Quản lý và sử dụng bộ nhớ động

Kỹ thuật lập trình

ThS. Đặng Bình Phương (dbphuong@fit.hcmus.edu.vn)

Nội dung

- Mảng động trong thư viện chuẩn
- Khái niệm về con trỏ
- Dữ liệu có cấu trúc (dạng struct hay union) và con trỏ
- Cấp phát và sử dụng dữ liệu động
- Đồ án lập trình
- Các vấn đề tìm hiểu mở rộng kiến thức nghề nghiệp
- Thuật ngữ tiếng Anh và bài đọc thêm tiếng Anh

Mảng động trong thư viện chuẩn

Kiểu vector<T> (STL của C++)

- Do chịu ảnh hưởng của NNLT C nên NNLT C++ có những quy định không dễ dàng lắm về sử dụng dữ liệu động đối với người lập trình nhất là trường hợp mảng động nhiều chiều.
- Kiểu vector<T> trong thư viện chuẩn STL (Standard Template Library) phục vụ cho việc lập trình với dữ liệu động.

Kiểu vector<T> (STL của C++)

- Để lập trình với kiểu vector<T> của C++ STL, cần phải có các chỉ thị sau đây ở đầu chương trình: #include <vector> và using namespace std;
- Các phương thức thường dùng:
  - size(): trả về kích thước hiện hành của mảng.
  - resize(int newsize): thay đổi kích thước mảng.
  - push_back(T x): thêm phần tử x có kiểu T vào cuối mảng (mảng tự động thay đổi kích thước).
  - pop_back(): xóa phần tử cuối cùng của mảng (mảng tự động thay đổi kích thước).

Mảng động một chiều

- Kích thước mảng được xác định từ đầu

```
#include <vector>
#include <iostream>
using namespace std;
void main() {
	int i, n;
	vector<int> a;
	cout << "Nhap so luong phan tu: ";
	cin >> n;
	a.resize(n);
	for (i = 0; i < n; i++) {
		cout << "Nhap a[" << i << "]: ";
		cin >> a[i];
	}
}
```

Mảng động một chiều

- Kích thước mảng tự động điều chỉnh

```
#include <vector>
#include <iostream>
using namespace std;
void main() {
	int i, n, nTam;
	vector<int> a;
	cout << "Nhap so luong phan tu: ";
	cin >> n;
	for (i = 0; i < n; i++) {
		cout << "Nhap a[" << i << "]: ";
		cin >> nTam;
		a.push_back(nTam);
	}
}
```

Mảng động nhiều chiều

- Ví dụ mảng động 2 chiều

```
#include <vector>
#include <iostream>
using namespace std;
typedef vector<int> intArray;
void main() {
	int i, m, n;
	vector<intArray> a;
	cout << "Nhap so luong dong, cot: ";
	cin >> m >> n;
	a.resize(m);
	for (i = 0; i < m; i++)
		a[i].reisze(n);
}
```

Khái niệm về con trỏ

Khái niệm

- Bộ nhớ máy tính
  - Bộ nhớ RAM chứa rất nhiều ô nhớ, mỗi ô nhớ có kích thước 1 byte.
  - RAM dùng để chứa một phần hệ điều hành, các lệnh chương trình, các dữ liệu…
  - Mỗi ô nhớ có địa chỉ duy nhất và địa chỉ này được đánh số từ 0 trở đi.

Khái niệm

- Quy trình xử lý của trình biên dịch khi khai báo biến trong C
  - Dành riêng một vùng nhớ với địa chỉ duy nhất để lưu biến đó.
  - Liên kết địa chỉ ô nhớ đó với tên biến.
  - Khi gọi tên biến, nó sẽ truy xuất tự động đến ô nhớ đã liên kết với tên biến.
  - Biến con trỏ là biến chứa địa chỉ ô nhớ.

Khai báo và khởi tạo

- Khai báo
  - Giống như mọi biến khác, biến con trỏ muốn sử dụng cũng cần phải được khai báo.

<kiểu dữ liệu>* <tên biến con trỏ>;

- Khởi tạo
  - Con trỏ lưu địa chỉ của ô nhớ mà nó trỏ đến hoặc NULL nếu không trỏ đến đâu cả.
- Ví dụ

```
int n;
int* p1 = &n; int* p2 = NULL;
```

Các thao tác xử lý

- Các toán tử có thể thực hiện trên con trỏ:
  - Toán tử gán: =
  - Toán tử lấy địa chỉ: &
  - Toán tử lấy giá trị gián tiếp: *
  - Toán tử tăng và giảm: + và –
  - Toán tử lấy khoảng cách giữa 2 con trỏ: -
  - Toán tử so sánh: > >= < <= == !=

Một số nguyên tắc an toàn

- Nắm rõ quy tắc sau: int a, *pa = &a;
  - *pa và a đều chỉ nội dung của biến a.
  - pa và &a đều chỉ địa chỉ của biến a.
- Không nên sử dụng con trỏ khi chưa được khởi tạo, kết quả sẽ không lường trước được.

```
int* pa; *pa = 1904; // lỗi truy xuất bộ nhớ
```

