KIẾN THỨC
STATIC AND DYNAMIC MEMORY ALLOCATION IN C

(Tài liệu sử dụng trong buổi đào tạo c/c++ - 2022)

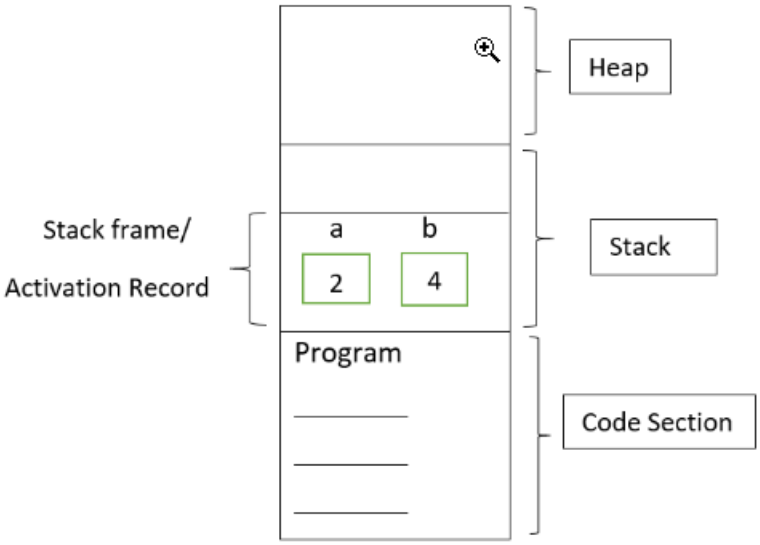
Tài liệu này được sử dụng trong buổi đào tạo và dưới đây là chi tiết các nội dung mà học viên cần nắm rõ để thực hiện. Gồm có:

Bộ nhớ được chia thành các đơn vị địa chỉ nhỏ hơn được gọi là byte. Giả sử rằng đây là những hộp nhỏ dưới dạng byte. Mỗi byte có địa chỉ riêng theo bảng bên dưới. Ví dụ: 0, 1, 2, 3, 4, 5, 6, v.v.

..	..	..	..	..	..
..	..	..	..	..	..
..	..	..	..	..	..
12	13	14	15	16	..
6	7	8	9	10	11
0	1	2	3	4	5

Chương trình sử dụng bộ nhớ như thế nào?

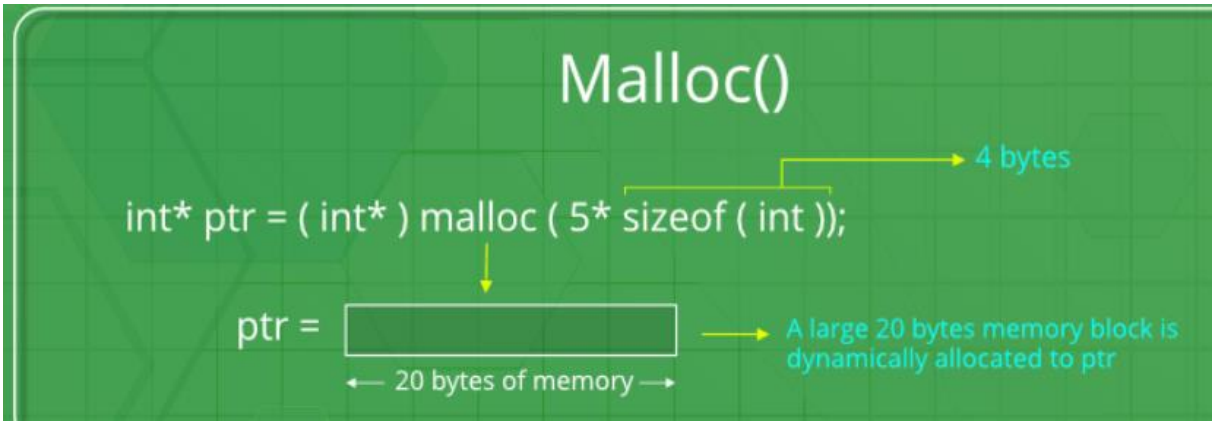
- Bộ nhớ được chia thành 3 phần.
  - Heap Memory
  - Stack Memory
  - Code Section
- Chi tiết:
  - ***Heap Memory***: Nó là một phần của bộ nhớ chính. Bộ nhớ heap không thể được sử dụng trực tiếp với sự trợ giúp của con trỏ.
  - ***Stack Memory***: Nó lưu trữ các biến tạm thời được tạo bởi một hàm. Trong ngăn xếp, các biến được khai báo, lưu trữ và khởi tạo trong thời gian chạy. Nó tuân theo phương thức ***First in last out*** có nghĩa là bất kỳ phần tử nào sẽ được lưu trữ sau cùng sẽ bị xóa trước khi nó không được sử dụng.
  - ***Code Section***: Bất cứ khi nào chương trình được thực thi, nó sẽ được đưa vào bộ nhớ chính. Chương trình này sẽ được lưu trữ trong phần mã. Dựa trên chương trình, nó sẽ quyết định nên sử dụng các phần Stack hay Heap.



