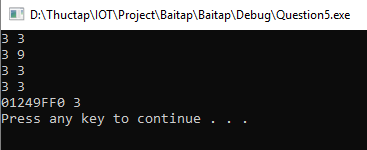
Question 5

Q 5.1

Kết quả in ra



Giải thích:

Cấp phát động tạo vung nhớ tại heap.

Con trỏ \*ptr\_a = 3 => vùng nhớ heap có giá trị bằng 3, ptr\_b = ptr\_a = 3 => in ra kết quả 3 3 từ địa chỉ ô nhớ của các biến ptr\_a, ptr\_b.

Cấp phát động tạo cùng nhớ tiếp theo tại heap.

Con trỏ \*ptr\_b = 9 => vùng nhớ heap tiếp theo có giá trị bằng 9, => in ra kết quả 3 9 từ địa chỉ ô nhớ của các biến ptr\_a, ptr\_b tại các vùng nhớ heap trỏ đến

\*ptr\_b = \*ptr\_a => vùng nhớ heap có địa chỉ ptr\_b bây giờ bằng 3, => in ra kết quả 3 3

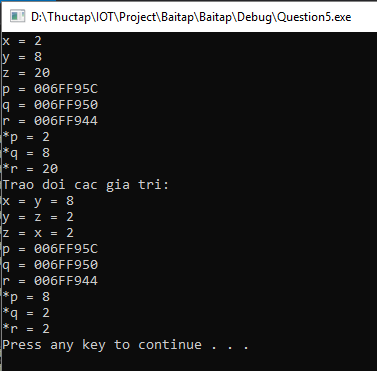
delete ptr\_a sẽ giải phóng vùng nhớ tại heap của prt\_a nhưng không giải phóng vùng nhớ heap của ptr\_b, sau đó , ptr\_a = ptr\_b => in ra kết quả là 3 3 từ địa chỉ ô nhớ các biến ptr\_a, ptr\_b.

ptr\_c = &ptr\_a => in ra kết quả là 01249FF0 3 do vùng nhớ tại heap của ptr\_a bị xóa nên chỉ in ra địa chỉ của vùng nhớ của ptr\_a, còn \*\*ptr\_c có giá trị vùng nhớ của ptr\_a = 3.

Q 5.2

Q 5.3

Kết quả:



Sơ đồ:

Giải thích:

x, y, z, là các giá trị gán từ ban đầu.

p, q, r là các địa chỉ ô nhớ của x, y, z

\*p, \*q, \*r là các giá trị mà con trỏ trỏ đến địa chỉ ô nhớ

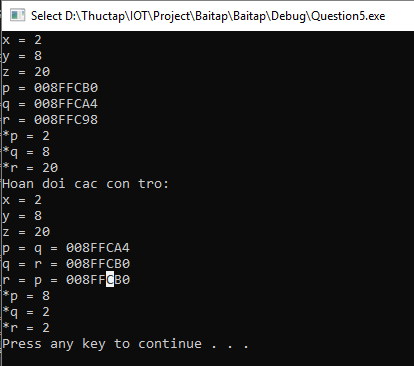
Sau khi trao đổi giá trị :

z = x =2, x = y = 8, y = z = 2 => giá trị của các ô nhớ ban đầu bị thay đổi khi các biến x, y, z trao đổi lẫn nhau

tương tự với p, q, r và \*p, \*q, \*r như trên không thay đổi

Q 5.4

Kết quả



Giải thích :

x, y, z, là các giá trị gán từ ban đầu.

p, q, r là các địa chỉ ô nhớ của x, y, z

\*p, \*q, \*r là các giá trị mà con trỏ trỏ đến địa chỉ ô nhớ

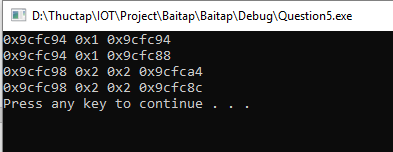
Sau khi trao đổi con trỏ :

Thì giá trị của các biến không thay đổi khi trao đổi con trỏ với nhau vì con trỏ chỉ thay đổi địa chỉ ô nhớ chỉ tới chứ không thay đổi giá trị ô nhớ

p, q, r bị thay đổi nơi chỉ tới ô nhớ của nhau khi bị trao đổi với nhau dẫn tới \*p, \*q, \*r cũng thay đổi giá trị do địa chỉ trỏ tới bị thay đổi.

