TIN ĐẠI CƯƠNG

BÀI 9: MẢNG VÀ KIỂU DỮ LIỆU VECTOR

Nội dung buổi trước: STRING

- Kiểu "string": bản chất là dãy các kí tự viết liên tiếp, được đánh thứ tự theo số nguyên bắt đầu từ 0 (gọi là chỉ mục)
- Thư viện: #include <string>
- Khai báo:
  - string str;
  - string w("Hello");
  - string s = "Hello";
- Các hàm cơ bản: tham khảo giáo trình
- Chú ý về nhập liệu: dùng hàm getline

Nội dung

1. Khuôn mẫu (template)
2. Kiểu dữ liệu mảng (vector)
   - Giới thiệu
   - Hệ thống chỉ mục
   - Khởi tạo, nhập và xuất dữ liệu
   - Một số hàm hỗ trợ
3. Kiểu mảng gốc
4. Bài tập

Phần 1

Khuôn mẫu (template)

Khuôn mẫu (template)

- Nhiều thuật toán có tính tổng quát, có thể áp dụng được cho nhiều loại dữ liệu khác nhau
- Ví dụ: tìm phần tử lớn nhất trong 2 phần tử

```
int max(int a, int b) {
    if (a > b) return a; else return b;
}
double max(double a, double b) {
    if (a > b) return a; else return b;
}
string max(string a, string b) {
    if (a > b) return a; else return b;
}
```

Khuôn mẫu (template)

- Ngôn ngữ C++ cho phép chúng ta "tổng quát hóa" các đoạn mã tương tự này bằng cách sử dụng template
- Ví dụ: tìm phần tử lớn nhất trong 2 phần tử

```
template <class T> T max(T a, T b) {
    if (a > b) return a; else return b;
}
```

- Sử dụng: máy tính sẽ tự động thay thế kiểu dữ liệu thích hợp trong từng tính huống cụ thể

```
cout << max(100,200) << endl;
```

Hàm max với kiểu int

```
cout << max(1.5,1.3) << endl;
```

Hàm max với kiểu double

Phần 2

Kiểu dữ liệu mảng (vector)

Giới thiệu

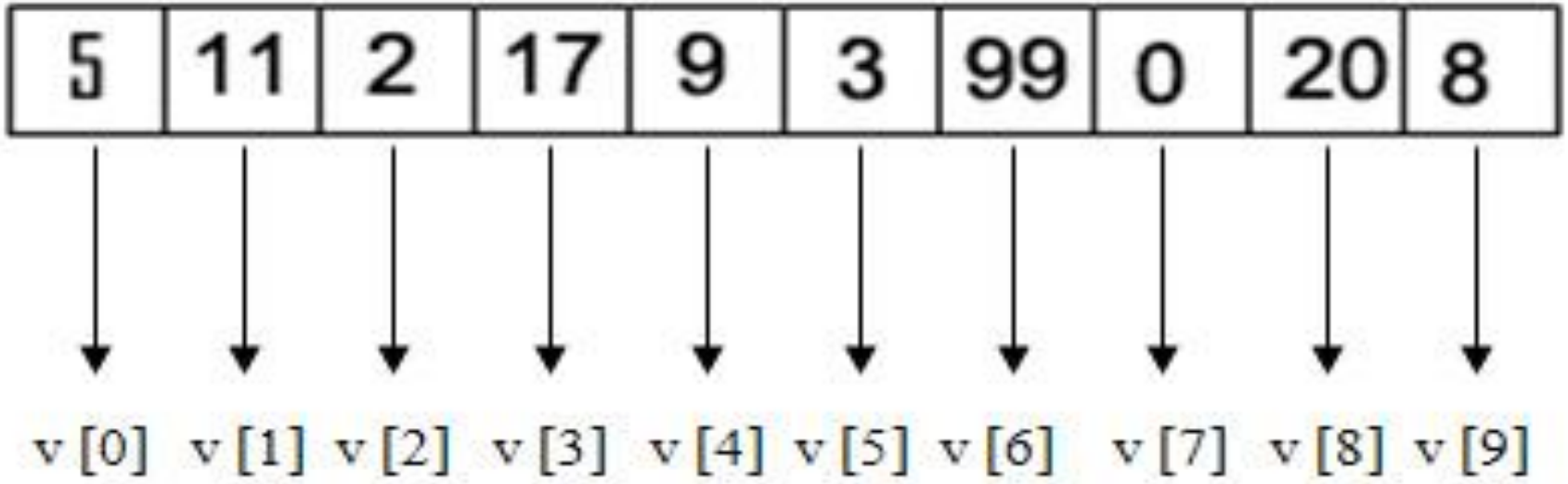
- Kiểu dữ liệu vector (mảng, dãy,…) lấy ý tưởng từ khái niệm dãy số trong toán học
  - Toán: $X = (x_1, x_2, \ldots, x_n)$
  - C++: X = (x[0], x[1],…, x[n-1])
- Đặc điểm:
  - Vector = các biến có cùng tên, phân biệt bởi chỉ số
  - Vector không nhất thiết chỉ là dãy số, mà có thể là một dãy bất kỳ, chẳng hạn:
    - Dãy các giá trị bool: vector<bool>
    - Dãy string: vector<string>
    - Dãy của các dãy số nguyên: vector<vector<int>>

