NHẬP MÔN LẬP TRÌNH

BÀI 12: CON TRỎ CƠ BẢN

CĐR buổi học

- Sau khi học xong buổi học, sinh viên có khả năng:
  - Hiểu được khái niệm con trỏ, địa chỉ của biến và quản lý các biến trong C++.
  - Sử dụng con trỏ trong lập trình
  - Biết được một số thuật ngữ và tiếng Anh tương ứng

Bảng các thuật ngữ Việt- Anh liên quan nội dung con trỏ

Thuật ngữ tiếng Việt	Thuật ngữ tiếng Anh
Con trỏ	Pointer
Hằng con trỏ	Constant pointer
Địa chỉ bộ nhớ	Memory Address
Toán tử &	Address-of Operator
Toán tử *	Dereferencing Operator, or: Indirection Operator
Cấp phát bộ nhớ	Memory Allocation
Giải phóng bộ nhớ	De-Allocate Memory
Cấp phát tĩnh	Static Memory Allocation
Cấp phát động	Dynamic Memory Allocation
Biến động	Dynamic Variable
Phép toán số học trên con trỏ	Pointer Arithmetic

Nội dung

1. Khái niệm và cách sử dụng con trỏ
2. Con trỏ và mảng 1 chiều
3. Bài tập

1. Khái niệm và cách sử dụng

1.1 Biến và vùng nhớ

1.2 Khái niệm con trỏ

1.3 Khai báo con trỏ

1.4 Con trỏ và toán tử &, *

1.5 Con trỏ NULL

1.6 Kích cỡ con trỏ

1.7 Từ khóa const và con trỏ

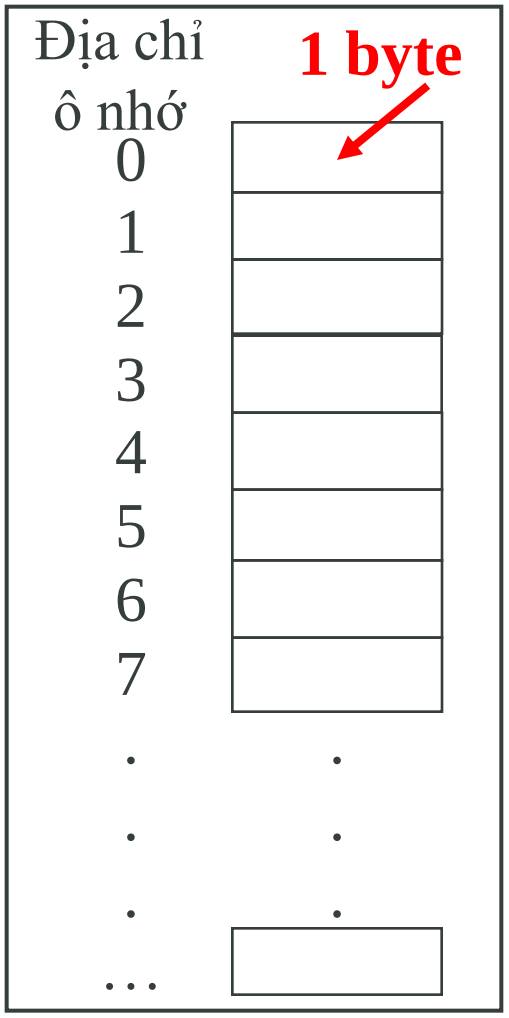
1.8 Con trỏ và hàm

Bài tập

Một số lưu ý

1.1 Biến và vùng nhớ

- Bộ nhớ máy tính
  - 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.



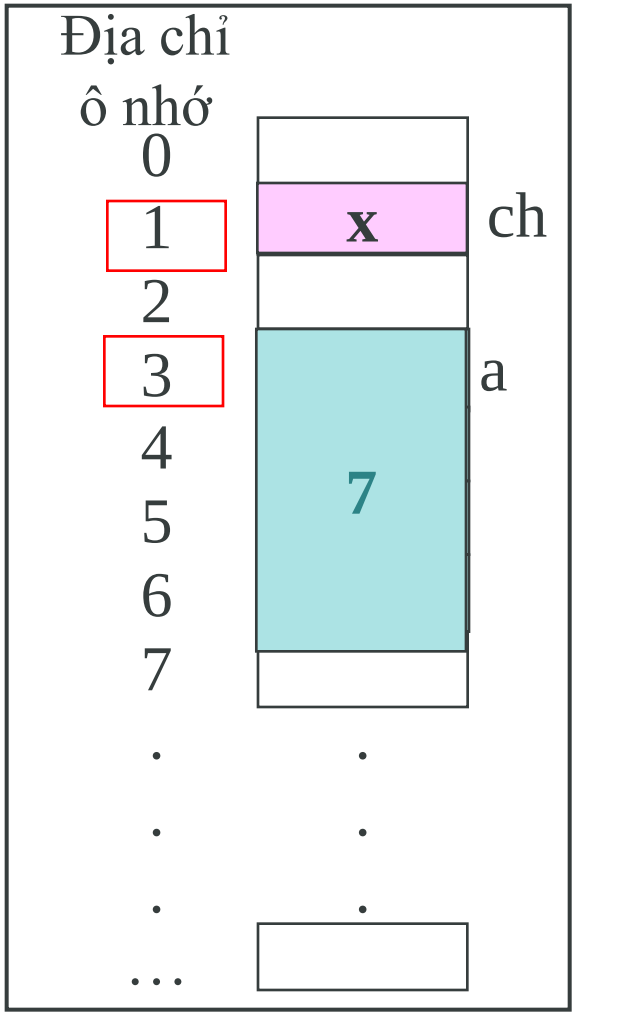
Memory Layout (bytes)

1.1 Biến và vùng nhớ

- Khi khai báo biến, máy tính sẽ **dành riêng một vùng nhớ** để lưu biến đó.
- Khi tên biến được gọi, máy tính sẽ thực hiện 2 bước sau:
  - **Tìm kiếm địa chỉ ô nhớ** của biến.
  - **Truy xuất hoặc thiết lập giá trị** của biến được lưu trữ tại ô nhớ đó.
- Ví dụ:

1.1 Biến và vùng nhớ

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



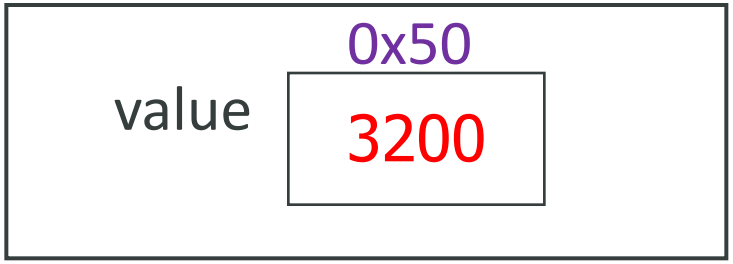
Memory Layout (bytes)

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:

Toán tử & và *

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



Memory Layout

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

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

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

1.2 Khái niệm con trỏ

- Khái niệm:

  Con trỏ (**Pointer**) là một **biến lưu trữ địa chỉ** của một **địa chỉ bộ nhớ**. Địa chỉ này thường là địa chỉ của một biến khác.