Q 5.5

Kết quả



Giải thích :

a là địa chỉ ô nhớ vùng source code của phần tử thứ nhất a[0] của mảng a[]

\*a là giá trị của con trỏ a chỉ từ địa chỉ của phần tử thứ nhất a[0] của mảng a[]

&a là địa chỉ tuyệt đối vùng ô nhớ source code của phần tử thứ nhất a[0] của mảng a[]

\*p = a nên

p là địa chỉ ô nhớ vùng source code của phần tử thứ nhất a[0] của mảng a[] do con trỏ p chỉ đến

\*p là giá trị của con trỏ p chỉ từ địa chỉ của phần tử thứ nhất a[0] của mảng a[]

&p là địa chỉ tuyệt đối vùng ô nhớ source code của con trỏ p.

(a + 1) là địa chỉ ô nhớ vùng source code của phần tử thứ hai a[1] của mảng a[]

\*(a + 1) là giá trị của con trỏ a chỉ từ địa chỉ của phần tử thứ hai a[1] của mảng a[]

\*a +1 là giá trị 4 byte tiếp theo của con trỏ a từ địa chỉ của phần tử thứ nhất a[0] của mảng a[]

&a + 1 là địa chỉ tuyệt đối 16 byte tiếp theo thoát khỏi mảng vùng ô nhớ source code

(p + 1) là địa chỉ ô nhớ vùng source code của phần tử thứ hai a[1] của mảng a[] do con trỏ (p + 1) chỉ đến

\*(p + 1) là giá trị của con trỏ \*(p + 1) chỉ từ địa chỉ của phần tử thứ hai a[1] của mảng a[]

\*p +1 là giá trị 4 byte tiếp theo của con trỏ p từ địa chỉ của phần tử thứ nhất a[0] của mảng a[]

&a + 1 là địa chỉ tuyệt đối 4 byte tiếp theo của con trỏ p tại vùng ô nhớ source code

Q 5.6

Vấn đề của chương trình là khi delete p thì nó xóa luôn mảng a mà không chờ đến delete a.

Cách giải quyết :

Q 5.7

* Tại sao chúng ta nên sử dụng delete là do khi cấp phát động nó sẽ lưu và không xóa trong khi chương trình chạy dẫn đến gây tốn bộ nhớ. Nếu cấp phát tĩnh thì không cần delete vì trong quá trình chạy sẽ tự động giải phóng khi nó làm xong việc
* Khi nào chúng ta nên sử dụng delete là khi chúng ta cấp phát động nên delete khi dùng xong để giải phóng bộ nhớ cấp phát
* Các delete toán tử deallocates bộ nhớ và gọi destructor cho một đối tượng duy nhất được tạo ra với new.

Các delete [] toán tử deallocates bộ nhớ và gọi hàm hủy đối với một mảng của các đối tượng được tạo ra với new [].

Sử dụng deletetrên một con trỏ được trả về bởi new []hoặc delete []trên một con trỏ được trả về bởi newkết quả trong hành vi không xác định.

Q 5.8

Source code edit :

#include <iostream>

using namespace std;

#define COUNT 10

#define MIN(a, b) ((a) < (b) ? (a):(b))

void main()

{

int \*p = new int[COUNT];

int a[COUNT];

for (int i = 0; i < COUNT; i++)

{

a[i] = i;

}

for (int i = 0; i < COUNT; i++)

{

p[i] = i;

}

cout << sizeof(a) << endl;

cout << sizeof(p) << endl;

for (int i = 0; i < MIN(sizeof(a), sizeof(p)); i++)

{

printf("%d %d\n", a[i], p[i]);

}

system("pause");

}

Q 5.9

Bị lỗi bộ nhớ.