CÁC LỚP DỰNG SẴN

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

GVHD: Trương Toàn Thịnh

NỘI DUNG

- Các kiểu dữ liệu cơ sở
- Kiểu chuỗi trong C++
- Kiểu dữ liệu động
  - Dùng vector tạo mảng động một chiều
  - Dùng vector tạo mảng động hai chiều

CÁC KIỂU DỮ LIỆU CƠ SỞ

- C++ xây dựng sẵn nhiều kiểu dữ liệu cơ sở chia thành các nhóm:
  - Số nguyên: int, long, short, char (unsigned)
  - Số thực: float, double, long double
  - Luận lý: bool
  - Kí tự: char, wchar_t

CÁC KIỂU DỮ LIỆU CƠ SỞ

- Số thực
  - Float: kích thước 32 bits lưu giá trị từ 1.4×10^{-45} tới 3.4×10^{38}.
  - Double: kích thước 64 bits lưu giá trị từ 4.94×10^{-324} tới 1.79×10^{308}.
- Định nghĩa sẵn bốn phép toán: + – × ÷
- Thư viện <cmath> hỗ trợ thêm các hàm phức tạp khác: sqrt, pow, exp, log, abs, labs, fabs, cos, sin, tan, acos, asin, atan, floor, ceil.

CÁC KIỂU DỮ LIỆU CƠ SỞ

- Ví dụ: Tính chương trình $log_a b$
  - Gợi ý: dùng công thức $log_a N = log_a b \times log_b N$

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

void main(){
  double a, b;
  cin >> a >> b;
  double kq = log(b)/log(a);
  cout << kq << endl;
}
```

CÁC KIỂU DỮ LIỆU CƠ SỞ

- Số nguyên
  - char: kích thước 8 bits lưu giá trị từ -128 tới 127.
  - int: kích thước 16/32 bits lưu giá trị từ -2^{n-1} tới $2^{n-1} - 1$ (n = 16 hay 32 bits).
  - short: kích thước 16 bits lưu giá trị từ -32768 tới 32767.
  - long: kích thước 32 bits lưu giá trị từ -2,147,483,648 tới 2,147,483,647.
- Định nghĩa sẵn bốn phép toán: + − × ÷ %
- Có thể biểu diễn số nguyên theo ba dạng:
  - Dạng bát phân: $0225 = 2 \times 8^2 + 2 \times 8^1 + 5 = 149$
  - Dạng thập lục: $0x3B = 3 \times 16^1 + 11 = 59$
  - Dạng thập phân

CÁC KIỂU DỮ LIỆU CƠ SỞ

- Luận lý
  - bool: kích thước 1 bit lưu giá trị false (0) hay true (1).
- Định nghĩa các phép toán luận lý:
  - Phép &&: biểu thức (A && B) là true nếu tất cả đều đúng và false nếu A hay B sai
  - Phép ||: biểu thức (A || B) là false nếu tất cả đều sai và true nếu A hay B đúng
  - Phép !: biểu thức (!B) là false khi B đúng và true khi B sai.

CÁC KIỂU DỮ LIỆU CƠ SỞ

- Kí tự
  - unsigned char: dùng kiểu char không dấu để lưu giá trị kí tự, ví dụ unsigned char c = ‘c’;
  - Kiểu char dùng để lưu giá trị từ [0, 255] trong bảng ASCII.
  - Lưu ý: sizeof(‘c’) = 1 (byte)
  - Unsigned wchar_t: dùng kiểu wchar_t không dấu để lưu giá trị kí tự, ví dụ unsigned wchar_t c = L‘c’ hoặc unsigned wchar_t c = ‘c’;
  - Kiểu wchar_t dùng để lưu ~ 65000 giá trị trong bảng mã quốc tế
  - Lưu ý: sizeof(L‘c’) = 2 (byte)
  - Cần include <cwchar> khi muốn dùng kí tự 2-byte

NỘI DUNG

- Các kiểu dữ liệu cơ sở
- Kiểu chuỗi trong C++
- Kiểu dữ liệu động
  - Dùng vector tạo mảng động một chiều
  - Dùng vector tạo mảng động hai chiều

KIỂU CHUỖI TRONG C++ (Chuỗi 8-bit)

- Trong C++ không dựng sẵn kiểu chuỗi 8-bit
- Cần "#include <string>" để sử dụng kiểu string
- Kiểu string thuộc thư viện C++ Standard Template Library (C++ STL)
- Kiểu string chứa một mảng (chuỗi) các kí tự, mỗi kí tự kích thước 8-bit (gọi là chuỗi 8-bit)

KIỂU CHUỖI TRONG C++ (Chuỗi 8-bit)

- Phân tích kiểu string
  - Dữ liệu (thuộc tính)
    - Số lượng kí tự chuỗi có thể chứa
    - Con trỏ chứa địa chỉ chuỗi kí tự
  - Phương thức
    - Lấy độ dài – length(): int
    - Lấy chuỗi con – substring(int starPos, int nChar): string
    - Toán tử nhập/xuất (>>, <<), so sánh (==, !=, >, <…)
    - Toán tử trích xuất ([])
    - …

KIỂU CHUỖI TRONG C++ (Chuỗi 8-bit)

- Ví dụ: cách viết dạng con trỏ và đối tượng