  VD: Biến 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, …

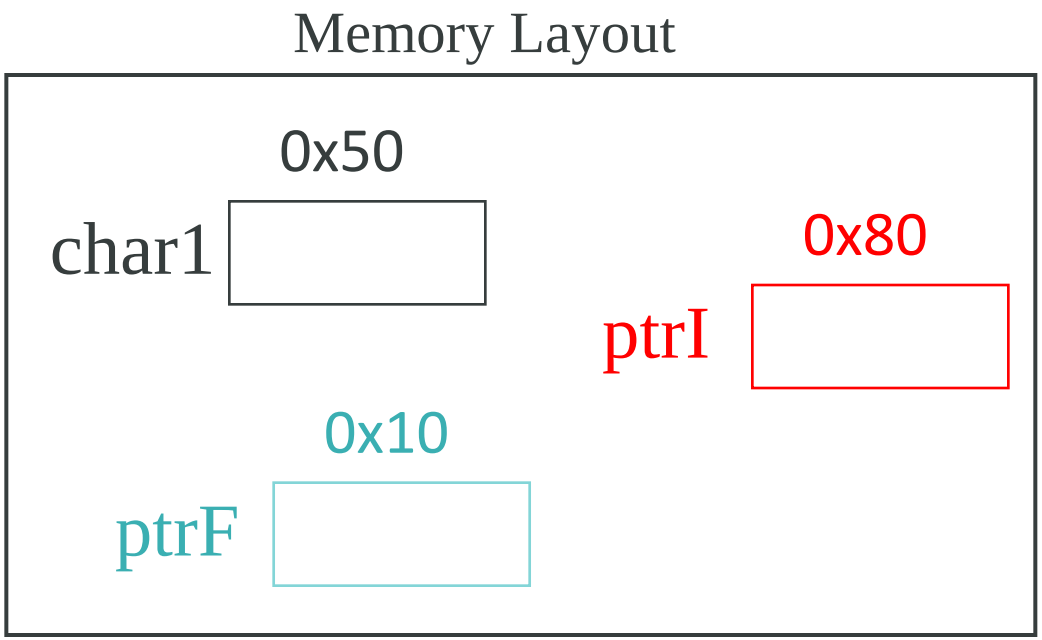
1.3 Khai báo con trỏ

- Khai báo
  - Giống như mọi biến khác, biến con trỏ muốn sử dụng cũng cần phải được khai báo.

```
<kiểu dữ liệu> *<tên biến con trỏ>;
```

- Ví dụ

```
char char1;
int *ptrI;
float *ptrF;
```



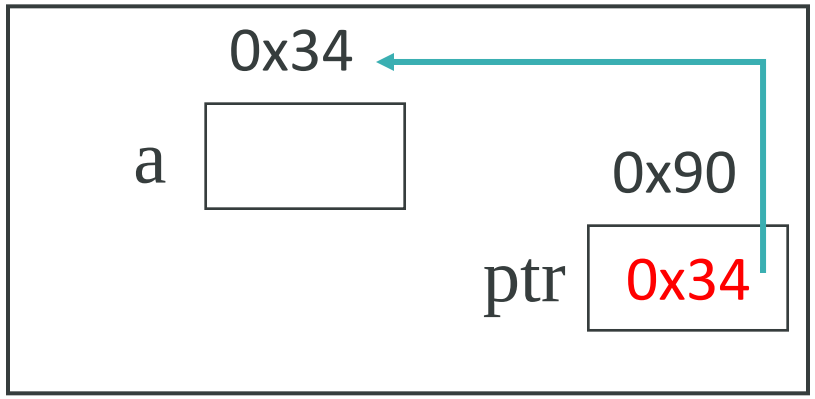
1.4 Con trỏ và toán tử &, *

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

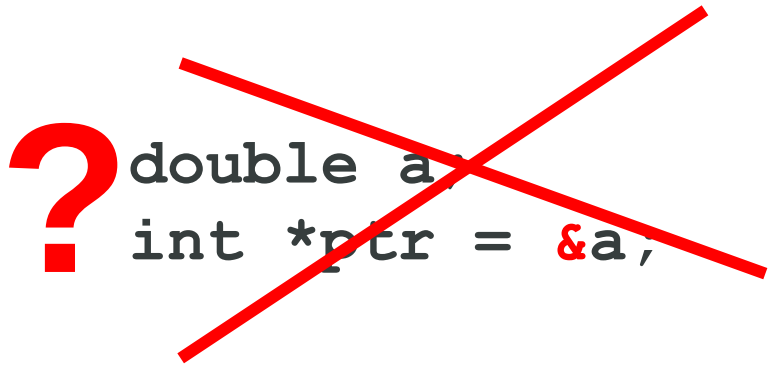
```
<kiểu dữ liệu> *<tên biến con trỏ> = & <tên biến>;
```

- Ví dụ:

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



Memory Layout

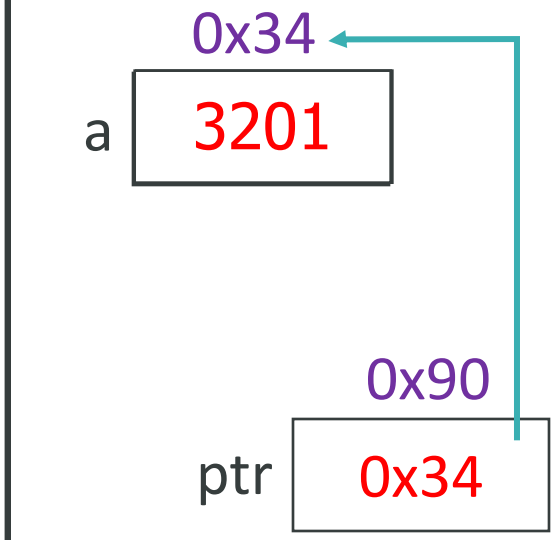


1.4 Con trỏ và toán tử &, *

- 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;
cout << ptr << " " << *ptr;
// a = 3200
*ptr = 3200;
cout << *ptr;
(*ptr) ++;
```

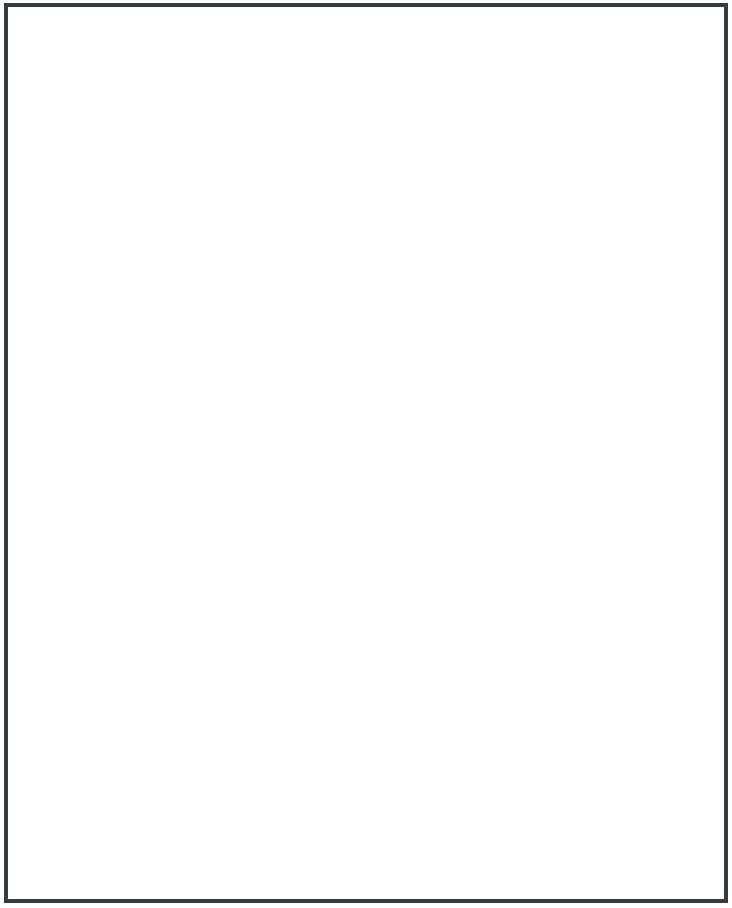


Memory Layout

Ví dụ

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


int main() {
    int a;
    int *ptr;
    int value;
    a = 3200;
    ptr = &a;
    value = --(*ptr);
}
```

