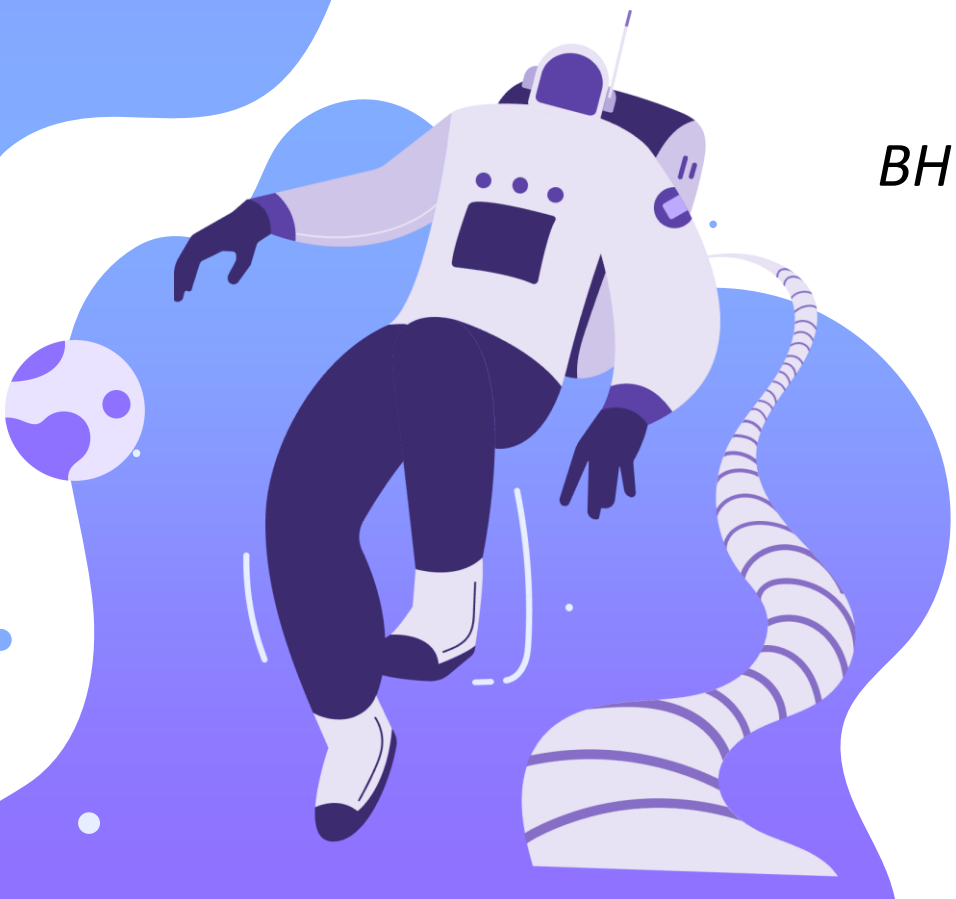


NC LEARNING SPACE
Ban Học tập Đoàn khoa Mạng máy tính và Truyền thông

Training Nhập Môn Lập Trình

BHT Đoàn khoa MMT&TT – Training cuối kì I K16

Trainers: Tống Võ Anh Thuận – MMCL 2021

Lê Khắc Trung Nam – ATCL 2021

Phạm Nguyễn Hải Anh – ATTT 2021

NỘI DUNG

TOPIC 01

Kiểu cấu trúc - Struct

TOPIC 02

Con trỏ - Pointer

NỘI DUNG

TOPIC 01

Kiểu cấu trúc - Struct

TOPIC 02

Con trỏ - Pointer

TOPIC 01: Kiểu cấu trúc - Struct

Đặt vấn đề.

Thông tin 1 sinh viên (SV) như sau:

- MSSV - kiểu chuỗi
- Tên SV - kiểu chuỗi
- Ngày tháng năm sinh - kiểu chuỗi
- Giới tính - ký tự
- Điểm toán, lý, hóa - số thực

Yêu cầu.

- Lưu thông tin cho N sinh viên.
- Truyền thông tin N sinh viên vào một hàm

TOPIC 01: Kiểu cấu trúc - Struct

Giải quyết vấn đề (cách thứ nhất).

```
1  char mssv[7];
2  char hoTen[40];
3  char ntns[8];
4  float toan[100]
5  float ly[100]
6  float hoa[100];
7
8  void nhapThongTin(char mssv[],char hoTen[],char ntns[], float toan[], float ly[],float hoa[]);
9  void xuatThongTin(char mssv[],char hoTen[],char ntns[], float toan[], float ly[],float hoa[]);
```


TOPIC 01: Kiểu cấu trúc - Struct

Nhận xét.

- Đặt tên biến khó khăn và khó quản lý.
- Truyền tham số cho hàm quá nhiều.
- Tìm kiếm, sắp xếp, sao chép,... khó khăn.
- Tốn nhiều bộ nhớ.

=> Gom những thông tin của cùng 1 sinh viên thành một kiểu dữ liệu mới.

TOPIC 01: Kiểu cấu trúc - Struct

Khái niệm về kiểu struct.

- Struct (cấu trúc) là kiểu dữ liệu do người dùng định nghĩa, được tạo thành từ các kiểu dữ liệu được xây dựng sẵn trong C++.

- Một struct cho phép chúng ta nhóm nhiều biến của nhiều kiểu dữ liệu khác nhau để lưu trữ một tập hợp các dữ liệu cần thiết cho việc mô tả một đơn vị nào đó.

TOPIC 01: Kiểu cấu trúc - Struct

Cú pháp khai báo struct.

```
1  struct sinhVien{
2      char mssv[7], hoTen[40], ntns[8];
3      float toan,ly,hoa;
4  };
```

TOPIC 01: Kiểu cấu trúc - Struct

Cú pháp khai báo biến struct.

```
1  struct sinhVien{
2      char mssv[7], hoTen[40], ntns[8];
3      float toan, ly, hoa;
4  };
5  sinhVien SV1, SV2;
```

```
1  struct sinhVien{
2      char mssv[7], hoTen[40], ntns[8];
3      float toan, ly, hoa;
4  } SV1, SV2;
```

TOPIC 01: Kiểu cấu trúc - Struct

Cú pháp khai báo biến kiểu struct.

```
1  struct toaDo{
2      int x;
3      int y;
4  };
5  int main(){
6      toaDo hinhVuong[4];
7  }
```