```
#include <string>
#include <iostream>
using namespace std;
void main() {
  string *s = new string();
  getline(cin, *s);
  string *t = new string();
  getline(cin, *t);
  cout << (*s) << endl;
  cout << s->length() << endl;
  cout << (*t) << endl;
  cout << t->length() << endl;
  string kq = (*s) + (*t);
  cout << kq << endl;
  cout << kq.length() << endl;
  string* d = new string(kq.c_str());
  cout << *d << endl;
  cout << d->length() << endl;
}
```

```
#include <string>
#include <iostream>
using namespace std;
void main() {
  string s;
  getline(cin, s);
  string t;
  getline(cin, t);
  cout << s << endl;
  cout << s.length() << endl;
  cout << t << endl;
  cout << t.length() << endl;
  string kq = s + t;
  cout << kq << endl;
  cout << kq.length() << endl;
  string d(kq.c_str());
  cout << d << endl;
  cout << d.length() << endl;
}
```

KIỂU CHUỖI TRONG C++ (Chuỗi 16-bit)

- Trong C++ không dựng sẵn kiểu chuỗi 16-bit
- Cần "#include <wstring>" để sử dụng kiểu wstring
- Kiểu wstring thuộc thư viện C++ Standard Template Library (C++ STL)
- Kiểu wstring chứa một mảng (chuỗi) các kí tự, mỗi kí tự kích thước 16-bit (gọi là chuỗi 16-bit)

KIỂU CHUỖI TRONG C++ (Chuỗi 16-bit)

- Mỗi kí tự trong chuỗi 16-bit có kiểu wchar_t.
- Hằng chuỗi kí tự 16-bit bắt đầu bằng kí tự L, ví dụ: L“This is a string”
- Phương thức length() của đối tượng kiểu wstring vẫn trả ra số lượng kí tự (KHÔNG trả ra kích thước theo byte)
- Thêm tiền tố ‘w’ trước các thư viện và đối tượng khi dùng với chuỗi 16-bit:
  - Ví dụ: wostream, wistream, wcout, wcin

NỘI DUNG

- Các kiểu dữ liệu cơ sở
- Kiểu chuỗi trong C++
- Kiểu dữ liệu động
  - Dùng vector tạo mảng động một chiều
  - Dùng vector tạo mảng động hai chiều

KIỂU DỮ LIỆU ĐỘNG

- C++ hỗ trợ một vài kiểu dữ liệu cấu trúc hỗ trợ quá trình lưu trữ và truy xuất.
- Nếu không dùng thư viện ta dùng con trỏ để xây dựng mảng động một chiều

```
#include <iostream>
using namespace std;
void main(){
  int n; float* a;
  cin>>n;
  a = new float[n];
  for(int i = 0; i < n; i++)
    cin>>a[i];
  for(int i = 0; i < n; i++)
    cout<<a[i]<<“ ”;
  delete[] a;
}
```

KIỂU DỮ LIỆU ĐỘNG (vector tạo mảng động một chiều)

- C++ hỗ trợ kiểu vector hỗ trợ quá trình lưu trữ và truy xuất.

```
#include <iostream>
using namespace std;
void main(){
  int n; vector<float> a;
  cin>>n;
  a.resize(n);
  for(int i = 0; i < a.size(); i++)
    cin>>a[i];
  for(int i = 0; i < a.size(); i++)
    cout<<a[i]<<“ ”;
}
```

- Lưu ý: Không cần hủy (dọn dẹp) đối tượng ‘a’ sau khi đã dùng xong.

KIỂU DỮ LIỆU ĐỘNG (vector tạo mảng động một chiều)

- Phương thức push_back của đối tượng kiểu vector

```
#include <iostream>
using namespace std;
void main(){
  int n; vector<float> a; float tmp;
  cin>>n;
  for(int i = 0; i < n; i++){
    cin>>tmp;
    a.push_back(tmp);
  }
  for(int i = 0; i < a.size(); i++)
    cout<<a[i]<<“ ”;
}
```

- Lưu ý: Không cần dùng phương thức resize() để qui định kích thước của vector

KIỂU DỮ LIỆU ĐỘNG (vector tạo mảng động một chiều)

- Quá tải toán tử nhập/xuất cho lớp vector