Giới thiệu

- Rất nhiều bài toán kĩ thuật và quản lý sử dụng vector để xử lý dãy (nhưng không phải là cách duy nhất), vài tình huống thực tế:
  - Quản lý điểm số của sinh viên
  - Thống kê xử lý số liệu (có bao nhiêu sinh viên loại giỏi, có bao nhiêu thi trượt,…)
  - Các bài toán kĩ thuật, tính toán dãy hoặc ma trận
  - Xử lý hiệu ứng âm thanh, hình ảnh, video,…
- Vector sử dụng kĩ thuật template (được giới thiệu ở phần trước)

Hệ thống chỉ mục

- Tương tự hệ thống chỉ mục của kiểu string
- Đánh thứ tự số nguyên, bắt đầu từ 0
- Viết bên trong cặp ngoặc vuông
- Mỗi một ô có thể xem như một biến độc lập



Khai báo, khởi tạo dữ liệu

- Thư viện: `#include <vector>`
- Khai báo biến:

```
vector<bool> m;                // dãy giá trị logic
vector<int> a(10);             // dãy 10 số nguyên
vector<double> b(10, 0.5);     // dãy 10 số 0.5
```

- Một vài chú ý khi thao tác biến vector:
  - Nên sử dụng hàm size() để lấy độ dài của dãy
  - Nếu không được chỉ rõ, vector sẽ có độ dài = 0
  - Rất cẩn thận khi sử dụng cách khai báo thứ 2
- Vector có thể khai báo lồng nhau (phức tạp)

```
vector<vector<double>> A(10);
```

Nhập dữ liệu

```
// nhập kích cỡ của dãy trước
cout << "N = "; cin >> n;
// tạo dãy có đúng n phần tử
vector<int> a(n);
// nhập từng phần tử từ bàn phím
for (int i = 0; i < a.size(); i++) {
    // in ra lời mời: "A[0] = "
    cout << "A[" << i << "] = ";
    // nhập dữ liệu vào vector
    cin >> a[i];
}
```

Xuất dữ liệu

```
// in ra dòng thông báo "A = "
cout << "A = ";
// in ra từng phần tử của vector
// mỗi phần tử cách nhau bởi dấu trống
for (int i = 0; i < a.size(); i++) {
    cout << a[i] << " ";
}
// in xong thì xuống dòng
cout << endl;
```

Một số hàm hỗ trợ

- Có rất nhiều hàm do thư viện vector cung cấp để thao tác dãy (xem bảng 6-2 và phụ lục)
- Một số hàm thông dụng:
  - v.clear(): xóa rỗng vector v
  - v.empty(): trả về true nếu vector v rỗng
  - v.pop_back(): bỏ phần tử cuối cùng ra khỏi v
  - v.push_back(e): chèn e vào cuối vector v
  - v.size(): trả về số phần tử của vector v
  - v.back(): trả về giá trị của phần tử cuối cùng của v
  - v.resize(m): chỉnh lại cỡ của vector thành m phần tử (giữ nguyên giá trị những phần tử cũ)

Hàm thành phần thường dùng

- Lấy phần tử đầu tiên: `v.front()`
- Lấy phần tử cuối cùng: `v.back()`
- Lấy phần tử ở vị trí n: `v.at(n)` ~ `v[n]`
- Thêm x vào cuối v: `v.push_back(x)`
- Xóa phần tử cuối cùng của v: `v.pop_back()`
- Chèn x vào vị trí n: `v.insert(v.begin()+n, x)`
- Xóa phần tử thứ n: `v.erase(v.begin()+n)`
- Hoán đổi nội dung giữa v và y: `v.swap(y)`
- Chỉnh lại cỡ của vector: `v.resize(n)`

Phần 3

Kiểu mảng gốc

Kiểu mảng gốc

- Ngoài vector, C/C++ còn có thể sử dụng kiểu mảng gốc với cách viết đơn giản hơn
- Khai báo biến:

```
bool x[100];  // dãy x có 100 giá trị logic
```

- Ưu điểm:
  - Viết đơn giản, không cần thư viện vector
  - Sử dụng chỉ mục để truy cập các biến bên trong
- Nhược điểm:
  - Không có hàm hỗ trợ
  - Kích thước là hằng số, không thay đổi được

Phần 4

Bài tập

Một số bài tập cơ bản

1. *Nhập số nguyên dương N và dãy N số thực, in ra các số vừa nhập*
2. *Nhập dãy N số thực và tính tổng tất cả các số trong dãy*
3. *Nhập dãy N số thực và tính trung bình cộng của các số trong dãy*
4. *Nhập dãy N số nguyên và tính trung bình cộng các số dương trong dãy*
5. *Nhập dãy N số thực, tìm giá trị lớn nhất trong dãy*

Một số bài tập cơ bản

6. *Nhập dãy N số nguyên, hãy tìm xem có bao nhiêu số trong dãy có giá trị bằng số lớn nhất của dãy*
7. *Nhập dãy N số thực, đếm và in ra màn hình các số trong dãy có giá trị nhỏ hơn trung bình cộng của dãy*
8. *Nhập dãy N số thực, sắp xếp lại các số trong dãy giảm dần theo giá trị*
9. *Nhập danh sách N sinh viên, sắp xếp lại danh sách theo sinh viên theo thứ tự từ điển*