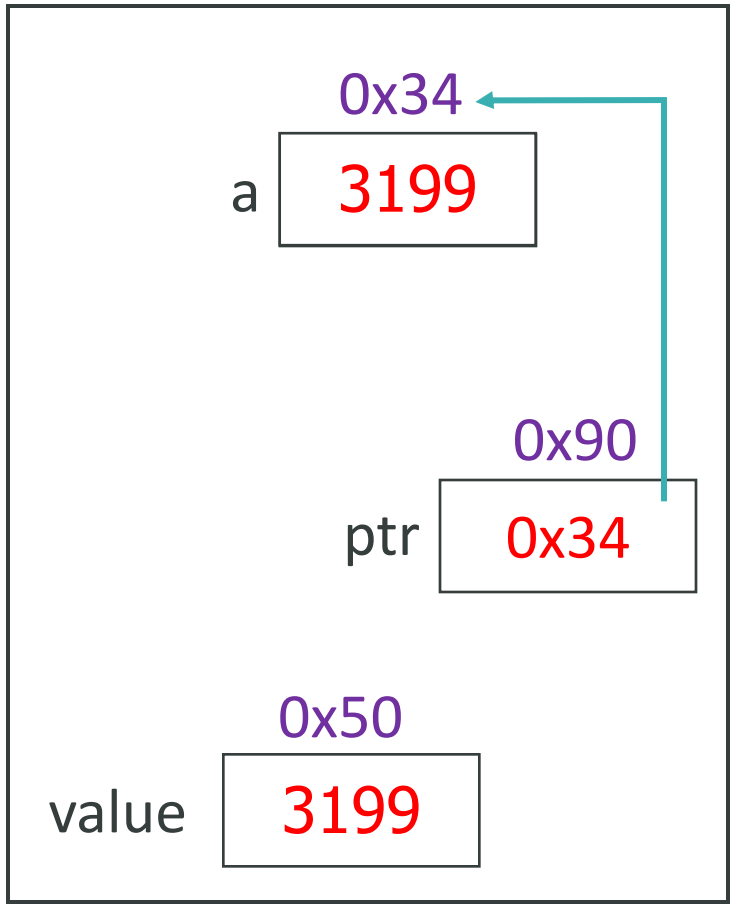


Memory Layout

Ví dụ

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


int main() {
    int a;
    int *ptr;
    int value;
    a = 3200;
    ptr = &a;
    value = --(*ptr);
}
```



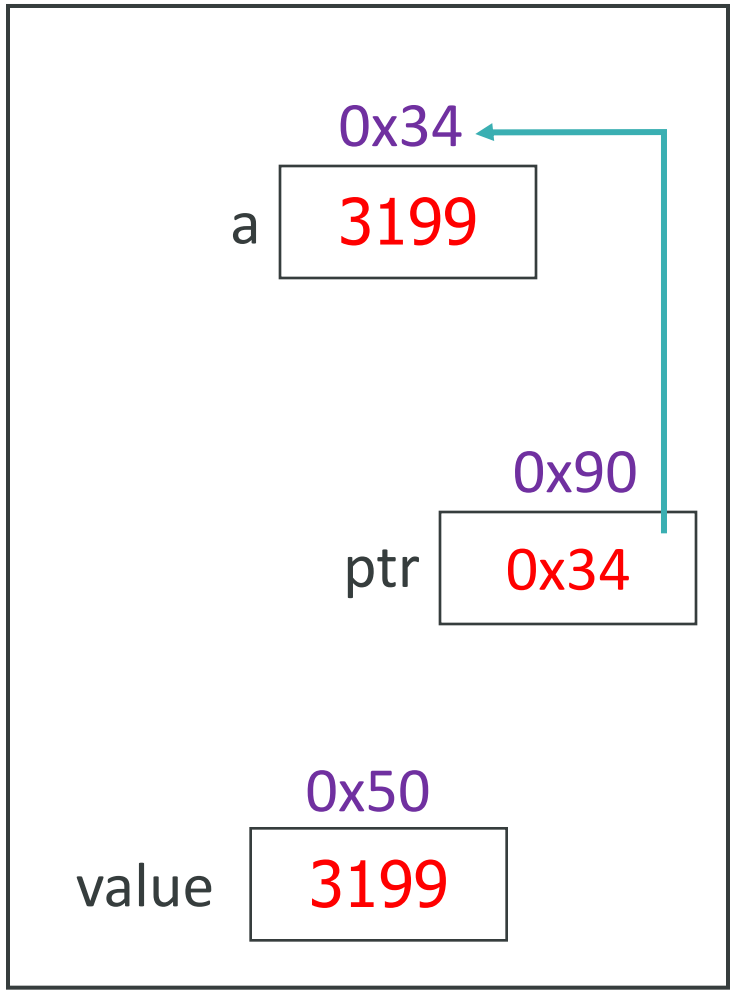
Memory Layout

Ví dụ

```
value =   3199
  ptr =   0x34
    a =   3199

&value =  0x50
 &ptr =   0x90
   &a =   0x34

 *ptr =   3199
&(*ptr) = 0x34
*(*ptr) = error
*(&(*ptr)) = 3199
```



Memory Layout

Phép gán con trỏ

- Có thể gán biến con trỏ:

```
int *p1, *p2;
p2 = p1;
```

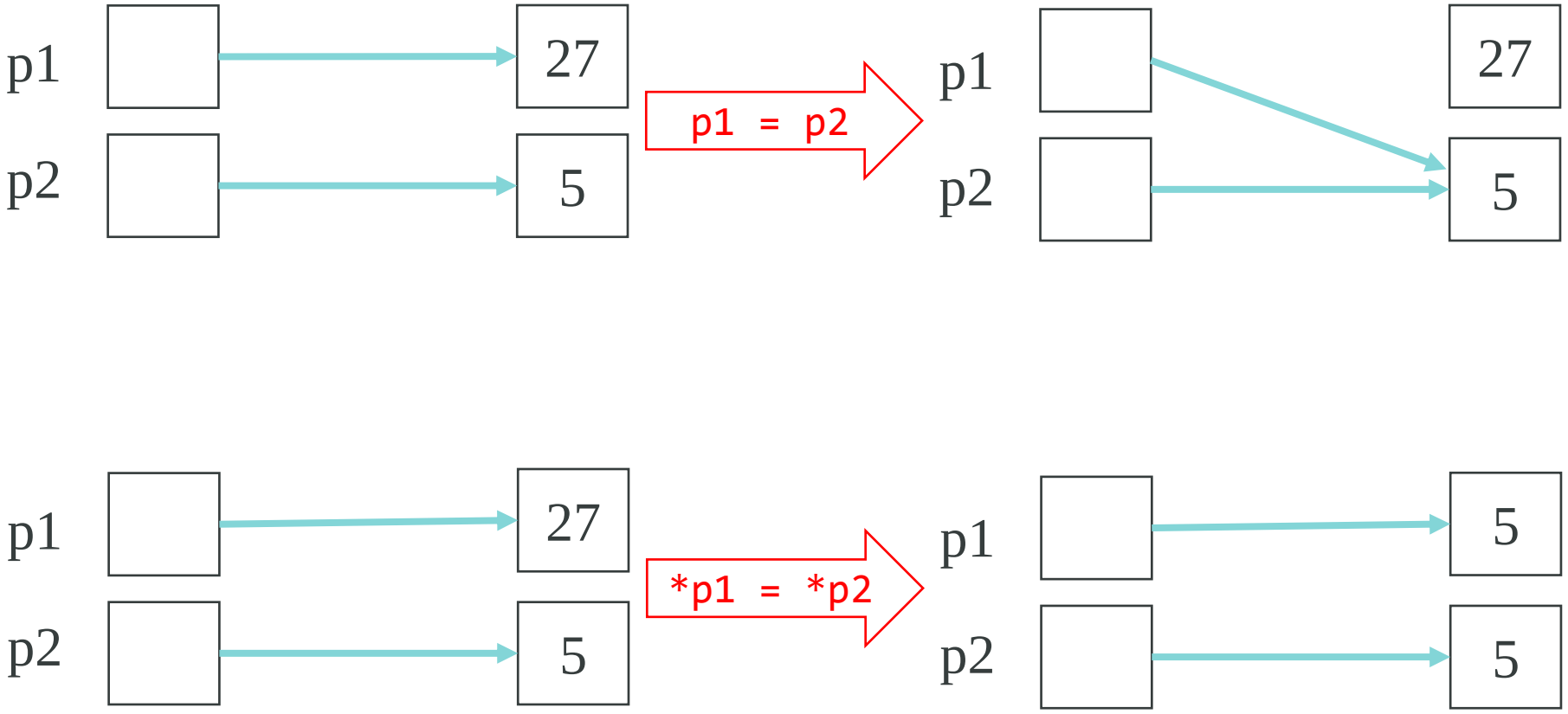
⇒ Gán một con trỏ cho con trỏ khác

⇒ "Chỉ định p2 trỏ tới nơi mà p1 đang trỏ tới"

Dễ bị lẫn với: `*p2 = *p1;`

⇒ Gán "giá trị trỏ bởi p1" cho "giá trị trỏ bởi p2"

Ví dụ



1.5 Con trỏ NULL

- Khái niệm
  - Con trỏ NULL là con trỏ không trỏ vào đâu cả.
  - Khác với con trỏ chưa được khởi tạo.