TOPIC 01: Kiểu cấu trúc - Struct

Cú pháp khởi tạo cho biến kiểu struct.

```
1  struct toaDo{
2      int x;
3      int y;
4      int z;
5  };
6  toaDo diemA = {3,4,5}, diemB = {0,1,0};
```

TOPIC 01: Kiểu cấu trúc - Struct

Truy xuất dữ liệu kiểu struct.

```
1  struct toaDo{
2      int x;
3      int y;
4      int z;
5  };
6  int main(){
7      toaDo diemA = {3,4,5}, diemB = {0,1,0};
8      cout << "Toa do diem A la: " << diemA.x << ";" << diemA.y << ";" << diemA.z;
9  }
```

TOPIC 01: Kiểu cấu trúc - Struct

Gán dữ liệu kiểu struct.

```
1   struct toaDo{
2       int x;
3       int y;
4       int z;
5   };
6   int main(){
7       toaDo diemA = {3,4,5}, diemB = {0,1,0};
8       diemA = diemB;
9       diemA.x = 4;
10      diemA.x = diemB.z * 3;
11  }
```

TOPIC 01: Kiểu cấu trúc - Struct

Cấu trúc phức tạp.

```
1  struct nhanVien{
2      string chucVu;
3      int luong;
4  };
5  struct congTy{
6      string maChungKhoan;
7      int soLuongNV;
8      nhanVien Thuan;
9  } Apple;
10 int main(){
11     Apple.Thuan.chucVu = "CEO";
12 }
```

TOPIC 01: Kiểu cấu trúc - Struct

Truyền cấu trúc cho hàm

```
1  struct toaDo{
2      int x;
3      int y;
4  };
5  void xuat1(toaDo diem);
6  void xuat2(toaDo &diem);
7  void xuat3(toaDo *diem);
```

Bài tập áp dụng :

Cho đoạn chương trình sau:

```
1  #include <iostream>
2  using namespace std;
3  struct sinhVien{
4      int stt;
5      string hoTen;
6      float diemToan, diemLy, Hoa;
7  } SV;
8  struct UIT{
9      int soLuongSV;
10     int hocPhi = 100000000;
11     sinhVien Thuan;
12 } uit;
```

Câu lệnh nào sau đây nếu viết trong hàm **main** thì chương trình sẽ báo lỗi?

A. cin >> SV.diemToan >> SV.Hoa;

B. cin >> uit.hocPhi >> uit.hocPhi;

C. cin >> SV.stt >> uit.Thuan.hoTen;

D. cin >> uit.soLuongSV >> Thuan.stt;

Bài tập áp dụng :

Cho đoạn chương trình sau:

```
1  #include <iostream>
2  using namespace std;
3  struct toaDo{
4      int x;
5      int y;
6      int z;
7  };
8  int main(){
9      toaDo tdA, tdB = {3,4,5};
10     cin >> tdA.x >> tdA.z >> tdA.y;
11     cout << tdA.x << tdB.y << tdA.z;
12     return 0;
13 }
```

Khi chạy chương trình, nếu nhập vào các số lần lượt là 3, 4, 5 thì sẽ cho ra kết quả là gì?

A. 354

B. 345

C. Chương trình báo lỗi.

D. 344

Bài tập áp dụng :

Cho đoạn chương trình sau:

```
#include <iostream>
using namespace std;
struct toaDo{
    int x;
    int y;
    int z;
};
int main(){
    toaDo *tdp, td;
    tdp = &td;
    tdp->y = 3;
    td.z = 0;
    if(tdp->z = 1)
        td.x = 5;
        td.z = 5;
    if(tdp->z != 1)
        tdp->x = 1;
        td.z = 5;
    cout << tdp->x << td.z;
    return 0;
}
```

Khi chạy chương trình, kết quả in ra màn hình là?

A. Chương trình báo lỗi.

B. 15

C. 51

D. 55

Bài tập áp dụng :

Trong không gian 3D, một đa diện (POLYHEDRON) được mô tả bằng một tập hợp các điểm (POINT). Mỗi điểm có 3 giá trị tương ứng với giá trị trên 3 trục tọa độ là trục x, y và z.

a. Hãy khai báo (định nghĩa) các cấu trúc dữ liệu POINT và POLYHEDRON (Biết các tọa độ đều mang giá trị nguyên),

b. Từ hai cấu trúc dữ liệu trên. Viết hàm (CUBE) để nhập các tọa độ của một hình lập phương.

Bài tập áp dụng :

Trong không g ập hợp các điểm (POINT). ò là trục x, y và z.

a. Hãy khai b t các tọa độ đều mang

b. Từ hai cấu một hình lập phương.

```
#include <iostream>
using namespace std;
struct POINT{
    int x;
    int y;
    int z;
};
struct POLYHEDRON{
    POINT a[100];
    int n;
};
POLYHEDRON CUBE(){
    POLYHEDRON cube;
    for(int i=0;i<8;i++){
        cin >> cube.a[i].x >> cube.a[i].y >> cube.a[i].z;
    }
    return cube;
}
```

NỘI DUNG

TOPIC 01

Kiểu cấu trúc - Struct

TOPIC 02

Con trỏ - Pointer

TOPIC 02: Con trỏ - Pointer

Biến và vùng nhớ:

- Biến là một ô nhớ hoặc 1 vùng nhớ dùng để chứa dữ liệu trong quá trình thực hiện chương trình và có kích thước tùy thuộc vào kiểu dữ liệu của nó.
- Để truy xuất đến giá trị mà biến đang nắm giữ, chương trình cần tìm đến vùng nhớ (địa chỉ) của biến để đọc giá trị bên trong vùng nhớ đó.

