảo ngược chuỗi rồi in ra
```

```
#include <iostream>  
#include <string>  
using namespace std;  
  
int main() {  
    string s;  
    cout << "Moi nhap mot chuoai: "; getline(cin, s);  
    string w = "";  
    for (int i = 0; i < s.length(); i++)  
        w = s[i] + w;  
    cout << "Chuoi dao nguoc: " << w;  
}
```


Phần 6

Vài bài tập về xử lý string

Bài tập về xử lý string

- Rất nhiều và phong phú
- Là phần quan trọng của môn học
- Là phần xử lý cần thiết trong nhiều phần mềm
- Khá phức tạp đối với người mới học lập trình
- Một số dạng cơ bản:
 - Nhập dữ liệu và kiểm tra ràng buộc
 - Xử lý, chuẩn hóa chuỗi theo các điều kiện
 - Tìm, thay thế chuỗi
 - Xử lý danh sách các chuỗi
 - Tìm kiếm hoặc xác định các đặc trưng của chuỗi
 - Có phải là dạng số không, có phải dạng email không,...

Bài tập về xử lý string

1. Nhập hai chuỗi S và W, nếu S và W giống nhau thì in S, nếu S và W khác nhau thì in ra chuỗi được ghép bởi S+W
2. Nhập vào chuỗi S, kiểm tra xem S có chứa toàn các chữ số hay không?
3. *Nhập và đếm số từ trong chuỗi S (một từ là dãy các kí tự liên tiếp không chứa dấu cách)
4. Xóa mọi kí tự A trong chuỗi W nhập từ bàn phím
5. Chuỗi W chỉ chứa dấu (hoặc), kiểm tra xem W có phải chuỗi ngoặc đúng hay không? Ngoặc đúng là có phương án ghép cặp (và)
 - ((()()) là ngoặc đúng
 - ())() là ngoặc sai