# **Bài 10. Struct**

**Mục tiêu:**

1. *Luyện tập định nghĩa và sử dụng kiểu dữ liệu có cấu trúc struct*
2. *Thử nghiệm cấu trúc bộ nhớ và việc truyền tham số, trả về giá trị kiểu struct.*

***Giới hạn:*** *không dùng các thư viện stl (chẳng hạn vector, algorithm)*

**Yêu cầu nộp bài** *: Toàn bộ phần A và C1*

*Bài này đánh giá theo cố gắng làm việc. Bạn cần làm đủ. Chương trình của bạn có thể chưa chạy đúng, thậm chí chưa chạy, nhưng vẫn có thể được tính là có làm và đạt.*

***Lưu ý: chép bài hoặc cho chép bài sẽ dẫn đến trượt môn học!***

## Thực hành

1. ***Định nghĩa và sử dụng.*** Hãy định nghĩa cấu trúc Point đại diện cho một điểm trong không gian 2 chiều (gồm hai trường x và y đại diện cho tọa độ x và tọa độ y trên mặt phẳng). Viết một hàm print() nhận tham số là một biến kiểu Point và in ra tọa độ điểm đó dưới dạng (x, y). Demo sử dụng hàm đó tại hàm main.
2. class *Point*
3. {
4. int x, y;
5. public:
6. Point() {
7. x = 20;
8. y = 45;
9. }
10. void print() const {
11. std::cout << "(" << x << ", " << y << ")\n";
12. }
13. }
14. ***Tham trị và tham biến.*** Viết chương trình thử nghiệm truyền tham số kiểu Point vào hàm theo hai kiểu truyền tham trị và truyền tham biến. Hãy dùng phương pháp in địa chỉ của biến kiểu Point để chứng tỏ rằng khi truyền tham trị ta có tham số là một bản sao của đối số (khác địa chỉ), còn khi truyền tham chiếu, ta có tham số và đối số chính là một biến (trùng địa chỉ).
15. void temp1(*Point* *p*)
16. {
17. std::cout << "truyen tham tri. dia chi cua tham so la: " << &*p* << '\n';
18. }
19. void temp2(*Point*& *p*)
20. {
21. std::cout << "truyen tham chieu. dia chi cua tham so la: " << &*p* << '\n';
22. }
23. int main()
24. {
25. *Point* p;
26. p.print();
27. std::cout << &p << '\n';
28. temp1(p);
29. temp2(p);
30. }
31. ***Giá trị trả về.*** Viết một hàm mid\_point() nhận 02 tham số kiểu Point và trả về một cấu trúc Point là trung điểm của đoạn thẳng nối hai điểm đó. Hãy sử dụng const để đảm bảo hàm mid\_point không sửa giá trị của hai điểm được truyền vào.
32. *Point* mid\_point(const *Point* *p1*, const *Point* *p2*)
33. {
34. *Point* mid;
35. mid.x = (*p1*.x + *p2*.x) / 2;
36. mid.y = (*p1*.y + *p2*.y) / 2 ;
37. return mid;
38. }
39. ***Cấu trúc bộ nhớ của struct***; Với cấu trúc Point đã định nghĩa ở trên. Hãy dùng phương pháp in địa chỉ của biến để thử nghiệm tìm hiểu vị trí tương đối giữa các trường x, y của một biến kiểu Point và chính biến đó. Bạn có nhận xét gì?

Memory access granualarity : thay vì đọc và ghi từng ô nhớ thì CPU sẽ truy xuất bộ nhớ theo từng Block có thể có kích thước 2,4, 8, 16 tùy vào CPU. Kích thước này gọi là MAG

CPU chỉ có thể đọc trực tiếp được các ô nhớ có dữ liệu chia hết cho kích thước của block

Địa chỉ của struct bắt đầu từ 0

Địa chỉ của 1 biến trong struct luôn chia hết cho kích thước của biến đó để dễ truy xuất

1. ***Phép gán và copy nông***. Bạn đã biết rằng với phép gán, C++ chỉ sao chép nông, có nghĩa là nếu một trường của biến struct là con trỏ thì trường tương ứng của bản sao của biến struct đó là một con trỏ có giá trị giống hệt. Có nghĩa là hai biến struct đó có con trỏ trỏ tới cùng một vùng dữ liệu. Hãy thử nghiệm để xem chuyện gì xảy ra khi biến struct được sao chép có trường dữ liệu là mảng, là một struct khác.

## Câu hỏi

## Bài tập

1. ***Xử lý xâu.*** Tương tự định nghĩa cấu trúc Array trong bài giảng, hãy xây dựng cấu trúc **String** để lưu trữ một xâu kí tự.Mô tả như sau:
   1. Cấu trúc này cần có hai biến thành viên, một biến ***n*** lưu độ dài xâu, một con trỏ ***str*** chiếu tới mảng char được cấp phát động.
   2. Một constructor với tham số kiểu const char\* để có thể khởi tạo một biến kiểu String bằng lệnh như:

String s(“Hello”);

Constructor có nhiệm vụ cấp phát mảng động với kích thước thích hợp và chép nội dung xâu kí tự được cho vào mảng đó.

* 1. Destructor có nhiệm vụ giải phóng mảng động str.
  2. Hàm thành viên ***void print()*** với nhiệm vụ in nội dung xâu str ra màn hình
  3. Hàm thành viên ***void append(const char\* )*** với tham số là một hằng xâu kí tự và có nhiệm vụ nối thêm xâu kí tự đó vào cuối xâu. Ví dụ, sau đoạn lệnh sau thì greeting chứa nội dung “Hi there”:

String greeting(“Hi”);

greeting.append(“ there”);   
 // đến đây nếu chạy *greeting.print()* ta sẽ được output “Hi there”.

Gợi ý: Để làm việc này, bạn cần làm các bước sau: (1) dùng một con trỏ tạm temp để cấp phát một mảng động mới có kích thước đủ lớn, (2) copy xâu str hiện hành vào phần đầu mảng mà temp đang trỏ tới, (3) copy xâu tham số vào phần sau. Đến giờ mảng động temp đã chứa nội dung đầy đủ của xâu kết quả. Việc còn lại là trỏ str tới mảng đó.Bạn hãy tự làm tiếp.

**Chú ý** tránh thất thoát bộ nhớ, cần giải phóng vùng bộ nhớ đã từng cấp phát nhưng không còn cần dùng đến nữa.

Với mỗi hàm, cần demo tại hàm main của chương trình.

**Chú ý: dùng const ở bất cứ chỗ nào có thể.**

#include <iostream>

#include <cstring>

#include <string>

class *String* {

  int length;

  char\* p = **new** char[length];

public:

  String(const char\* *name*) {

    length = strlen(*name*);

    strcpy(p, *name*);

  }

  ~String() { **delete[]** p; }

  void print const ()

  {

    std::cout << p << '\n';

  }

  void append(const char\* *str*) {

    strcat(p, str);

  }

};

int main()

{

*String* s("hello");

  s.append(" world");

  s.print();

}