VIỆN ĐIỆN TỬ - VIỄN THÔNG

School of Electronics and telecommunications

KỸ THUẬT LẬP TRÌNH C/C++

MẢNG VÀ QUẢN LÝ BỘ NHỚ ĐỘNG

Giảng viên: TS. Nguyễn Thị Kim Thoa

Học kỳ: 20211

Nội dung

- Mảng động
- Cấp phát và giải phóng bộ nhớ động
- Chuỗi ký tự trong C và C++

Mảng

- Mảng cho phép biểu diễn và quản lý dữ liệu một cách khá hiệu quả:
  - Đọc và ghi dữ liệu rất nhanh qua chỉ số hoặc qua địa chỉ
  - Tiết kiệm bộ nhớ
- Các vấn đề của mảng tĩnh:

  VD: Student student_list[100];
  - Số phần tử phải là hằng số (biết trước khi biên dịch, người sử dụng không thể nhập số phần tử, không thể cho số phần từ là một biến) => kém linh hoạt
  - Chiếm chỗ cứng trong ngăn xếp (đối với biến cục bộ) hoặc trong bộ nhớ dữ liệu chương trình (đối với biến toàn cục) => sử dụng bộ nhớ kém hiệu quả, kém linh hoạt

Mảng động

- Mảng động là một mảng được cấp phát bộ nhớ theo yêu cầu, trong khi chương trình chạy

```
#include <stdlib.h>  /* C */
int n = 50;
...
float*  p1= (float*) malloc(n*sizeof(float)); /* C */
double* p2= new double[n]; // C++
```

- Sử dụng con trỏ để quản lý mảng động: Cách sử dụng không khác so với mảng tĩnh

```
p1[0] = 1.0;
p2[0] = 2.0;
```

- Sau khi sử dụng xong => giải phóng bộ nhớ:

```
free(p1);       /* C */
delete [] p2;   // C++
```

Cấp phát và giải phóng bộ nhớ động

- C:
  - Hàm malloc() yêu cầu tham số là số byte, trả về con trỏ không kiểu (void*) mang địa chỉ vùng nhớ mới được cấp phát (nằm trong heap), trả về 0 nếu không thành công.
  - Hàm free() yêu cầu tham số là con trỏ không kiểu (void*), giải phóng vùng nhớ có địa chỉ đưa vào
- C++:
  - Toán tử new chấp nhận kiểu dữ liệu phần tử kèm theo số lượng phần tử của mảng cần cấp phát bộ nhớ (trong vùng heap), trả về con trỏ có kiểu, trả về 0 nếu không thành công.
  - Toán tử delete[] yêu cầu tham số là con trỏ có kiểu.
  - Toán tử new và delete còn có thể áp dụng cho cấp phát và giải phóng bộ nhớ cho một biến đơn, một đối tượng chứ không nhất thiết phải một mảng.

Một số điều cần lưu ý

- Con trỏ có vai trò quản lý mảng (động), chứ con trỏ không phải là mảng (động)
- Cấp phát bộ nhớ và giải phóng bộ nhớ chứ không phải cấp phát con trỏ và giải phóng con trỏ
- Chỉ giải phóng bộ nhớ một lần
- Ví dụ:

```
int* p;
p[0] = 1;                    // ??
new(p);              // ??
p = new int[100];    // OK
p[0] = 1;                    // OK
int* p2=p;           // OK
delete[] p2;         // OK
p[0] = 1;                    //??
delete[] p;          //??
p = new int[50];             // OK, new array
…
```

Cấp phát bộ nhớ động cho biến đơn

- Ý nghĩa: Các đối tượng có thể được tạo ra động, trong khi chương trình chạy (bổ sung sinh viên vào danh sách, vẽ thêm một hình trong bản vẽ, bổ sung một khâu trong hệ thống,...)
- Cú pháp

```
int* p = new int;
*p = 1;
p[0]= 2;                    // giống như trên
p[1]= 1;                    // ??
int* p2 = new int(1);       // có khởi tạo
delete p;
delete p2;
Student* ps = new Student;
ps->code = 1000;
...
delete ps;
```