TOPIC 02: Con trỏ - Pointer

Một số kiểu dữ liệu cơ bản:

Kiểu dữ liệu	Kích thước	Phạm vi
char	1 byte	-128 đến 127
int	4 bytes	-2147483648 đến 2147483647
short int	2 bytes	-32768 đến 32767
long long int	8 bytes	-9,223,372,036,854,775,808 đến 9,223,372,036,854,775,807
float	4 bytes	1.2e-38 đến 3.4e+38(~ 6 chữ số)
double	8 bytes	2.3e-308 đến 1.8e+308(~ 15 chữ số)

TOPIC 02: Con trỏ - Pointer

Biến và vùng nhớ:

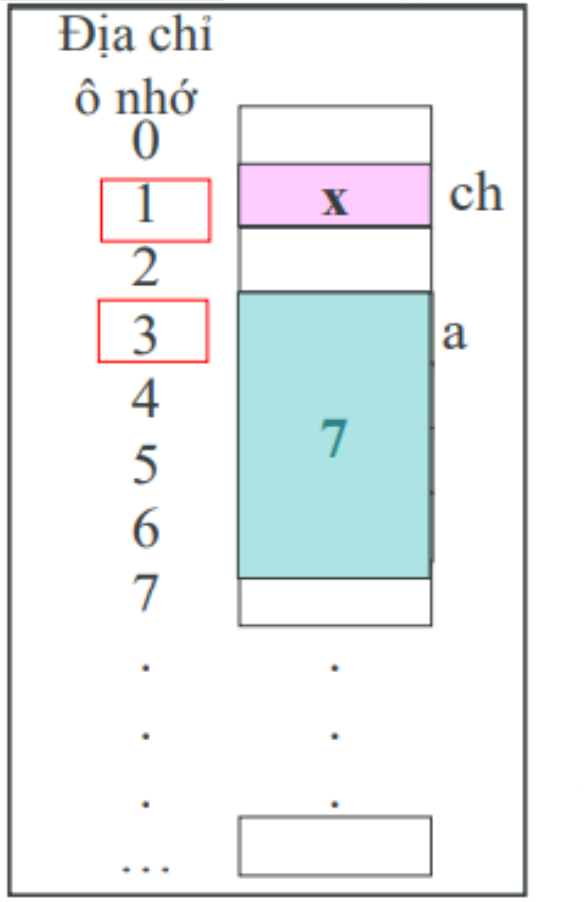
- Bộ nhớ RAM chứa rất nhiều ô nhớ, mỗi ô nhớ có kích thước 1 byte.
- Mỗi ô nhớ có địa chỉ duy nhất và địa chỉ này được đánh số từ 0 trở đi.
- RAM để lưu trữ mã chương trình và dữ liệu trong suốt quá trình thực thi.

TOPIC 02: Con trỏ - Pointer

Biến và vùng nhớ:

- Ví dụ:

```
int main() {
   char ch='x';
   int a = 7;
}
```



Memory Layout (bytes)

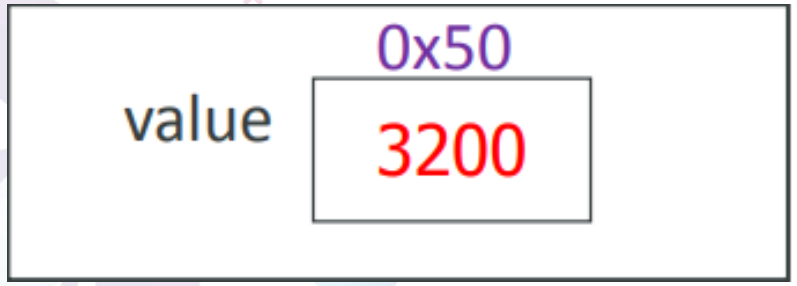
TOPIC 02: Con trỏ - Pointer

Toán tử & và *:

- Toán tử & (Address-of Operator) đặt trước tên biến và cho biết địa chỉ của vùng nhớ của biến.
- Toán tử * (Dereferencing Operator hay Indirection Operator) đặt trước một địa chỉ và cho biết giá trị lưu trữ tại địa chỉ đó
- Ví dụ:

TOPIC 02: Con trỏ - Pointer

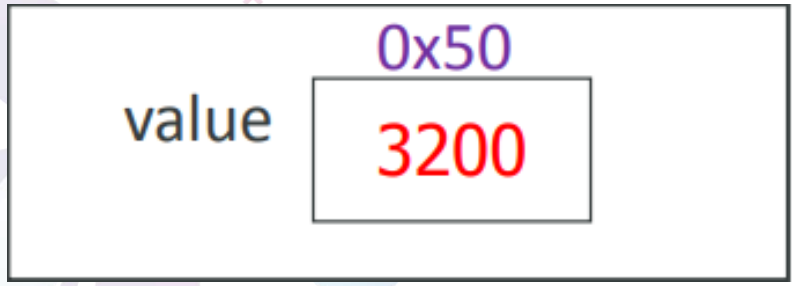
```
int value;
value = 3200;
```




```
cout << " value = " << value;
=> value = 3200;
```

TOPIC 02: Con trỏ - Pointer

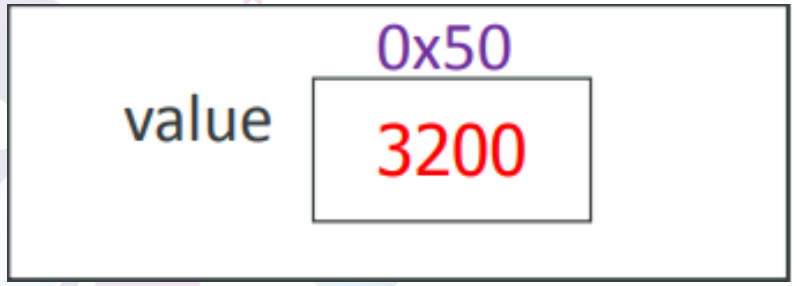
```
int value;
value = 3200;
```




```
cout << " &value = " << &value;
=> &value = 0x50;
```

TOPIC 02: Con trỏ - Pointer

```
int value;
value = 3200;
```




```
cout << " *(&value) = " << *(&value);
=> *(&value) = 3200;
```

TOPIC 02: Con trỏ - Pointer

Khái niệm con trỏ:

- Khái niệm:
  - Con trỏ (Pointer) là một biến lưu trữ địa chỉ của một địa chỉ trên bộ nhớ. Địa chỉ này thường là địa chỉ của một biến khác.
  - VD: Con trỏ x chứa địa chỉ của biến y. Vậy ta nói biến x "trỏ tới" y.
- Phân loại con trỏ:
  - Con trỏ kiểu int dùng để chứa địa chỉ của các biến kiểu int. Tương tự ta có con trỏ kiểu float, double, …

TOPIC 02: Con trỏ - Pointer

Khai báo con trỏ:

<Kiểu dữ liệu> *<Tên biến con trỏ>

- Ví dụ:

```
char *char1;
int *ptrI;
```

TOPIC 02: Con trỏ - Pointer

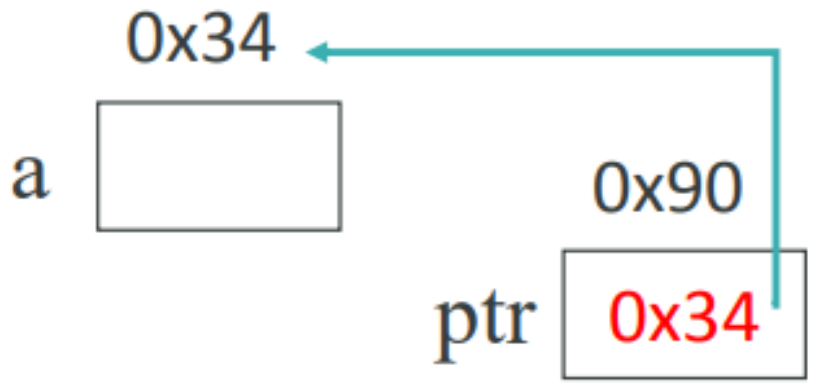
Con trỏ và toán tử &, *:

- Toán tử & dùng trong khởi tạo giá trị cho con trỏ.