```
int n;

int *p1 = &n;

int *p2;  // unreferenced local variable

int *p3 = NULL;
```

1.6 Toán tử sizeof

- 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

- Kích thước của kiểu dữ liệu không giống nhau **cho tất cả máy tính**. Nên dùng toán tử sizeof để biết chính xác kích thước của dữ liệu.

- Con trỏ chỉ lưu địa chỉ nên kích thước của mọi con trỏ là như nhau. (Kết quả sau mang tính chất tham khảo)

```
int a;          sizeof a  = 4    sizeof(int)     = 4
double b;       sizeof(b) = 8    sizeof(double)  = 8
char c;         sizeof(c) = 1    sizeof(char)    = 1
int *pa;        sizeof pa = 4    sizeof(int*)    = 4
double *pb;     sizeof pb = 4    sizeof(double*)= 4
char *pc;       sizeof(pc)= 4    sizeof(char*)   = 4
```

1.7 Từ khóa const và con trỏ

- Hằng số dùng trong khai báo một biến cho biết giá trị của biến không được phép thay đổi trong quá trình thực hiện chương trình.
- 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.

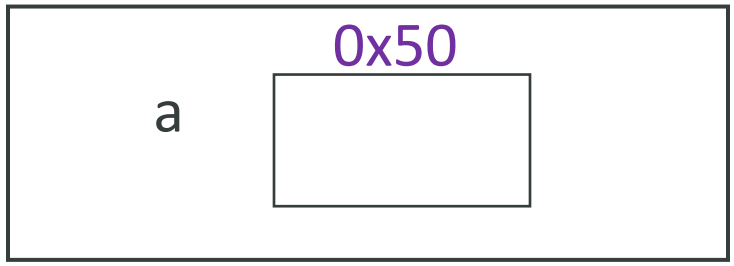
1.7 Con trỏ và hàm

- Xét ví dụ sau:

Hãy viết hàm để nhập giá trị cho 1 biến.

Cách viết nội dung hoàn toàn ở hàm main như sau:

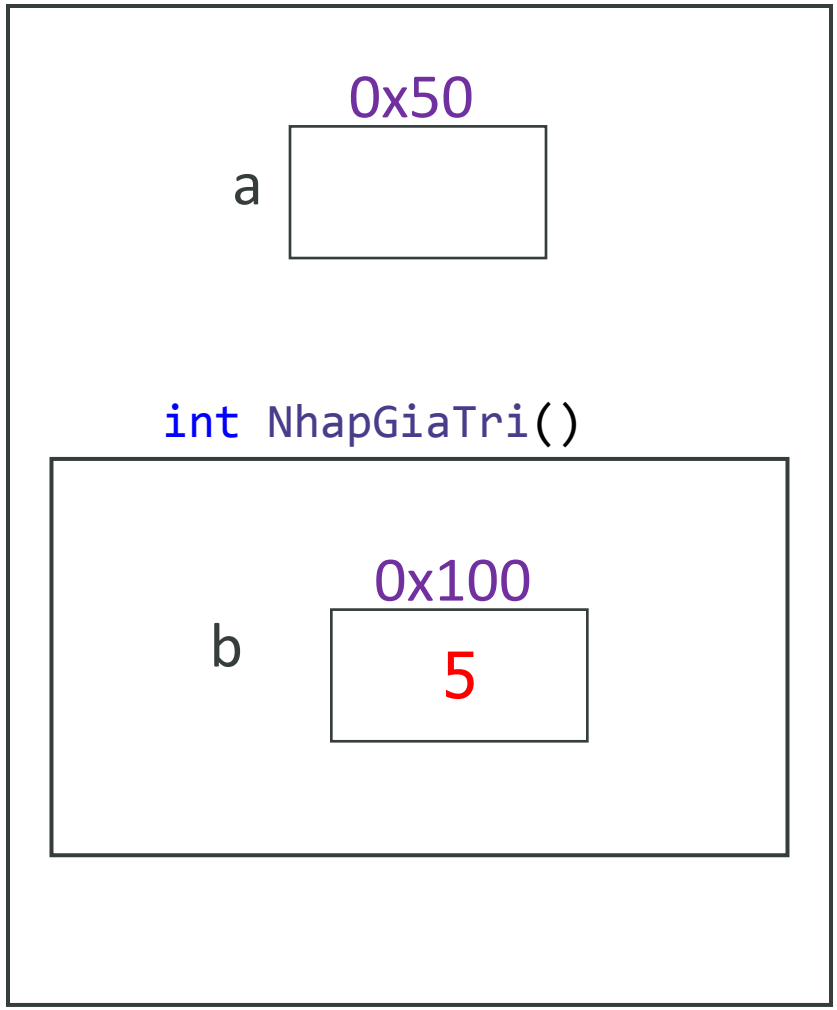
```
int main() {
    int a;
    cout << "Nhap gia tri vao";
    cin >> a;
    cout << a;
}
```



1.7 Con trỏ và hàm

```
// Cách 1:
int NhapGiaTri(){
  int b;
  cout << "Nhap gia tri vao";
  cin >> b;
  return b;
}

int main() {
  int a;
  a = NhapGiaTri();
  cout << a;
}
```



1.7 Con trỏ và hàm

? Hỏi cách này có đúng không

```
void NhapGiaTri(int b) {
  cout << "Nhap gia tri vao";
  cin >> b;
}

int main() {
  int a;
  NhapGiaTri(a);
  cout << a;
}
```

1.7 Con trỏ và hàm

```
// Cách 2
void NhapGiaTri(int *b) {
  cout << "Nhap gia tri vao";
  cin >> *b;
}

int main() {
  int a;
  NhapGiaTri(&a);
  cout << a;
}
```

1.7 Con trỏ và hàm

```
// Cách 3
void NhapGiaTri(int &b) {
  cout << "Nhap gia tri vao";
  cin >> b;
}

int main() {
  int a;
  NhapGiaTri(a);
  cout << a;
}
```

Bài tập 1

- Tìm lỗi sai trong đoạn chương trình sau:

```
int main() {
        int x, *p;
        x = 10;
        *p = x;
        return 0;
}
```

Bài tập 2

Dùng C++ viết một đoạn chương trình với 2 biến:

+ Biến i có kiểu int với giá trị khởi đầu là 12

+ Biến p1 là một con trỏ trỏ tới vùng nhớ kiểu int.

? Hãy dùng biến p1 để thay đổi giá trị của biến i từ 12 sang 24.

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

int main() {
    int i = 12;
    int *p1;
    // i = 24;
    p1 = &i;
    *p1 = 24;
    cout << *p1 << " "
    << i << endl;
}
```

Bài tập 3

- Hãy viết hàm hoán đổi giá trị của 2 tham số
- Giải:

```
// Cách 1
void Swap(int *a, int *b) {
     int temp = *b;
     *b = *a;
     *a = temp;
}

int main() {
     int x=7, y=8;
     Swap(&x, &y);
     cout << "x= "<< x << ",
     y= " << y;
}
```

```
//Cách 2
void Swap(int &a, int &b) {
     int temp = b;
     b = a;
     a = temp;
}

int main() {
     int x=7, y=8;
     Swap(x, y);
     cout << "x= "<< x << ",
     y= " << y;
}
```