Ý nghĩa của sử dụng bộ nhớ động

- Hiệu suất:
  - Bộ nhớ được cấp phát đủ dung lượng theo yêu cầu và khi được yêu cầu trong khi chương trình đã chạy
  - Bộ nhớ được cấp phát nằm trong vùng nhớ tự do còn lại của máy tính (heap), chỉ phụ thuộc vào dung lượng bộ nhớ của máy tính
  - Bộ nhớ có thể được giải phóng khi không sử dụng tiếp.
- Linh hoạt:
  - Thời gian "sống" của bộ nhớ được cấp phát động có thể kéo dài hơn thời gian "sống" của thực thể cấp phát nó.
  - Có thể một hàm gọi lệnh cấp phát bộ nhớ, nhưng một hàm khác giải phóng bộ nhớ.
  - Sự linh hoạt cũng dễ dẫn đến những lỗi "rò rỉ bộ nhớ".

Ví dụ sử dụng bộ nhớ động trong hàm

```
Date* createDateList(int n) {
      Date* p = new Date[n];
      return p;
}
void main() {
      int n;
      cout << "Enter the number of your national holidays:";
      cin >> n;
      Date* date_list = createDateList(n);
      for (int i=0; i < n; ++i) {
            ...
      }
      for (....) { cout << ....}
      delete [] date_list;
}
```

Bài tập

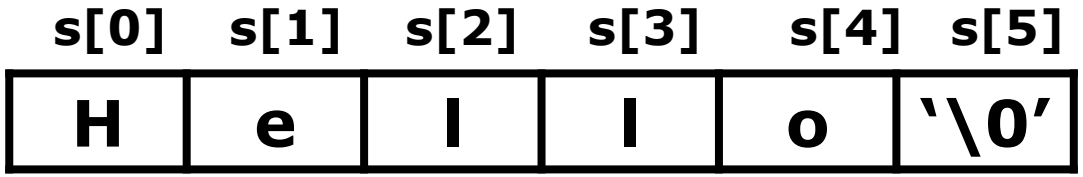
BT: Viết một hàm lưu các số chẵn của một mảng số nguyên cho trước ra một mảng khác.

Viết chương trình chính sử dụng hàm đó trong 2 trường hợp:

a. Mảng đầu vào là mảng tĩnh với các phần tử cho trước
b. Mảng đầu vào là mảng động với số lượng phần tử và các phần tử được nhập từ bàn phím

Chuỗi ký tự trong C (C-Style Strings)

- Chuỗi ký tự C-style bản chất là mảng 1 chiều các ký tự, kết thúc bằng ký tự '\0' (null). Hay còn gọi là null-terminated string.
  Ví dụ1: char s[]="Hello";//chuỗi s có bao nhiêu phần tử?
- Trong vùng nhớ:



- Ví dụ 2:

```
char szFullName[30]; // xâu dài bao nhiêu ký tự
char szDayOfBirth[9]; // xâu dài bao nhiêu ký tự
```

Khởi tạo giá trị cho mảng ký tự

```
char sz[10] = { ‘H’, ‘e’, ‘l’, ‘l’, ‘o’, ‘\0’ };
char sz[10] = “Hello"; // Tự động thêm '\0' vào cuối chuỗi
char sz[] = { ‘H’, ‘e’, ‘l’, ‘l’, ‘0’, ‘\0’ };
char sz[] = “Hello"; // Tự động thêm '\0' vào cuối chuỗi
```

Nhập mảng ký tự (C-style strings)

- Dùng **std::cin,** có 2 vấn đề
  - Không nhập được chuỗi ký tự có chứa khoảng trắng
    - Khi đọc thông tin từ bàn phím, đối tượng **std::cin** sẽ đọc các ký tự cho đến khi gặp ký tự khoảng trắng ' ', hoặc ký tự enter '\n'.
    - Nếu chuỗi chứa các khoảng trắng, đối tượng std::cin chỉ đọc được từ đầu tiên.
  - Tràn mảng khi nhập quá số lượng ký tự so với khai báo
    - Khai báo 1 mảng ký tự gồm n phần tử, nhưng người dùng cố tình (hoặc vô ý) nhập 1 chuỗi nhiều hơn n phần tử.
    - Lúc này, vấn đề tràn mảng xảy ra, và chương trình sẽ gặp lỗi.