<Kiểu dữ liệu> *<Tên biến con trỏ> = &<Tên biến>

- Ví dụ:

```
int a;
int *ptr = &a;
```

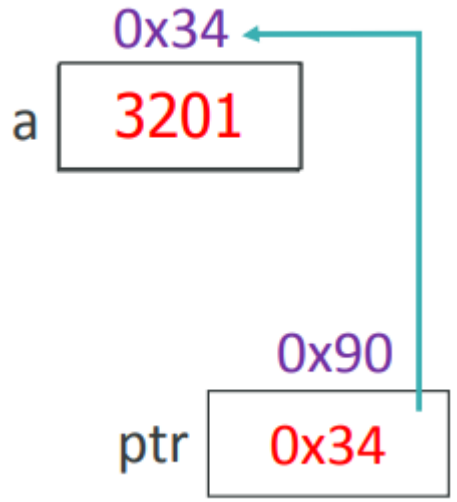


TOPIC 02: Con trỏ - Pointer

Con trỏ và toán tử &, *:

- Toán tử * đặt trước biến con trỏ cho phép truy xuất đến giá trị ô nhớ mà con trỏ trỏ đến.
- Ví dụ:

```
int a = 1000;
int *ptr = &a;
*ptr = 3200;
(*ptr) ++;
```



TOPIC 02: Con trỏ - Pointer

Kích thước của con trỏ trong bộ nhớ:

- Con trỏ chỉ lưu địa chỉ nên kích thước của mọi con trỏ là như nhau khi chạy trên cùng 1 kiến trúc máy tính.

- Để xác định kích thước (bytes) của một kiểu dữ liệu ta dùng toán tử sizeof. Cú pháp: sizeof (type) hoặc sizeof (value)
  - Trong đó type là kiểu dữ liệu, value là tên biến

TOPIC 02: Con trỏ - Pointer

Kích thước của con trỏ trong bộ nhớ:

- Kiểu dữ liệu của con trỏ không mô tả giá trị địa chỉ được lưu trữ bên trong con trỏ, mà dùng để xác định kiểu dữ liệu của biến mà nó trỏ đến trên bộ nhớ.
- Giá trị thực sự của con trỏ là kiểu số nguyên không dấu (unsigned int), trong nền tảng hệ điều hành 32 bits sẽ là unsigned __int32, và trong nền tảng hệ điều hành 64 bits sẽ có kiểu là unsigned __int64

TOPIC 02: Con trỏ - Pointer

Từ khóa const và con trỏ:

- Tùy thuộc vào vị trí đặt từ khóa const dùng trong khái báo biến con trỏ, mà quy định giá trị hằng cho con trỏ hay cho vùng nhớ con trỏ trỏ tới.
- Có 3 trường hợp trong khai báo biến con trỏ và từ khóa const:

```
// non-const pointer to const int
const int * p2 = &x;

// const pointer to non-const int
int * const p3 = &x;

// const pointer to const int
const int * const p4 = &x;
```

TOPIC 02: Con trỏ - Pointer

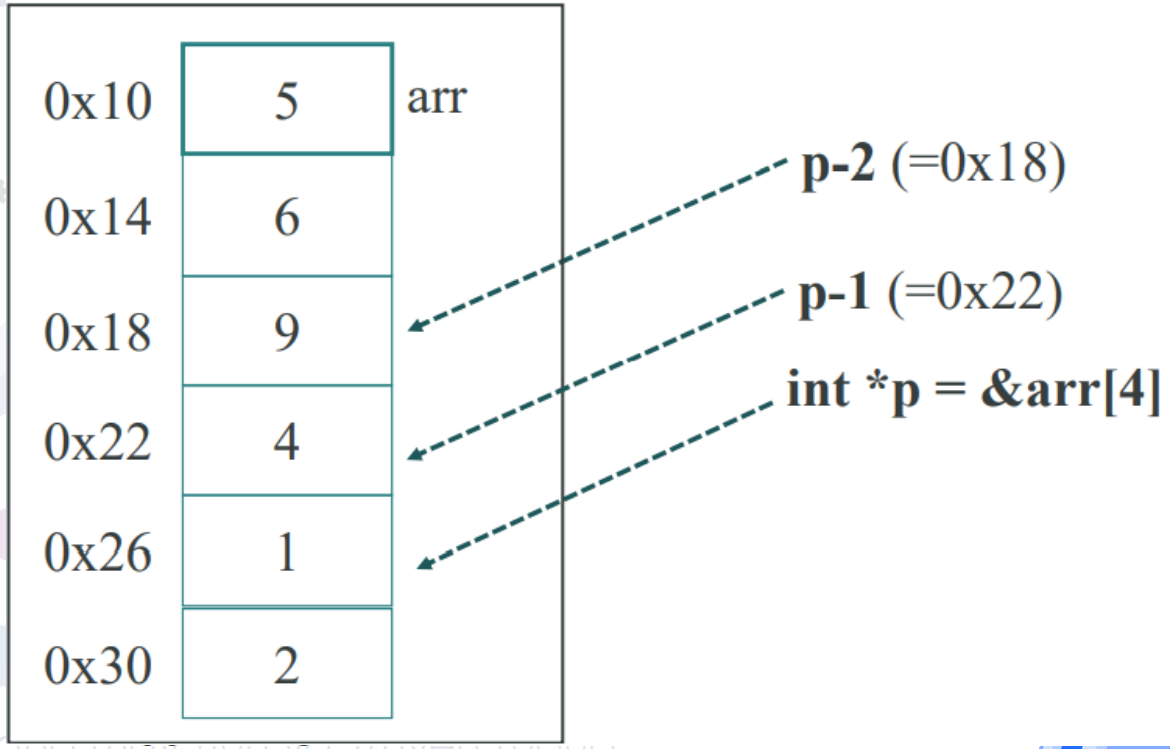
Một số lưu ý:

- Nắm rõ quy tắc sau, ví dụ int a, *pa = &a;
  - *pa và a đều chỉ nội dung của biến a.
  - pa và &a đều chỉ địa chỉ của biến a.
- Không nên sử dụng con trỏ khi chưa được khởi tạo. Kết quả sẽ không lường trước được.
  - int *pa; *pa = 1904; => không nên dùng
  - int *pa = NULL; => nên dùng

TOPIC 02: Con trỏ - Pointer

Các phép toán số học trên con trỏ - Pointer Arithmetics:

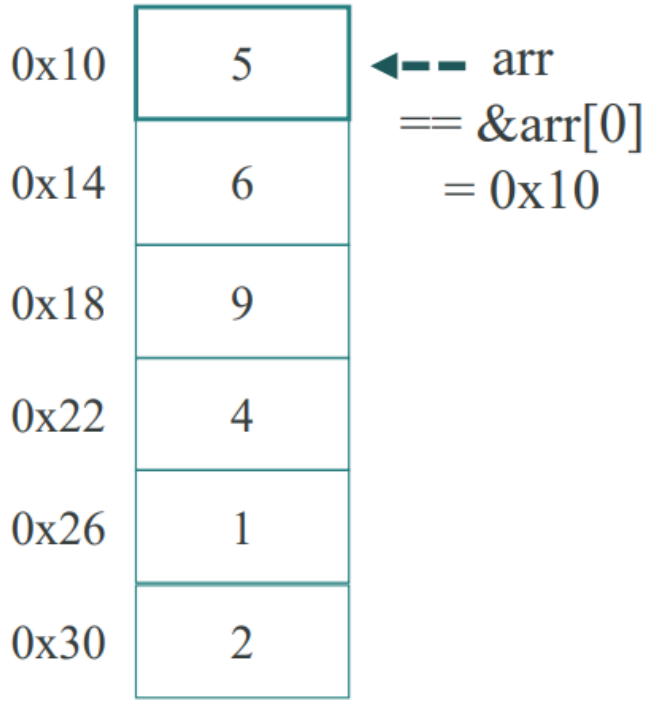
- Phép cộng (tăng)
  - + n <=> + n * sizeof (<kiểu dữ liệu>)
  - Có thể sử dụng toán tử gộp += hoặc ++
- Phép trừ (giảm)
  - – n <=> – n * sizeof (<kiểu dữ liệu>)
  - Có thể sử dụng toán tử gộp –= hoặc – –



TOPIC 02: Con trỏ - Pointer

Mảng 1 chiều và con trỏ:

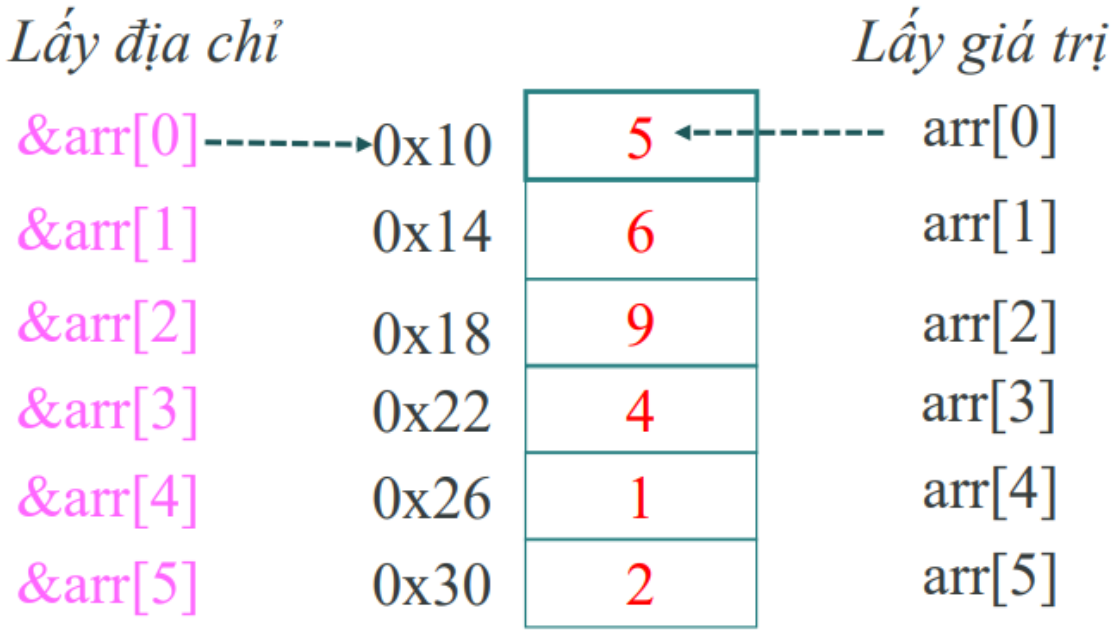
- Cho mảng 1 chiều: int arr[6] = {5, 6, 9, 4, 1, 2};
  - ➢ Tên mảng arr là một hằng con trỏ => không thể thay đổi giá trị của hằng này.
  - ➢ arr là địa chỉ đầu tiên của mảng => arr == &arr[0]



TOPIC 02: Con trỏ - Pointer

Mảng 1 chiều và con trỏ:

- Cho mảng 1 chiều: int arr[6] = {5, 6, 9, 4, 1, 2};



TOPIC 02: Con trỏ - Pointer

Mảng 1 chiều và con trỏ:

- Truy xuất tới giá trị của phần tử thứ i của mảng (xét i là chỉ số hợp lệ của mảng):
  - arr[i] == *(arr+i) == parr[i] == *(parr + i)
- Truy xuất tới địa chỉ của phần tử thứ i của mảng (xét i là chỉ số hợp lệ của mảng):
  - &arr[i] == arr+i == &parr[i] == parr + i

TOPIC 02: Con trỏ - Pointer

Cấp phát tĩnh (static memory allocation):

- **Static memory allocation** được áp dụng cho biến tĩnh và biến toàn cục.
  - Vùng nhớ của các biến này được cấp phát ngay khi biên dịch.
  - Kích thước của vùng nhớ được cấp phát phải được cung cấp tại thời điểm biên dịch chương trình.
  - Đối với việc khai báo mảng một chiều, đây là lý do tại sao số lượng phần tử là hằng số.
- **Automatic memory allocation** sử dụng để cấp phát vùng nhớ cho các biến cục bộ, tham số của hàm.
  - Bộ nhớ được cấp phát tại thời điểm chương trình đang chạy, khi chương trình đi vào một khối lệnh.
  - Các vùng nhớ được cấp phát sẽ được thu hồi khi chương trình đi ra khỏi một khối lệnh.
  - Kích thước vùng cần cấp phát cũng phải được cung cấp rõ ràng.