Dữ liệu có cấu trúc (dạng struct hay union) và con trỏ

Dữ liệu struct với các trường tính theo bit

- Thành phần của cấu trúc có kích thước theo bit

```
struct bit_fields
{
        int bit_0 : 1;
        int bit_1_to_4 : 4;
        int bit_5 : 1;
        int bit_6_to_15 : 10;
};
```

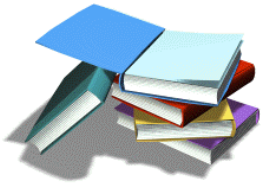
Cấp phát và sử dụng dữ liệu động

Cấp phát bộ nhớ động

- Trong C (sử dụng thư viện <stdlib.h> hoặc <alloc.h>)
  - malloc
  - calloc
  - realloc
  - free
- Trong C++
  - new
  - delete

Cấp phát bộ nhớ động

void *malloc(size_t size)



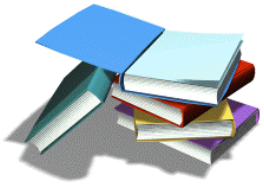
Cấp phát trong HEAP một vùng nhớ size (bytes)
size_t thay cho unsigned (trong <stddef.h>)

- Thành công: Con trỏ đến vùng nhớ mới được cấp phát.
- Thất bại: NULL (không đủ bộ nhớ).

```
int *p = (int *)malloc(10 * sizeof(int));
if (p == NULL)
        cout << "Khong du bo nho!";
```

Cấp phát bộ nhớ động

void *calloc(size_t num, size_t size)



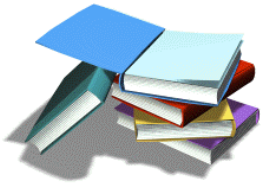
Cấp phát vùng nhớ gồm num phần tử trong HEAP, mỗi phần tử kích thước size (bytes)

- Thành công: Con trỏ đến vùng nhớ mới được cấp phát.
- Thất bại: NULL (không đủ bộ nhớ).

```
int *p = (int *)calloc(10, sizeof(int));
if (p == NULL)
        cout << "Khong du bo nho!";
```

Cấp phát bộ nhớ động

void *realloc(void *block, size_t size)



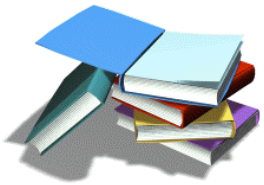
Cấp phát lại vùng nhớ có kích thước size do block trỏ đến trong vùng nhớ HEAP.
block == NULL ➔ sử dụng malloc
size == 0 ➔ sử dụng free

- Thành công: Con trỏ đến vùng nhớ mới được cấp phát.
- Thất bại: NULL (không đủ bộ nhớ).

```
int *p = (int *)malloc(10*sizeof(int));
p = (int *)realloc(p, 20*sizeof(int));
if (p == NULL)
        cout << "Khong du bo nho!";
```

Cấp phát bộ nhớ động

void free(void *ptr)



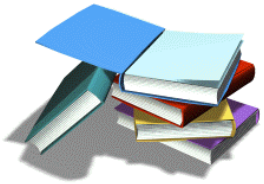
Giải phóng vùng nhớ do ptr trỏ đến, được cấp bởi các hàm malloc(), calloc(), realloc(). Nếu ptr là NULL thì không làm gì cả.

- Không có.

```
int *p = (int *)malloc(10*sizeof(int));
free(p);
```

Cấp phát bộ nhớ động

<pointer_to_datatype> = new <datatype>[size]



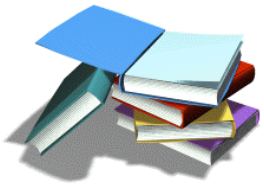
Cấp phát vùng nhớ có kích thước sizeof(<datatype>)*size trong HEAP

- Thành công: Con trỏ đến vùng nhớ mới được cấp phát.
- Thất bại: NULL (không đủ bộ nhớ).

```
int *a1 = (int *)malloc(sizeof(int));
int *a2 = new int;
int *p1 = (int *)malloc(10*sizeof(int));
int *p2 = new int[10];
```

Cấp phát bộ nhớ động

delete []<pointer_to_datatype>



Giải phóng vùng nhớ trong HEAP do <pointer_to_datatype> trỏ đến (được cấp phát bằng new)

- Không có.

```
int *a = new int;
delete a;
int *p = new int[10];
delete []p;
```

Các vấn đề tìm hiểu mở rộng kiến thức nghề nghiệp

Tìm hiểu thêm

- Con trỏ void.
- Con trỏ và tham chiếu.
- Từ khóa const và con trỏ.
- Liên hệ với các ngôn ngữ lập trình khác.

Thuật ngữ
và bài đọc thêm tiếng Anh

Thuật ngữ tiếng Anh

- ***pointer***: con trỏ.
- ***STL (Standard Template Library)***: thư viện chuẩn của C++.

Bài đọc thêm tiếng Anh

- **Theory and Problems of Fundamentals of Computing with C++**, John R.Hubbard, Schaum's Outlines Series, McGraw-Hill, 1998.