Một số lưu ý

- Con trỏ là khái niệm quan trọng và khó nhất trong C++. Mức độ thành thạo C++ được đánh giá qua mức độ sử dụng con trỏ.
- 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; => sai
```

2. Con trỏ và Mảng 1 chiều

2.1 Mảng 1 chiều và cách lấy địa chỉ

2.2 Mảng 1 chiều và hằng con trỏ

2.3 Các phép toán số học trên con trỏ

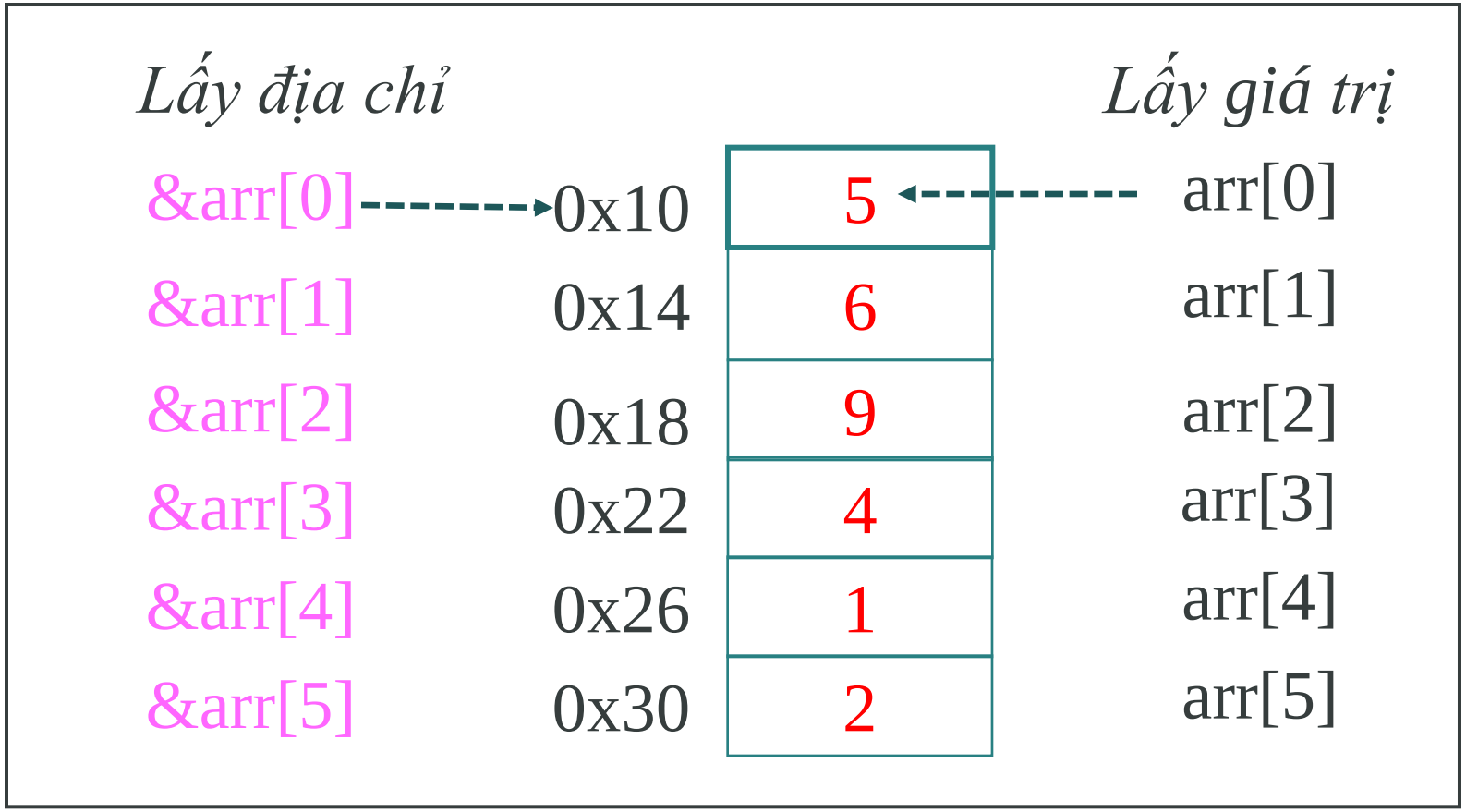
2.4 Con trỏ và mảng 1 chiều

Bài tập

Một số lưu ý

2.1 Mảng 1 chiều và cách lấy địa chỉ

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



Memory Layout

2.2 Mảng 1 chiều và hằng con trỏ

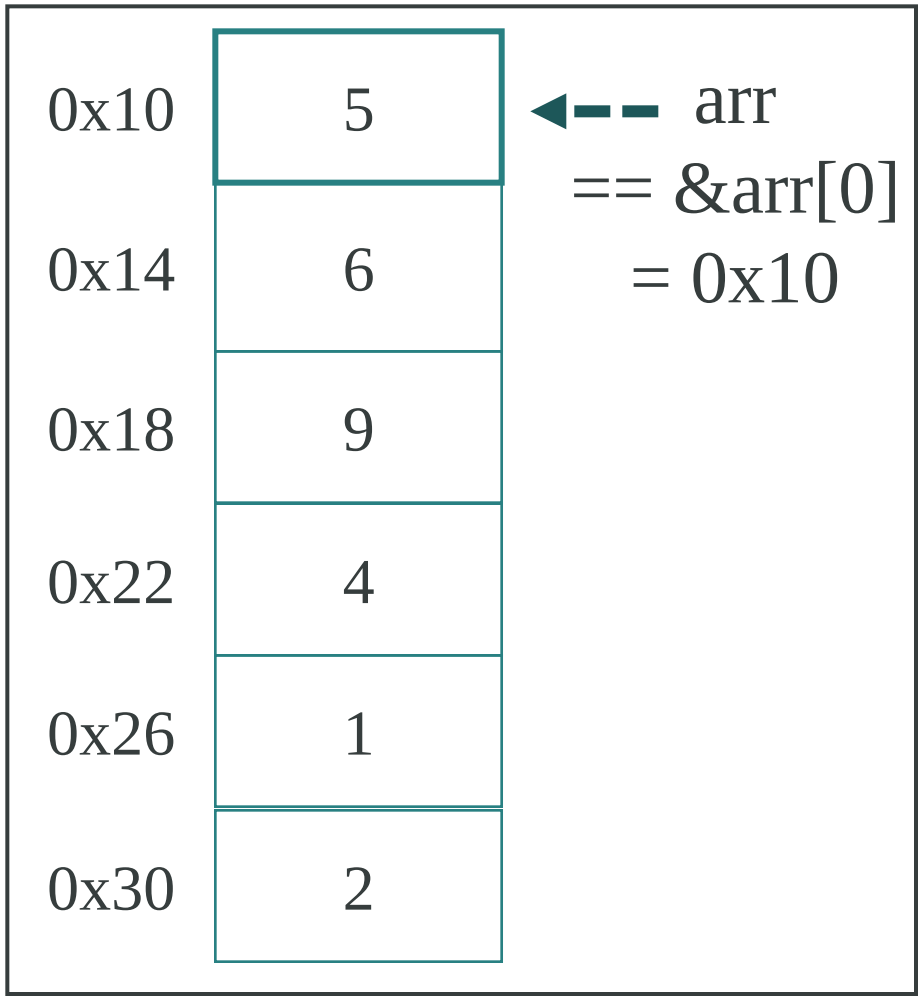
- Cho mảng 1 chiều