TOPIC 02: Con trỏ - Pointer

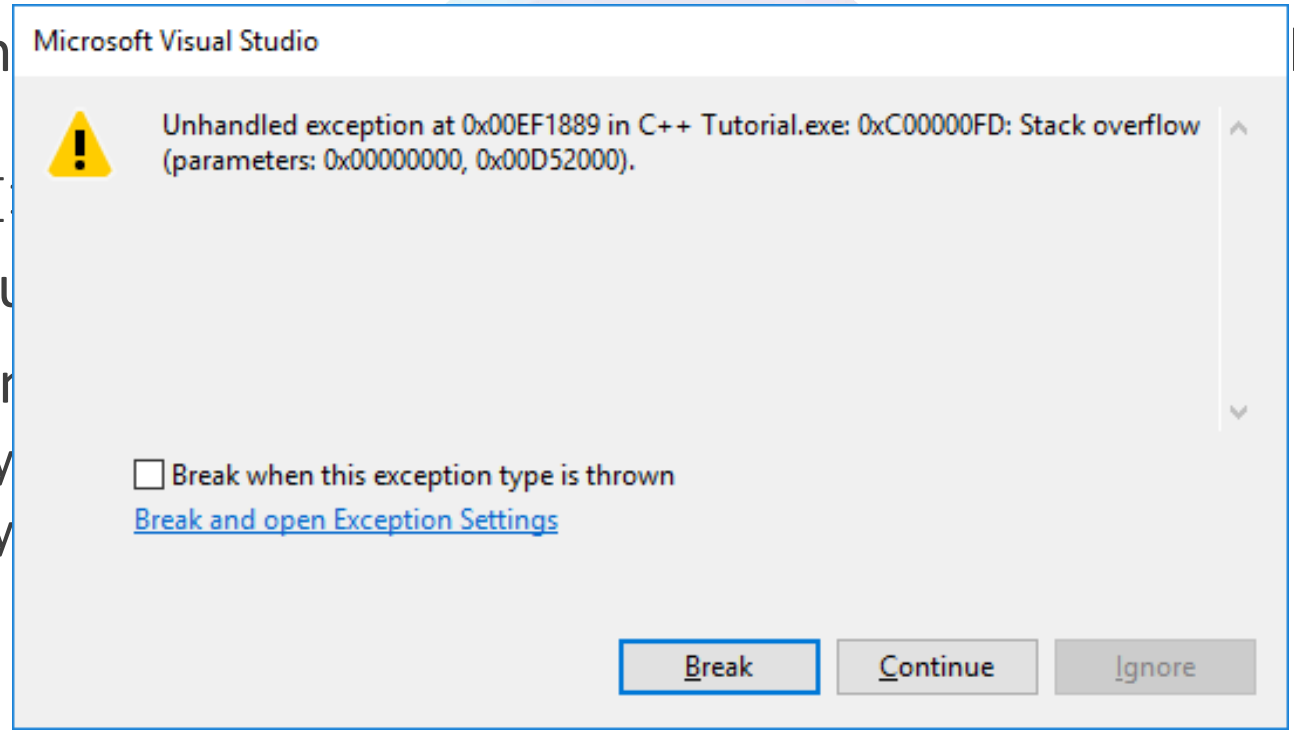
Nhược điểm của cấp phát tĩnh :

- Kích thước vùng nhớ cấp phát phải được cung cấp tại thời điểm biên dịch chương trình.
  - VD: int array[100], char str[] = "abc";
- Cấp phát và thu hồi vùng nhớ do chương trình quyết định.
- Kích thước bộ nhớ dùng cho cấp phát tĩnh bị giới hạn.
  - char ch_array[1024 * 1000]; => chạy bình thường
  - char ch_array[1024 * 1024]; => lỗi "stack overflow"

TOPIC 02: Con trỏ - Pointer

Nhược điểm của cấp phát tĩnh :

- Kích thước vùn biên dịch chương trình.
  - VD: int array[
- Cấp phát và thu
- Kích thước bộ r
  - char ch_array
  - char ch_array



TOPIC 02: Con trỏ - Pointer

Cấp phát động (dynamic memory allocation):

- **Dynamic memory allocation** là một giải pháp cấp phát bộ nhớ cho chương trình tại thời điểm chương trình đang chạy **(run-time)**.
- **Dynamic memory allocation** sử dụng phân vùng **Heap** trên bộ nhớ để cấp phát cho chương trình.
- Cấp phát bộ nhớ
  - Trong C: Hàm malloc, calloc, realloc (<stdlib.h> hoặc <alloc.h>
  - Trong C++: Toán tử new
- Giải phóng bộ nhớ
  - Trong C: Hàm free
  - Trong C++: Toán tử delete