- Có một số chức năng có sẵn trong ***stdlib.h*** sẽ giúp cấp phát bộ nhớ động.
- ***malloc()***: Chức năng cấp phát bộ nhớ đơn giản nhất trong thời gian chạy được gọi là malloc(). Cần chỉ định số byte bộ nhớ được yêu cầu cấp phát vì đối số trả về địa chỉ của byte bộ nhớ đầu tiên được cấp phát vì bạn nhận được một địa chỉ được trả về, một con trỏ là nơi duy nhất để đặt nó.
- Cú pháp: ***int *p = (int*)malloc(No of values*size(int));***
- *Ví dụ:*

  ptr = (int*) malloc(100 * sizeof(int));

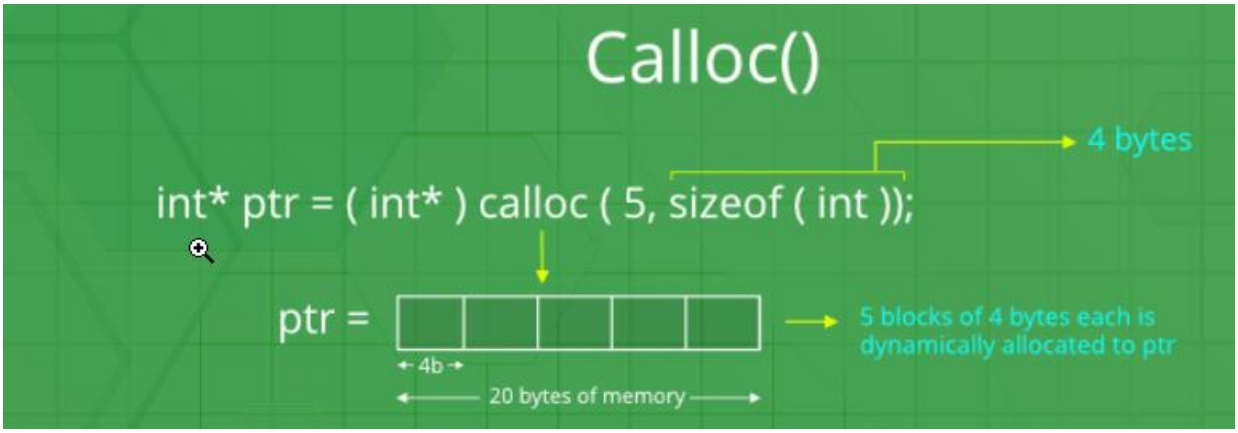
  Vì kích thước của int là 4 byte nên câu lệnh này sẽ phân bổ 400 byte bộ nhớ. Và, con trỏ ptr giữ địa chỉ của byte đầu tiên trong bộ nhớ được cấp phát.



