# Pointer and Memory Management

## Con trỏ, tham chiếu, tham trị

Con trỏ ( hay thường gọi là biến con trỏ) là biến mà giá trị của nó là địa chỉ của biến khác

Cách thức khai báo con trỏ: <KDL>\* name\_of\_pointer;

Có 3 cách truyền tham số trong lập trình C++: Truyền bởi tham trị, truyền bởi tham chiếu và truyền bằng con trỏ.

Tham số truyền bằng tham trị: Không thực hiện thay đổi được giá trị của biến truyền vào. Đối số truyền vào có thể là 1 hằng, một biến. Khi thực hiện lời gọi hàm, một bản sao của đối số được tạo ra và sẽ biến mất sau khi hàm chạy xong.

Tham số truyền bằng tham chiếu: Hữu ích khi truyền tham số vào và có thay đổi giá trị của nó. Và trực tiếp truy cập vào biến đối số được truyền vào. Đối số truyền vào phải là một biến.

Tham số truyền bằng địa chỉ (tham số truyền vào là con trỏ): Hữu ích khi truyền vào một con trỏ đang giữ một địa chỉ của một biến. Khi thực hiện lời gọi hàm, giá trị của con trỏ sẽ được sao chép và nó cũng được nếu ta thay đổi giá trị con trỏ thì thì chỉ thay đổi bản sao của con trỏ. Trên thực tế, con trỏ truyền vào sẽ không bị ảnh hưởng.

Dùng tham chiếu khi tham chiếu khi có thể, dùng truyền bằng địa chỉ khi cần thiết có thể.

## Vùng nhớ Stack và Heap

Trong C++ , dãy địa chỉ bộ nhớ ảo được chia thành nhiều phân vùng khác nhau để sử dụng cho những mục đích khác nhau. Sau đây là hình ảnh minh họa cho thứ tự các phân vùng trên bộ nhớ ảo theo thứ tự địa chỉ liên tiếp:



Code segment: Lưu trữ các mã lệnh đã được biên dịch của chương trình máy tính. Sẽ được chuyển tới CPU xử lý khi cần thiết. Code segment chỉ chịu sự chi phối của hệ điều hành, các phân vùng còn lại không can thiệp được vào phân vùng này.

Data segment: Là phân vùng hệ điều hành sử dụng để khởi tạo giá trị cho các biến kiểu static, biến toàn cục của chương trình

Heap segment: Sử dụng để cấp phát bộ nhớ động thông qua kỹ thuật Dynamic memory allocation. Bộ nhớ sẽ tự động được giải phóng sau khi kết thúc chương trình. Nên nếu chương trình chạy quá lâu và vùng nhớ này không được giải phóng một cách hợp lý sẽ làm ảnh hưởng tới cấp phát động cho các chương trình khác.

Ưu nhược điểm của phân vùng Heap:

* Việc cấp phát bộ nhớ trên Heap chậm hơn các phân vùng khác.
* Vùng nhớ đã được cấp phát sẽ vẫn thuộc quyền kiểm soát của chương trình đang chạy cho đến khi chúng được giải phóng, hoặc nhận được tín hiệu kết thúc chương trình.
* Vùng nhớ được cấp phát phải được quản lý bởi ít nhất 1 con trỏ.
* Toán tử dereference truy xuất đến vùng nhớ chậm hơn các biến thông thường.
* Phân vùng Heap có dung lượng lớn nhất, nên chúng ta có thể sử dụng một cách thoải mái hơn các phân vùng khác.

Stack segment: được dùng để cấp phát bộ nhớ cho tham số của hàm, biến cục bộ. Vùng nhớ này được triển khai theo cấu trúc dữ liệu stack(LIFO – vào sau ra trước). Khi phân vùng này đầy, chương trình sẽ bị lỗi Stack overflow( vượt quá kích thước cung cấp bộ nhớ cho phép)

Một số ưu và nhược điểm

* Việc cấp phát bộ nhớ trên Call Stack khá nhanh.
* Nhìn vào mã nguồn chương trình, chúng ta có thể biết được thời điểm cấp phát và hủy vùng nhớ của biến trên Stack.
* Kích thước vùng nhớ cấp phát trên phân vùng Stack phải được khai báo rõ ràng trước khi biên dịch.
* Vùng nhớ trên phân vùng Stack có thể được truy cập trực tiếp thông qua định danh.
* Kích thước của phân vùng Stack khá hạn chế.

## Bad-pointer, memory-leak

Bad – pointer là tên gọi khi con trỏ trỏ đến vùng nhớ nằm ngoài phạm vi chương trình đang quản lý. Giải quyết: cho con trỏ trỏ về NULL (00000000)

Memory Leak là tên gọi khi chương trình cấp phát không gian nhớ nhưng không được dùng hoặc không được giải phóng trong suốt thời gian chạy chương trình. Điều này gây lãng phí không gian nhớ. C++ không tự giải phóng vùng nhớ này mà ta phải thủ công thực hiện thông qua toán tử delete

## Rules for using pointer safely

## Implementation for single Linked List