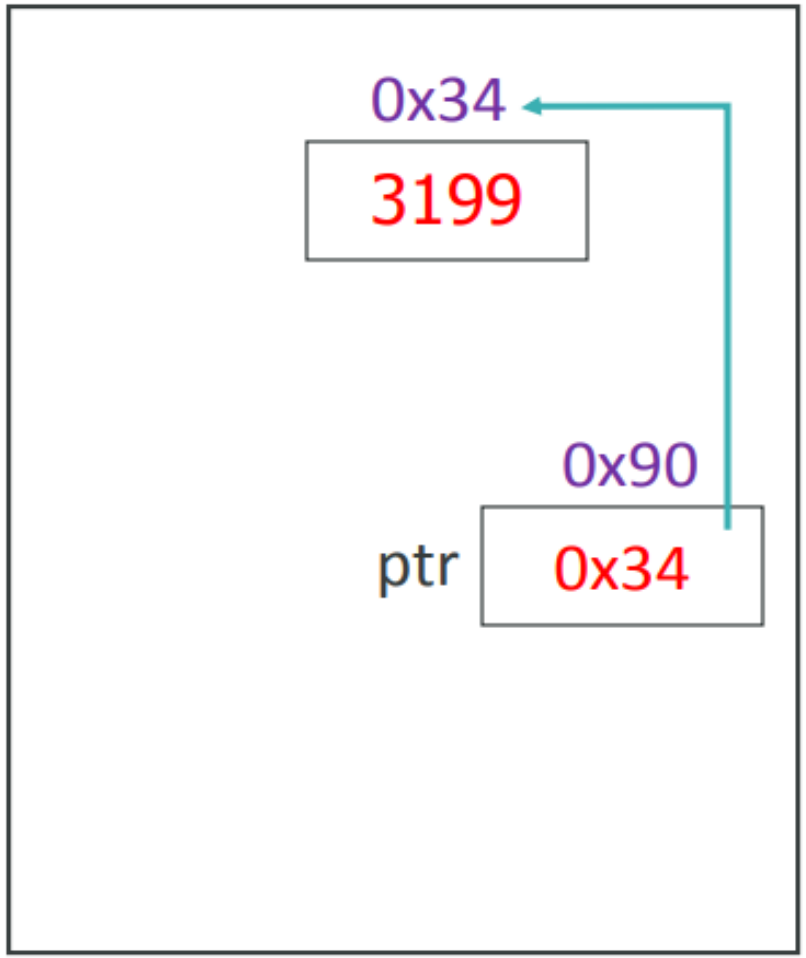
TOPIC 02: Con trỏ - Pointer

Cấp phát động (dynamic memory allocation):

- Có thể cấp phát động cho biến con trỏ bằng toán tử new. Toán tử new sẽ tạo ra biến "không tên" cho con trỏ trỏ tới.
- Cú pháp: <type> *<name> = new <type> ;
- Ví dụ: int *ptr = new int;
  - Tạo ra một biến "không tên" và gán ptr trỏ tới nó
  - Có thể làm việc với biến "không tên" thông qua con trỏ *ptr

TOPIC 02: Con trỏ - Pointer

```
#include <iostream>

using namespace std;


int main() {

        int *p = new int;

        if (p == NULL) {

                cout << "Error: Khong
                du bo nho.\n";

                exit(1);

        }

        *p = 3199;

}
```



TOPIC 02: Con trỏ - Pointer

Khởi tạo giá trị trong cấp phát động:

- Cú pháp: <type> *<name> = new <type> (value) ;
- Ví dụ: int *ptr = new int (100);

```
#include <iostream>
using namespace std;

int main() {
	int *p;
	p = new int(99); // initialize with 99
	cout << *p; // displays 99
	return 0;
}
```

TOPIC 02: Con trỏ - Pointer

Toán tử delete:

- Toán tử delete dùng để giải phóng vùng nhớ trong HEAP do con trỏ trỏ tới (con trỏ được cấp phát bằng toán tử new). Cú pháp:

delete <pointername>;

- Ghi chú: Sau khi gọi toán tử delete thì con trỏ vẫn trỏ tới vùng nhớ trước khi gọi hàm delete. Ta gọi là "con trỏ lạc". Ta vẫn có thể gọi tham chiếu trên con trỏ, tuy nhiên thường là nguy hiểm

=> Hãy tránh con trỏ lạc bằng cách gán con trỏ bằng NULL sau khi delete. VD:

```
delete ptr;
ptr = NULL;
```

TOPIC 02: Con trỏ - Pointer

Khởi tạo giá trị trong cấp phát động:

- Con trỏ có thể là tham số của hàm
- Con trỏ có thể là kiểu trả về của hàm
- Ví dụ: int* findOtherPointer(int* p);
  - Hàm này khai báo:
    - ✓ Có tham số kiểu con trỏ trỏ tới int
    - ✓ Trả về biến con trỏ trỏ tới int

TOPIC 02: Con trỏ - Pointer

Mảng động:

- Kích thước không xác định ở thời điểm lập trình mà xác định khi chạy chương trình

=> Loại bỏ hạn chế của mảng chuẩn:

- ✓ Không thay đổi được kích thước
- ✓ Bộ nhớ bị giới hạn

- Để xin cấp phát và giải phóng vùng nhớ cho mảng một chiều trên **Heap**, chúng ta cũng sử dụng toán tử **new và delete** để xử lý.
- Thao tác tương tự như mảng thường.

TOPIC 02: Con trỏ - Pointer

Cấp phát mảng động:

- Cú pháp:

<type> *<pointerName> = new <type> [số lượng phần tử] ;

- Ví dụ:

```
double *d = NULL;
d = new double[10];
```

=> Tạo biến mảng cấp phát động d có 10 phần tử, kiểu dữ liệu là double.

TOPIC 02: Con trỏ - Pointer

Cấp phát mảng động:

- Dùng toán tử delete kết hợp dấu [] để xóa mảng động.
- Ví dụ:

```
double *d = new double[10];
//... Processing
delete[] d;
```

- ➢ Giải phóng tất cả vùng nhớ của mảng động này
- ➢ Cặp ngoặc vuông báo hiệu có mảng
- ➢ Nhắc lại: d vẫn trỏ tới vùng nhớ đó. Vì vậy sau khi delete, cần gán d = NULL;

TOPIC 02: Con trỏ - Pointer

Mảng động 2 chiều:

- Là 1 mảng được cấp phát có các phần tử là con trỏ.
- Mỗi con trỏ sẽ được cấp phát 1 mảng 1 chiều:

```
int **m = new int* [3];
```

➢ Tạo ra mảng 3 con trỏ int

➢ Sau đó biến mỗi con trỏ này thành mảng 4 biến int

```
for (int i = 0; i < 3; i++)
    m[i] = new int[4];
```

➢ Kết quả là mảng động 3 x 4

```
1. int main()
2. {
3.     // Cấp phát vùng nhớ
4.     int row = 2, col = 3;
5.     int **p = new int *[row];
6.     for (int i = 0; i < row; i++)
7.         p[i] = new int[col];
8.     //Giải phóng vùng nhớ
9.     for (int i = 0; i < row; i++)
           delete[] p[i];
1.     delete[] p;
2.     return 0;
3. }
```

TOPIC 02: Con trỏ - Pointer

Con trỏ và hàm:

- Tham số của hàm là 1 biến con trỏ
  - ➢ Trường hợp này thay đổi giá trị của đối số:

```
1  void Hoanvi(int *x, int *y)
2  {
3    int z = *x;
4    *x = *y;
5    *y = z;
6  }
7  int main()
8  {
9    int a = 1, b = 2;
10   cout << a << b;
11   Hoanvi(&a, &b);
12   cout << a << b;
13 }
```