```
int arr[6] = {5,
6, 9, 4, 1, 2};
```

  - Tên mảng arr là một hằng con trỏ

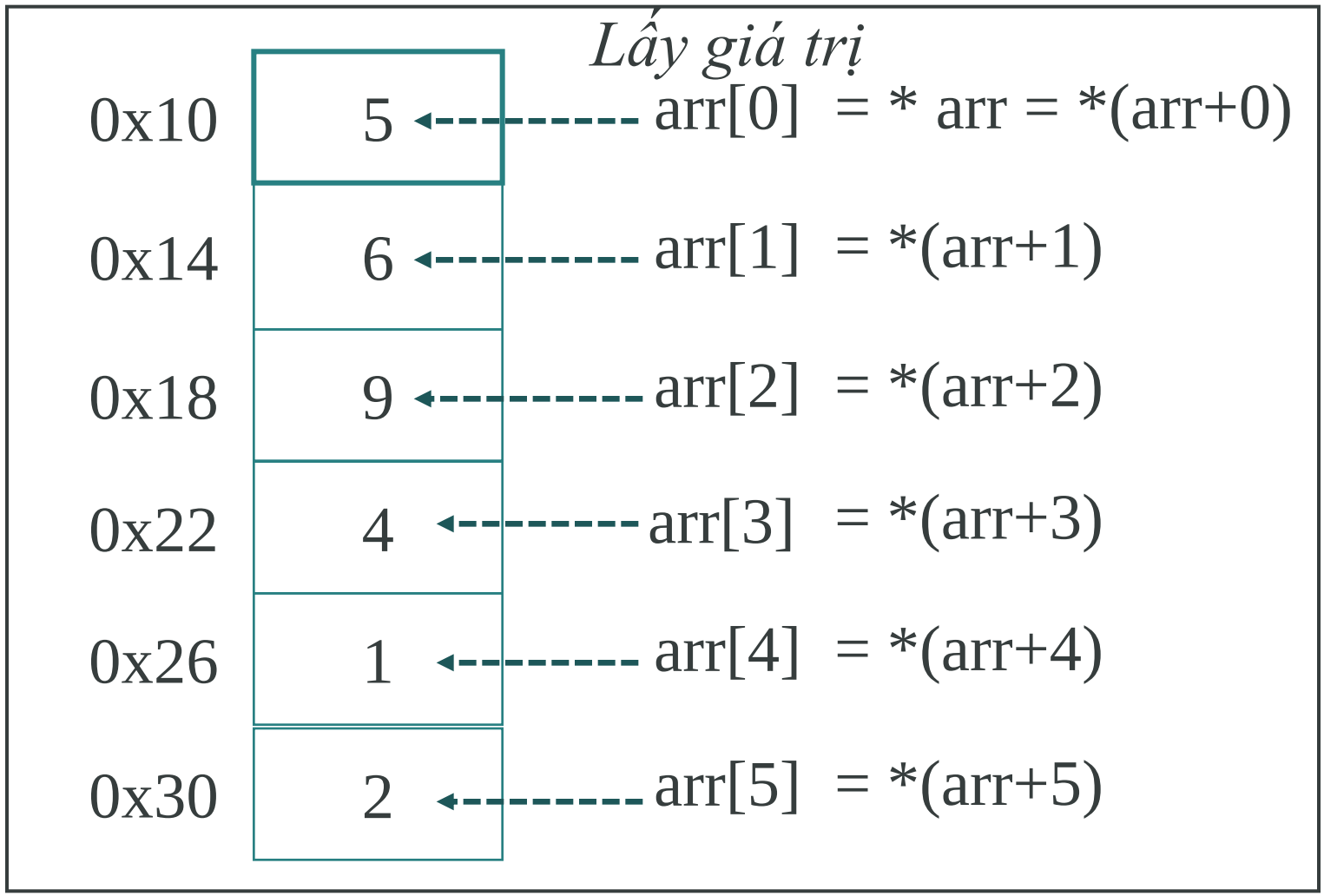
    ➔ không thể thay đổi giá trị của hằng này.
  - arr là địa chỉ đầu tiên của mảng

    ➔arr == &arr[0]



Memory Layout

2.2 Mảng 1 chiều và hằng con trỏ



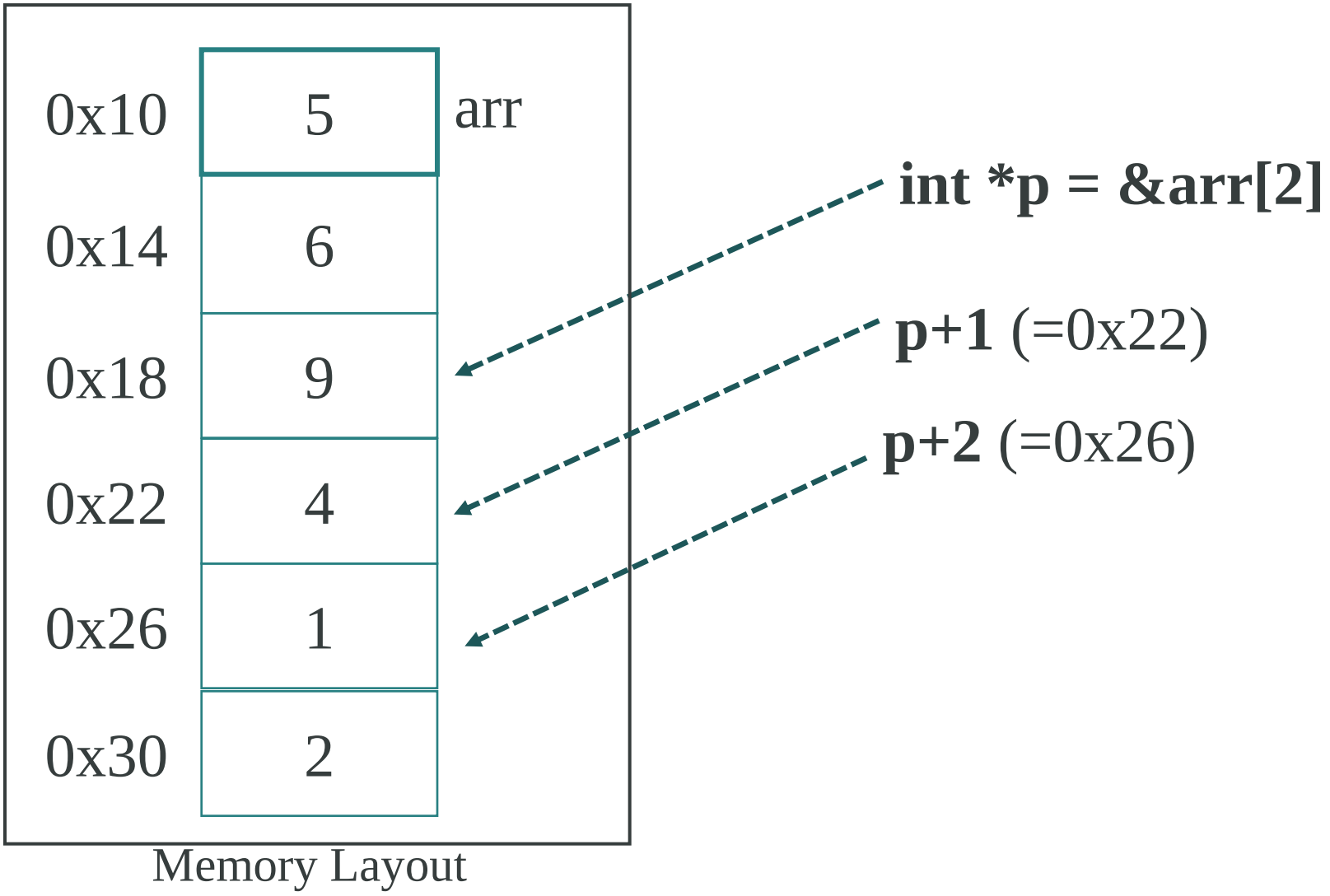
Memory Layout

2.3 Các phép toán số học trên con trỏ

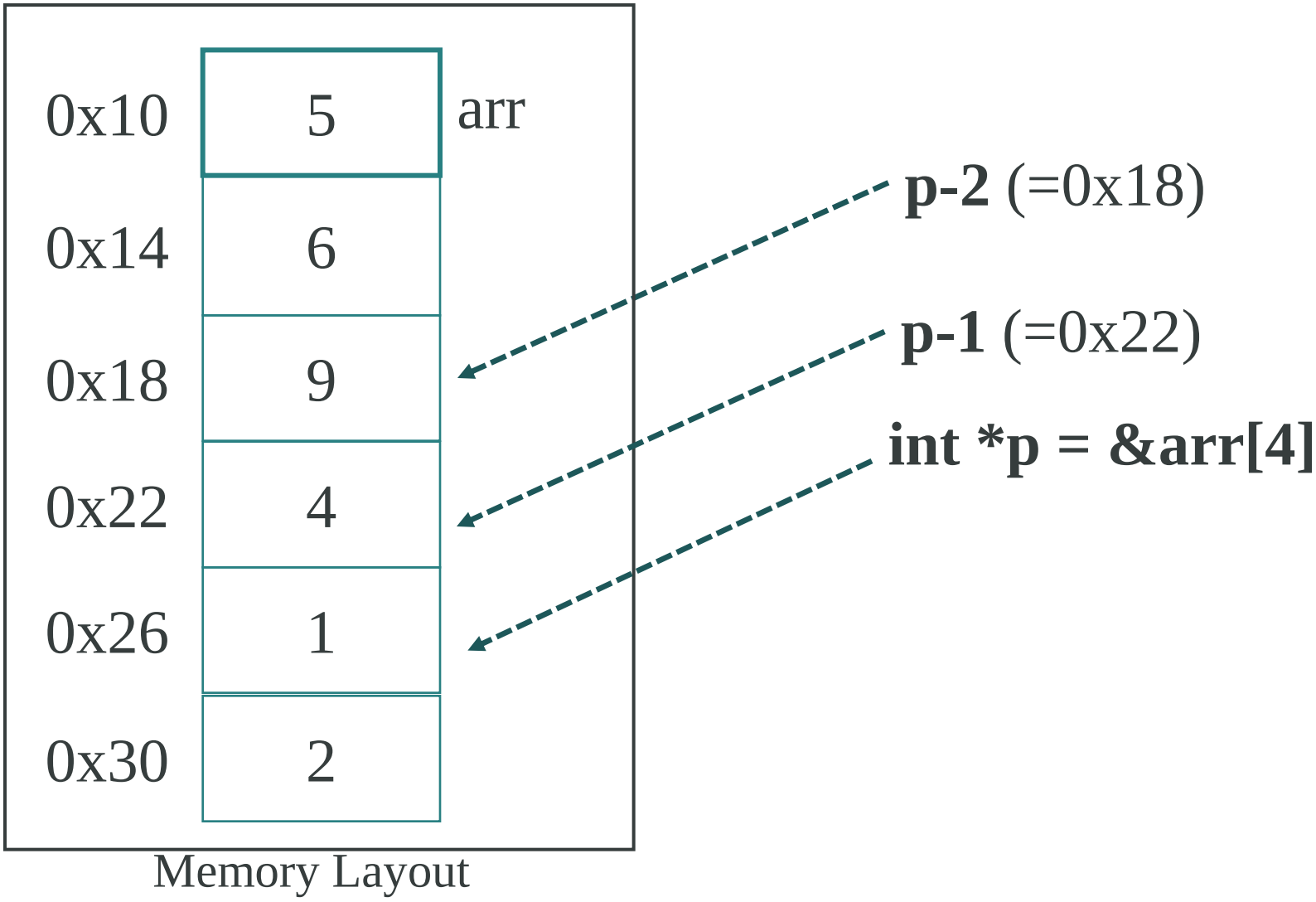
Phép toán số học trên con trỏ- **Pointer Arithmetics**

- 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 – –

2.3 Các phép toán số học trên con trỏ