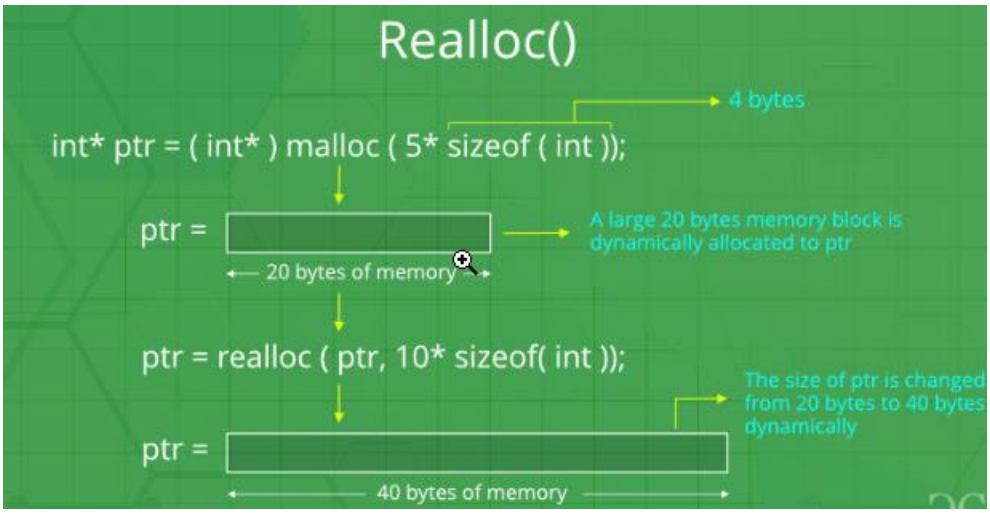
- Đối số cho malloc() ở trên chỉ ra rõ ràng rằng phải cung cấp đủ byte để cung cấp số lượng giá trị của loại int. Cũng lưu ý ép kiểu (int*), chuyển đổi địa chỉ được trả về bởi hàm thành con trỏ kiểu thành int. hàm malloc() trả về một con trỏ có giá trị NULL.
- calloc(): Hàm calloc() cung cấp một vài ưu điểm so với malloc(). Nó phân bổ bộ nhớ dưới dạng một số phần tử có kích thước nhất định. Nó khởi tạo bộ nhớ được cấp phát sao cho tất cả các byte đều bằng 0. hàm calloc() yêu cầu hai giá trị đối số:
    - Số lượng mục dữ liệu mà không gian được yêu cầu.
    - Kích thước của mỗi mục dữ liệu.
- Nó rất giống với việc sử dụng malloc() nhưng điểm cộng lớn là bạn biết vùng bộ nhớ sẽ được khởi tạo bằng 0.
- Cú pháp: ***int *p = (int*)calloc(Number of data items, sizeof(int));***
- Ví dụ:

  ptr = (float*) calloc(25, sizeof(float));

  Câu lệnh này phân bổ không gian liền kề trong bộ nhớ cho 25 phần tử, mỗi phần tử có kích thước bằng float.



- ***realloc()***: Hàm realloc() cho phép bạn sử dụng lại hoặc mở rộng bộ nhớ mà bạn đã cấp phát trước đó bằng cách sử dụng malloc() hoặc calloc(). Một con trỏ chứa địa chỉ đã được trả về trước đó bởi lệnh gọi malloc(), calloc(). Kích thước tính bằng byte của bộ nhớ mới cần được cấp phát. Nó cấp phát bộ nhớ được chỉ định bởi đối số thứ hai và chuyển nội dung của bộ nhớ đã cấp phát trước đó được tham chiếu bởi con trỏ được truyền dưới dạng đối số thứ nhất sang bộ nhớ mới được cấp phát.



- Cú pháp: ***int *np = (type cast) realloc (previous pointer type, new number of elements * sizeof(int));***
- ***free()***: Khi bộ nhớ được cấp phát động, nó sẽ luôn được giải phóng khi không còn cần thiết. Bộ nhớ được phân bổ trên heap sẽ tự động được giải phóng khi chương trình kết thúc nhưng tốt hơn hết là giải phóng bộ nhớ một cách rõ ràng khi hoàn thành nó, ngay cả khi nó ngay trước khi thoát khỏi chương trình. Xảy ra rò rỉ bộ nhớ, bộ nhớ được cấp phát động và tham chiếu đến nó không được giữ lại, do đó không thể giải phóng bộ nhớ.
- Cú pháp: ***free(pointer);***