TOPIC 02: Con trỏ - Pointer

Con trỏ và hàm:

- Tham số của hàm là 1 biến con trỏ

  - Trường hợp này không thay đổi giá trị của đối số:

```
1  void Capphat(int *a)
2  {
3    a = new int[5];
4    for (int i = 0; i < 5; i++)
5      a[i] = i + 1;
6    delete[] a;
7    a = NULL;
8  }
9  int main()
10 {
11   int n = 5;
12   int *b = &n;
13   Capphat(b);
14   cout << b << '\n';
15 }
```

TOPIC 02: Con trỏ - Pointer

Con trỏ và hàm:

- Kiểu trả về của hàm là 1 con trỏ

```
1  int *Capphat(int n)
2  {
3     int *a = new int[n];
4     for (int i = 0; i < n; i++)
5        a[i] = i + 1;
6     return a;
7  }
```

Bài tập áp dụng :

Cho đoạn chương trình sau:

```
1  #include <iostream>
2  using namespace std;
3  int main(){
4      int *ptr1,*ptr2;
5      int a = 5;
6      ptr1 = &a; a = 10;
7      cout << ptr1;
8      ptr2 = ptr1; *ptr1 = 20;
9      cout << *ptr2;
10     cout << ptr2;
11     ptr2 = new int [5]; *ptr2 = 5;
12     cout << ptr2;
13     delete[] ptr2;
14     return 0;
15 }
```

Địa chỉ của ptr1 là 0xfa20. Địa chỉ của a là 0xff15. Địa chỉ của ptr2 là 0xfaab. Địa chỉ của mảng cấp phát mới là 0xd888.

Tại dòng 7, màn hình in ra là:

A. 5

B. 10

C. 0xff15

D. 0xfa20

Bài tập áp dụng :

Cho đoạn chương trình sau:

```
1  #include <iostream>
2  using namespace std;
3  int main(){
4      int *ptr1,*ptr2;
5      int a = 5;
6      ptr1 = &a; a = 10;
7      cout << ptr1;
8      ptr2 = ptr1; *ptr1 = 20;
9      cout << *ptr2;
10     cout << ptr2;
11     ptr2 = new int [5]; *ptr2 = 5;
12     cout << ptr2;
13     delete[] ptr2;
14     return 0;
15 }
```

Địa chỉ của ptr1 là 0xfa20. Địa chỉ của a là 0xff15. Địa chỉ của ptr2 là 0xfaab. Địa chỉ của mảng cấp phát mới là 0xd888.

Tại dòng 9, màn hình in ra là:

A. 5

B. 10

C. 90

D. Giá trị khác (20)

Bài tập áp dụng :

Cho đoạn chương trình sau:

```
1  #include <iostream>
2  using namespace std;
3  int main(){
4      int *ptr1,*ptr2;
5      int a = 5;
6      ptr1 = &a; a = 10;
7      cout << ptr1;
8      ptr2 = ptr1; *ptr1 = 20;
9      cout << *ptr2;
10     cout << ptr2;
11     ptr2 = new int [5]; *ptr2 = 5;
12     cout << ptr2;
13     delete[] ptr2;
14     return 0;
15 }
```

Địa chỉ của ptr1 là 0xfa20. Địa chỉ của a là 0xff15. Địa chỉ của ptr2 là 0xfaab. Địa chỉ của mảng cấp phát mới là 0xd888.

Tại dòng 10, màn hình in ra là:

A. 0xfa20

B. 0xff15

C. 0xfaab

D. 0xd888

Bài tập áp dụng :

Cho đoạn chương trình sau:

```
1  #include <iostream>
2  using namespace std;
3  int main(){
4      int *ptr1,*ptr2;
5      int a = 5;
6      ptr1 = &a; a = 10;
7      cout << ptr1;
8      ptr2 = ptr1; *ptr1 = 20;
9      cout << *ptr2;
10     cout << ptr2;
11     ptr2 = new int [5]; *ptr2 = 5;
12     cout << ptr2;
13     delete[] ptr2;
14     return 0;
15 }
```

Địa chỉ của ptr1 là 0xfa20. Địa chỉ của a là 0xff15. Địa chỉ của ptr2 là 0xfaab. Địa chỉ của mảng cấp phát mới là 0xd888.

Tại dòng 12, màn hình in ra là:

A. 0xfa20

B. 0xff15

C. 0xfaab

D. 0xd888

Bài tập áp dụng :

Cho đoạn chương trình sau:

```
1  #include <iostream>
2  using namespace std;
3  int main(){
4      int a[] = {1,2,3,4,5,6};
5      for(int i=0;i<6;i++){
6          cout << *a++;
7      }
8  }
```

Khi chạy chương trình có bị lỗi hay không? Giải thích.

Dòng 6. Lỗi biên dịch vì:

- Theo thứ tự ưu tiên toán tử thì a++ trước => *(a++)
- a == &a[0] là 1 con trỏ hằng

Bài tập áp dụng :

Cho đoạn chương trình sau:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int a[] = {3, 7, 8, 9};
6      int *p = a;
7      cout << (*p)++ << *(p++) << *p++ << *p;
8      return 0;
9  }
```

Khi chạy, chương trình cho ra kết quả gì?
Giải thích.

3478

cout << *p

(*p) += 1;

cout << *p;

p += 1;

cout << *p;

p += 1;

cout << *p;

Bài tập áp dụng :

Cho đoạn chương trình sau:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int a[] = {20, 30, 40};
6      int *p = a;
7      int q;
8      q = ++*p;
9      q = *p++;
10     q = *++p;
11     return 0;
12 }
```

Khi chạy chương trình, kết quả của q lần lượt ở các dòng 8,9 và 10 là gì? Giải thích.

Dòng 8: 21

q = (*p) + 1;

Dòng 9: 21

q = *p;

p += 1;

Dòng 10: 40

p += 1;

q = *p;

Bài tập áp dụng :

Cho đoạn chương trình sau:

```
1  #include <iostream>
2  using namespace std;
3  int main(){
4      int a[6] = {36,32,20,19,17,15}, x = 18;
5      int left = 0, right = 5, mid;
6      while(left < right){
7          mid = (left + right)/2;
8          if(a[mid] > x){
9              left = mid + 1;
10         }
11         else right = mid;
12     }
13 }
```

Cho biết giá trị của biến a[left] sau khi chạy chương trình trên.

A. 15

B. 17

C. 19

D. Một kết quả khác

19:30
11/02

NMLT → NMĐT → Thi CK

19:30
10/02

14/12