2.3 Các phép toán số học trên con trỏ

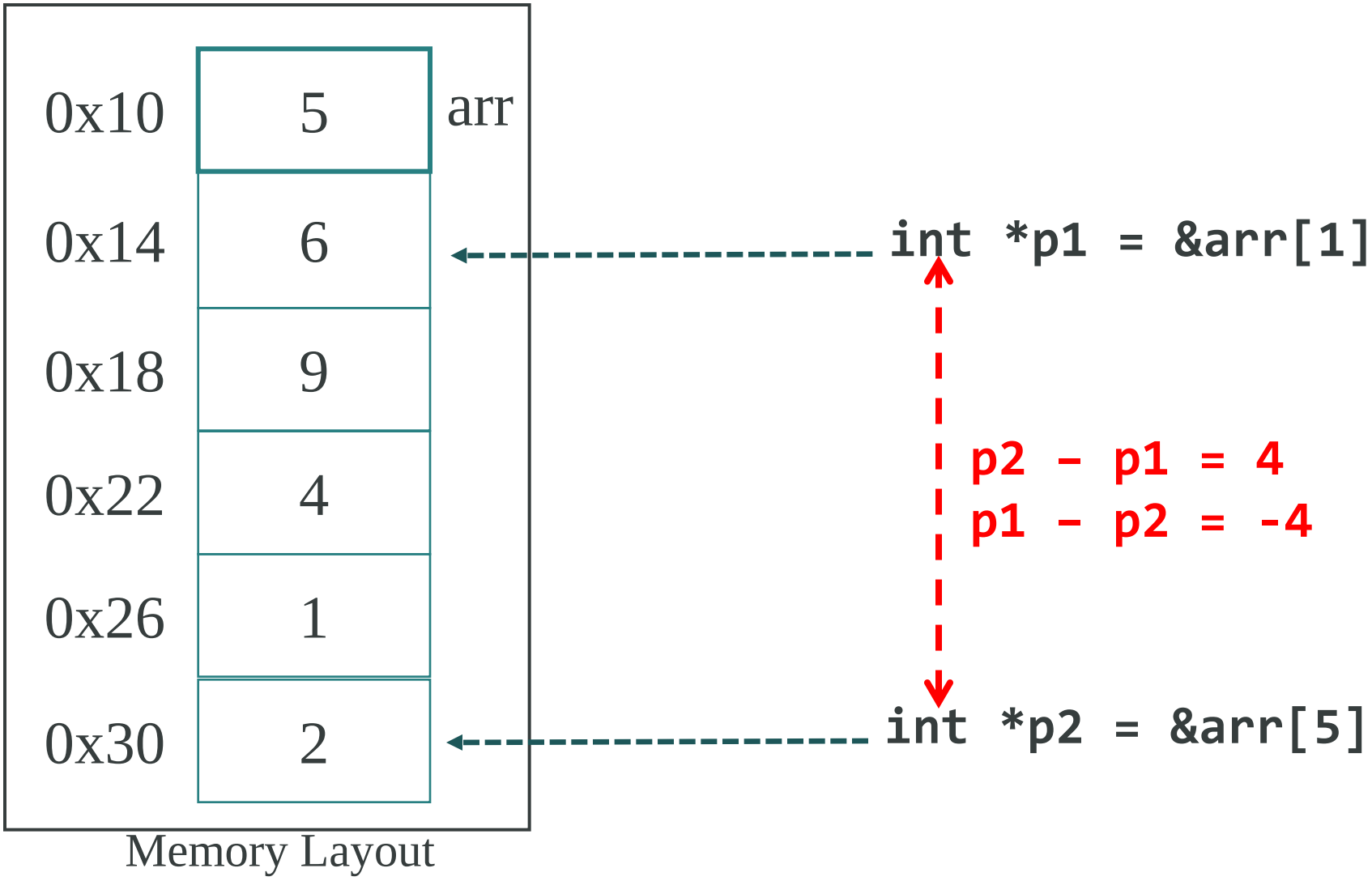


2.3 Các phép toán số học trên con trỏ

- Phép toán tính khoảng cách giữa 2 con trỏ
  - <kiểu dữ liệu> *p1, *p2;
  - p1 – p2 cho ta khoảng cách (theo số phần tử) giữa hai con trỏ (cùng kiểu)
- Các phép toán so sánh
  - Phép so sánh: So sánh địa chỉ giữa hai con trỏ (thứ tự ô nhớ)

    ==, !=, >, >=, <, <=
- Không thể thực hiện các phép toán: * / %

2.3 Các phép toán số học trên con trỏ

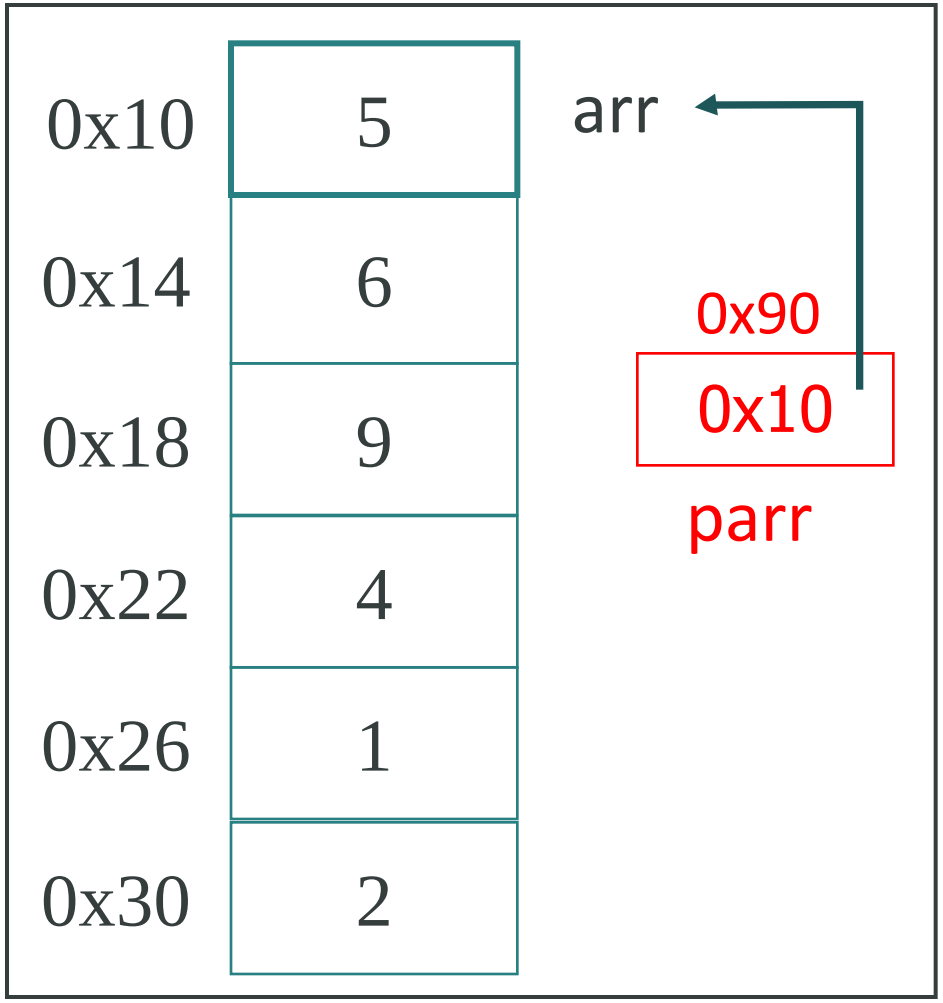


2.4 Con trỏ và mảng 1 chiều