```
istream& operator>>(istream& iDev, vector<float> &a) {
  float tmp;
  iDev.clear();
  while (iDev >> tmp)
    a.push_back(tmp);
  iDev.clear();
  return iDev;
}
ostream& operator<<(ostream& oDev, const vector<float> &a) {
  for (int i = 0; i < a.size(); i++)
    oDev << a[i] << " ";
  return oDev;
}
void main() {
  vector<float> a, b;
  cin >> a >> b;
  cout << a << b;
}
```

KIỂU DỮ LIỆU ĐỘNG (vector tạo mảng động một chiều)

- Câu lệnh "iDev >> tmp" trong vòng lặp while sẽ trả ra đối tượng iDev nếu nhập thành công. Nếu ta nhấn 'Ctrl + Z' thì quá trình nhập thất bại
- Hai câu lệnh iDev.clear() để dọn dẹp đường ống stdin trong trường hợp nó chứa kí tự 'EOF' do việc ta nhấn 'Ctrl + Z'
- Trong hàm main sử dụng đối tượng kiểu vector rất dễ dàng: cin >> a >> b…

KIỂU DỮ LIỆU ĐỘNG
vector tạo mảng động hai chiều

- Áp dụng kiểu vector hai lần sẽ có được mảng động hai chiều

```
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;
typedef vector<double> doubleArray;

void initMatrix(vector<doubleArray> &a, int n){
  a.resize(n);
  for(int i = 0; i < n; i++){
    a[i].resize(n);
  }
}
```

KIỂU DỮ LIỆU ĐỘNG
vector tạo mảng động hai chiều

- Quá tải toán tử nhập/xuất

```
istream& operator>>(istream& iDev, vector<doubleArray> &a) {
  int n; iDev>>n; initMatrix(a, n);
  for (int i = 0; i < a.size(); i++){
    for (int j = 0; j < a[i].size(); j++){
      a[i][j] = 0;
      if(iDev) iDev >> a[i][j];
    }
  }
  return iDev;
}
ostream& operator<<(ostream& oDev, const vector<float> &a) {
  oDev<<a.size()<<endl;
  for (int i = 0; i < a.size(); i++){
    for (int j = 0; j < a[i].size(); j++)
      oDev << a[i][j] << " ";
    oDev << endl;
  }
  return oDev;
}
```

KIỂU DỮ LIỆU ĐỘNG
vector tạo mảng động hai chiều

- Sử dụng kiểu vector<doubleArray> trong hàm main

```
void main() {
  vector<doubleArray> a;
  cin >> a;
  cout << a;
}
```

BÀI TẬP

- Xây dựng lớp MyString với các thao tác:
  - Trả về số lượng kí tự
  - Trả về chuỗi con
  - Toán tử + chuỗi mới vào chuỗi
  - Phương thức chèn chuỗi mới vào chuỗi
  - Phương thức xóa chuỗi con
  - Phương thức thay thế chuỗi
  - Phương thức tìm chuỗi con trong chuỗi cha

BÀI TẬP

- Gợi ý
  - Lớp MyString gồm hai trường quan trọng char* và int.
  - Hàm trả về số lượng kí tự: int length()
  - Hàm trả về chuỗi con: char* subString(int, int)
    - Tham số int thứ nhất: vị trí bắt đầu trích xuất
    - Tham số int thứ hai: số lượng kí tự cần trích xuất
  - Toán tử + chuỗi mới vào chuỗi: char* operator+(const MyString&)
    - Lưu ý: giá trị chuỗi gốc không thay đổi
  - Phương thức chèn chuỗi mới vào chuỗi: bool insert(int, char*)
    - Tham số int thứ nhất: vị trí chèn
    - Tham số char* thứ hai: chuỗi nội dung cần chèn

BÀI TẬP

- Gợi ý
  - Phương thức xóa chuỗi con: bool erase(int, int)
    - Tham số int thứ nhất: vị trí bắt đầu xóa
    - Tham số int thứ hai: số lượng kí tự cần xóa
  - Phương thức thay thế chuỗi: bool replace(int, int, char*)
    - Tham số int thứ nhất: vị trí bắt đầu thay
    - Tham số int thứ hai: số lượng kí tự sẽ bị thay
    - Tham số char* thứ hai: chuỗi nội dung cần chèn vào
  - Phương thức tìm chuỗi con trong chuỗi cha: bool find(int, char*)
    - Tham số int thứ nhất: vị trí bắt đầu tìm
    - Tham số char* thứ hai: chuỗi nội dung cần tìm

BÀI TẬP

- Ví dụ trong hàm main
  - MyString s, t;
  - cin>>s >> t; // vi du: s = “hello ” t = “world”
  - cout << s;
  - cout << s.length(); // 6
  - cout << s.subString(2, 4); // ‘llo ’
  - cout << s + t;//“hello world”
  - cout << s.insert(1, “!”);//“h!ello ”
  - cout << s.erase(0, 2);//“llo ”
  - cout << s.replace(0, 2, “abc”);//“abllo ”
  - cout << s.find(0, “ll”);//true