BÀI TẬP VỀ NHÀ
KĨ THUẬT LẬP TRÌNH

Phạm Văn Thông
MSSV: 20136495

Ngày 31 tháng 1 năm 2015

1 Cấp phát động trong C và C++

Trong lập trình thường xuyên phải làm việc với các mảng.Trong nhiều trường hợp, số lượng các phần tử của mảng là không biết trước, việc cấp phát tĩnh cho mảng thông qua khai báo ban đầu thường dẫn tới những trường hợp như không gian để lưu trữ dữ liệu hoặc dư thừa tài nguyên, chiếm dụng bộ nhớ của hệ thống. Giải pháp cho vấn đề này chính là cấp phát động.

1.1 Cấp phát động trong C

Trong C, ta có thể cấp phát động cho biến con trỏ bằng các hàm `malloc()`, `calloc()`, `realloc()` và giải phóng vùng nhớ được cấp phát động bằng hàm `free()`. Để sử dụng các hàm này phải khai báo tệp tiêu đề `stdlib.h`.

Cách sử dụng các hàm trên như sau:

malloc `void* malloc(size_t size)`

Hàm `malloc()` cấp phát (allocate) một vùng nhớ có kích thước `size` bytes và trả về con trỏ kiểu `void*` trỏ tới vùng nhớ được cấp phát. Vùng nhớ được cấp phát chưa được khởi tạo, nghĩa là *giá trị các byte nhớ trong vùng nhớ đó hoàn toàn là ngẫu nhiên*. Nếu giá trị của `size` là 0, hàm `malloc()` sẽ trả về `NULL` hoặc một con trỏ đặc biệt (unique pointer) có thể làm tham số cho hàm `free()`. Trả về `NULL` nếu thất bại.

calloc `void* calloc(size_t nmemb, size_t size)`

Hàm `calloc()` cấp phát cho một mảng `nmemb` phần tử, mỗi phần tử có kích thước `size` bytes và trả về con trỏ tới vùng nhớ đã được cấp phát. Chú ý rằng khi cấp phát bằng hàm calloc, *giá trị các bytes nhớ sẽ được đặt là 0*. Tương tự như `malloc()`, nếu `nmemb` hoặc `size` là 0, thì hàm `calloc()` sẽ trả về hoặc là `NULL`, hoặc một con trỏ đặc biệt có thể làm tham số cho hàm `free()`.

realloc `void* realloc(void *ptr, size_t size)`

Hàm `realloc()` thay đổi kích thước của khối nhớ được trỏ bởi `*ptr` thành `size` bytes. Nội dung của khối nhớ sẽ không thay đổi. Nếu kích thước mới lơn hơn kích thước ban đầu, phần khối nhớ thêm vào sẽ *không có giá trị xác định*. Nếu `ptr` là `NULL` thì hàm sẽ tương tự như sử dụng với hàm `malloc(size)`, với mọi giá trị bất kì của `size`. Nếu `size` là 0, và `ptr` không phải là `NULL`, thì lời gọi hàm sẽ tương đương với việc sử dụng hàm `free(ptr)`. Hàm trả về con trỏ tới khối nhớ vừa được cấp phát mới, hoặc `NULL` nếu thất bại. Nếu thất bại, khối nhớ sẽ không bị thay đổi (giải phóng hoặc di chuyển).

free `void free(void *ptr)`

Hàm `free()` giải phóng vùng nhớ được trỏ bởi `ptr`, `ptr` *phải là con trỏ được trả về từ các hàm* `malloc()`, `calloc()`, *hoặc* `realloc()`. Trong những trường hợp khác, hoặc `free(ptr)` đã được gọi từ trước thì sẽ có những lỗi không xác định [1] xảy ra. Nếu `ptr` là `NULL` thì không có thao tác nào được thực hiện.

Một số lưu ý:

- Khối nhớ được cấp phát từ hàm `malloc()` chưa được khởi tạo, các giá trị của nó là chưa xác định.
- Khối nhớ được cấp phát từ hàm `calloc()` thì các bytes nhớ của khối được đặt về 0.
- Với hàm `free()`, chỉ có thể hoạt động khi giá trị của con trỏ làm tham số cho nó là giá trị được trả về từ các hàm `malloc()`, `calloc()` và `realloc()`. Nếu vì một nguyên nhân nào đó mà các giá trị truyền vào cho free không phải là giá trị này thì sẽ gây ra lỗi. Ví dụ đoạn chương trình sau sẽ gây lỗi:

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

  Vì vậy khi lập trình phải chú ý với những con trỏ quản lý các khối nhớ được cấp phát động.
- Các khối nhớ được cấp phát nên được giải phóng khi không cần sử dụng đến. Trước khi kết thúc chương trình nên giải phóng hết các bộ nhớ, nếu không có thể gây ra hiện tượng rác.
- Khi sử dụng các hàm `malloc()`, `calloc()` và `realloc()` không nên ép kiểu, việc ép kiểu có thể dẫn đến gây lỗi nhưng trình dịch không thông báo trong trường hợp không `include` thành công thư viện `stdlib`.

[1] undefined behavior

1.2 Cấp phát động trong C++

Trong C++ sử dụng các toán tử `new` và `delete` dùng cho việc cấp phát và giải phóng bộ nhớ. Việc cấp phát và giải phóng bộ nhớ sử dụng các toán tử `new` và `delete` rất đơn giản.

Các cú pháp sử dụng các toán tử `new` và `delete` như sau:

- Cấp phát bộ nhớ
  `<tên con trỏ> = new <kiểu con trỏ>`
- Cấp phát cho mảng
  `<tên con trỏ> = new <kiểu con trỏ>[số phần tử]`
- Giải phóng bộ nhớ
  `delete <tên con trỏ>`
- Giải phóng bộ nhớ cho mảng
  `delete [] <tên con trỏ>`

Toán tử `new` sẽ gọi tới hàm dựng lớp.

1.3 Một số sự khác biệt giữa `new`, `delete` so với `malloc` và `free`

Các hàm như `malloc()` và `free()` so với các toán tử `new` và `delete` tuy cùng được sử dụng để cấp phát động trong lập trình C/C++ nhưng giữa chúng có nhiều điểm khác nhau.

- `new` và `delete` chỉ dùng được trong C++, `malloc()` và `free()` dùng được trong cả C và C++
- `new` là toán tử, có thể *overload operator* còn `malloc()` là hàm
- `new` trả về con trỏ của loại dữ liệu, còn `malloc()` trả về con trỏ `void`
- `new` gọi hàm dựng lớp
- Khi gặp vấn đề, `new` quăng ra lỗi (`throw exception`), còn `malloc()` chỉ trả về `NULL`
- Khi giải phóng `new` phải sử dụng `delete`, khi giải phóng `malloc()` phải sử dụng `free()`