**Phân vùng nhớ**

* **Text :**  
  – Quyền truy cập chỉ Read và nó chứa lệnh để thực thi nên tránh sửa đổi instruction.  
  – Chứa khai báo hằng số trong chương trình (.rodata)
* **Data**:  
  – Quyền truy cập là read-write.  
  – Chứa biến toàn cục (global) or biến static với giá trị khởi tạo khác không.  
  – Được giải phóng khi kết thúc chương trình.
* **Bss**:  
  – Quyền truy cập là read-write.  
  – Chứa biến toàn cục (global) or biến static với giá trị khởi tạo bằng không or không khởi tạo.  
  – Được giải phóng khi kết thúc chương trình.
* **Stack**:  
  – Quyền truy cập là read-write.  
  – Được sử dụng cấp phát cho biến local, input parameter của hàm,…  
  – Sẽ được giải phóng khi ra khỏi block code/hàm
* **Heap**:  
  – Quyền truy cập là read-write.  
  – Được sử dụng để cấp phát bộ nhớ động như: Malloc, Calloc,   
  – Sẽ được giải phóng khi gọi hàm free,…

Graphical user interface, application, Word

Description automatically generated

Diagram

Description automatically generated

**2. Stack và Heap?**

* Bộ nhớ Heap và bộ nhớ Stack bản chất đều cùng là vùng nhớ được tạo ra và lưu trữ trong RAM khi chương trình được thực thi.
* Bộ nhớ Stack được dùng để lưu trữ các biến cục bộ trong hàm, tham số truyền vào... Truy cập vào bộ nhớ này rất nhanh và được thực thi khi chương trình được biên dịch.
* Bộ nhớ Heap được dùng để lưu trữ vùng nhớ cho những biến con trỏ được cấp phát động bởi các hàm *malloc - calloc - realloc* (trong C)
* *Kích thước vùng nhớ*  
  Stack: kích thước của bộ nhớ Stack là cố định, tùy thuộc vào từng hệ điều hành, ví dụ hệ điều hành Windows là 1 MB, hệ điều hành Linux là 8 MB (lưu ý là con số có thể khác tùy thuộc vào kiến trúc hệ điều hành của bạn).  
  Heap: kích thước của bộ nhớ Heap là không cố định, có thể tăng giảm do đó đáp ứng được nhu cầu lưu trữ dữ liệu của chương trình.
* *Đặc điểm vùng nhớ*  
  Stack: vùng nhớ Stack được quản lý bởi hệ điều hành, dữ liệu được lưu trong Stack sẽ tự động hủy khi hàm thực hiện xong công việc của mình.

Heap: Vùng nhớ Heap được quản lý bởi lập trình viên (trong C hoặc C++), dữ liệu trong Heap sẽ không bị hủy khi hàm thực hiện xong, điều đó có nghĩa bạn phải tự tay hủy vùng nhớ bằng câu lệnh free (trong C), và delete hoặc delete [ ] (trong C++), nếu không sẽ xảy ra hiện tượng rò rỉ bộ nhớ.

Lưu ý: việc tự động dọn vùng nhớ còn tùy thuộc vào trình biên dịch trung gian.

* *Vấn đề lỗi xảy ra đối với vùng nhớ*  
  Stack: bởi vì bộ nhớ Stack cố định nên nếu chương trình bạn sử dụng quá nhiều bộ nhớ vượt quá khả năng lưu trữ của Stack chắc chắn sẽ xảy ra tình trạng tràn bộ nhớ Stack (Stack overflow), các trường hợp xảy ra như bạn khởi tạo quá nhiều biến cục bộ, hàm đệ quy vô hạn,...  
  Ví dụ về tràn bộ nhớ Stack với hàm đệ quy vô hạn:

int foo(int x){

printf("De quy khong gioi han\n");

return foo(x);

}

Heap: Nếu bạn liên tục cấp phát vùng nhớ mà không giải phóng thì sẽ bị lỗi tràn vùng nhớ Heap (Heap overflow).  
Nếu bạn khởi tạo một vùng nhớ quá lớn mà vùng nhớ Heap không thể lưu trữ một lần được sẽ bị lỗi khởi tạo vùng nhớ Heap thất bại.  
Ví dụ trường hợp khởi tạo vùng nhớ Heap quá lớn:

int \*A = (int \*)malloc(18446744073709551615);