Nhập mảng ký tự (C-style strings)

- Dùng hàm **std::cin.getline()**
  - Hàm std::cin.getline() sẽ đọc tất cả các ký tự từ bàn phím (bao gồm khoảng trắng ' ') cho đến khi gặp ký tự enter '\n' (mặc định).
  - Nếu số lượng ký tự nhập vào lớn hơn độ dài truyền vào hàm, mọi ký tự dư thừa sẽ bị loại bỏ.

Xuất mảng ký tự (C-style strings)

- Dùng std::cout
- Khi in mảng ký tự (C-style strings), đối tượng std::cout sẽ in tất cả ký tự cho đến khi gặp ký tự '\0' (null).

Các thao tác trên mảng ký tự (C-style string)

- Xem độ dài mảng ký tự:
  - Không bao gồm ký tự '\n': dùng hàm strlen(stringName).
  - Kích thước của toàn bộ mảng: sizeof(stringName)
- Chuyển chuỗi thành chuỗi thường: strlwr(stringName)
- Chuyển chuỗi thành chuỗi in hoa: strupr(stringName)
- Sao chép chuỗi s2 sang cho chuỗi s1: strcpy(s1,s2). Chuỗi s1 phải đủ lớn để chứa s2.
- Nối chuỗi s2 vào chuỗi s1: strcat(s1,s2). Chuỗi s1 phải đủ lớn để chứa thêm chuỗi s2.

Các thao tác trên mảng ký tự (C-style string)

- So sánh hai mảng ký tự s1 và s2 (phân biệt hoa thường), dùng hàm strcmp(s1,s2):
  - Giá trị trả về nhỏ hơn 0 nếu: chuỗi s1 < chuỗi s2
  - Giá trị trả về bằng 0 nếu: chuỗi s1 == chuỗi s2
  - Giá trị trả về lớn hơn 0 nếu: chuỗi s1 > chuỗi s2
- Tìm kiếm chuỗi s2 trong chuỗi s1: strstr(s1, s2).
  - Nếu tìm thấy: trả về con trỏ đến vị trí xuất hiện đầu tiên của chuỗi s2 trong chuỗi s1.
  - Nếu không tìm thấy: trả về NULL.

Chuỗi ký tự trong C++

- Trong C++, kiểu chuỗi ký tự được cài đặt trong một lớp string của thư viện chuẩn STL (C++ Standard Template Library).
- Khai báo, khởi tạo và gán giá trị một chuỗi ký tự:

```
/* Khai báo một chuỗi ký tự */
string str;
/* Khởi tạo một chuỗi theo nhiều cách */
string str0(""); // Khởi tạo chuỗi str0 rỗng
string str1(“Hello"); // Khởi tạo chuỗi "Hello" cho str1
string str2{ "Hello world" }; // Khởi tạo chuỗi "Hello world" cho str2
string str3 = "Hello ever body"; // Khởi tạo chuỗi "Hello ever body" cho str3
string str4 = str1; // Gán giá trị chuỗi str1 cho str4
```

Nhập một chuỗi kỹ tự trong C++

- Dùng std::getline(cin,ten_chuoi);
- Vấn đề xảy ra khi kết hợp std::cin với std::getline():

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
        cout << "Enter your age: ";
        int age;
        cin >> age;
        cout << "Enter your name: ";
        string name;
        getline(cin, name);
        cout << "Hello, " << name <<
endl;
        cout << "Your age " << age <<
endl;
        return 0;
 }
```

Thêm cin.ignore();

Nhập một văn bản bao gồm ký tự xuống dòng

- Mặc định, hàm std::getline() sử dụng ký tự '\n' khi nhấn phím Enter là ký tự. Nếu muốn nhập nhiều dòng văn bản vào một biến string, cần thay đổi lệnh std::getline() như sau:

getline(cin, ten_chuoi, '_');

- Khi đó, có thể nhập chuỗi ký tự cho đến khi chương trình nhận vào ký tự '_'.

Xuất chuỗi ký tự trong C++

- Dùng lệnh std::cout

Một số thao tác cơ bản với chuỗi ký tự

- Nối chuỗi: sử dụng toán tử cộng để nối hai chuỗi

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
string a(“123");
string b(“456");
string c = a + b;
cout << c << endl;
a += " string";
cout << a << endl;
return 0;
}
```

c=“123456”

a=“123 string”

Một số thao tác cơ bản với chuỗi ký tự

- Lấy ra độ dài chuỗi ký tự:

  stringName.length() hoặc stringName.size()