```
int arr[6] = {5, 6, 9, 4, 1, 2};
int *parr;
```

- Xét đoạn code sau:

```
// Cách 1
parr = arr;
// Cách 2
parr = &arr[0];
```



Memory Layout

2.4 Con trỏ và mảng 1 chiều

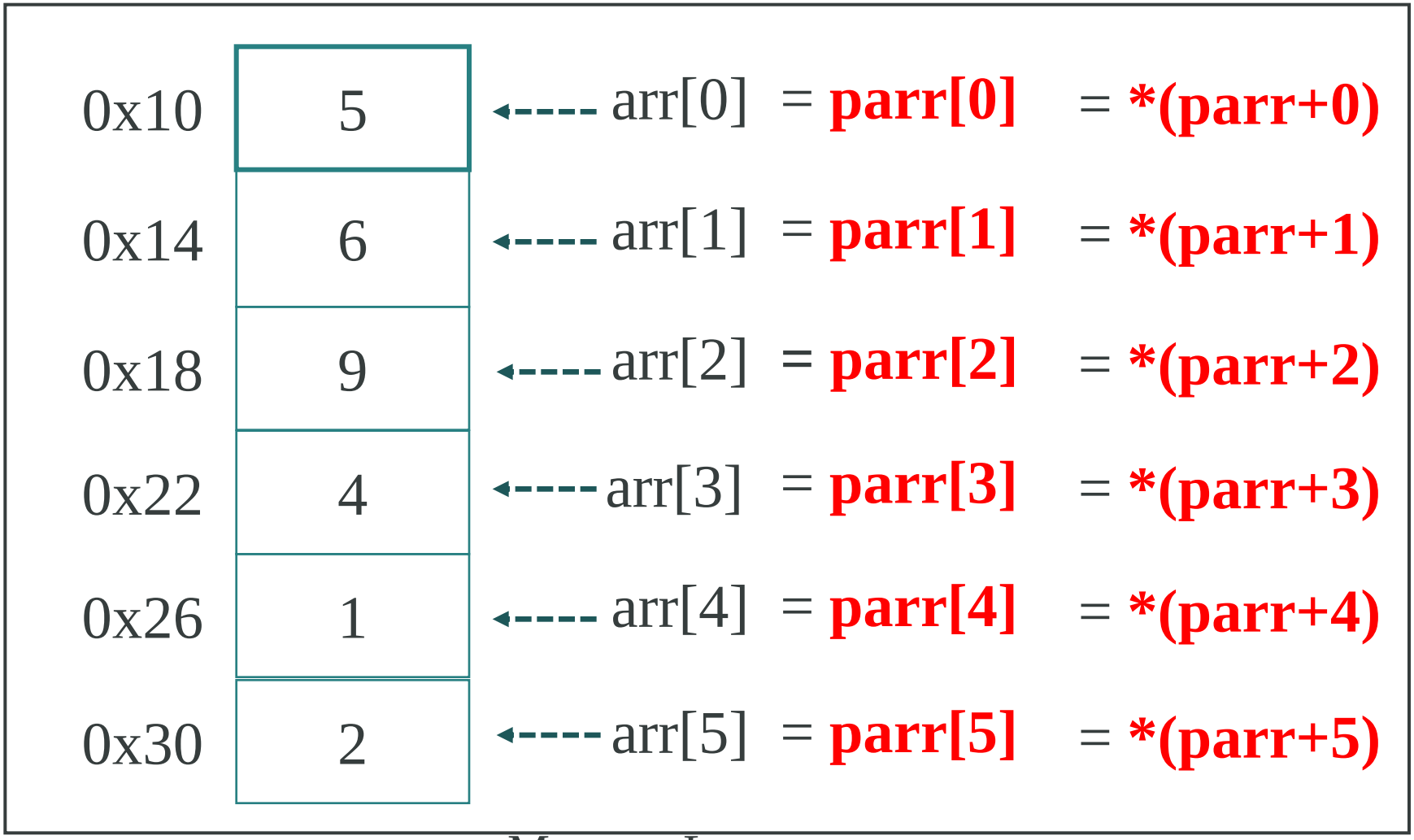
- Truy xuất tới **giá trị** của phần tử thứ ***i*** của mảng (xét i là chỉ số hợp lệ của mảng):

```
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
```

2.4 Con trỏ và mảng 1 chiều



Memory Layout

2.5 Truyền mảng 1 chiều cho hàm

- Xét 2 đoạn chương trình sau. Tìm lỗi sai và giải thích.

SAI

```
int main() {
   int a[] = {1,2,3,4,5,6};

   for (int i = 0; i<6; i++)
       printf("%d", *(a++));
}
```

ĐÚNG

```
void xuat(int *a, int n) {
   for (int i = 0; i<n; i++)
       printf("%d", *(a++));
}

int main() {
   int a[] = {1,2,3,4, 5, 6};
   xuat(a, 6);
}
```

Lý do: Đối số mảng truyền cho hàm không phải hằng con trỏ.

Bài tập 1

Nhóm các lệnh sau lại thành 2 nhóm tương ứng nhóm lấy địa chỉ biến và nhóm lấy giá trị biến:

```
*(parr+i)
arr[i]
&parr[i]
&arr[i]
parr[i]
arr+i
parr + i
*(arr+i)
```

Bài tập 2

Cho mảng 1 chiều a có 10 phần tử, biến con trỏ p trỏ tới mảng 1 chiều a.

a. Hãy dùng con trỏ p để gán giá trị 100 cho phần tử thứ 5 của mảng.

b. Hãy viết chương trình nhập và xuất mảng 1 chiều thông qua con trỏ p.

Lời giải

```
#include <iostream>
using namespace std;
const int n = 10;
int main() {
    int a[n], *p = a;
    *(p+5) =100; // câu a
    for (int i = 0; i < n; i++) {// câu b
         cin >> *(p + i);
    }
    for (int i = 0; i < n; i++) {// câu b
         cout << *(p + i) << " ";
    }
}
```

Bài tập 2

- Tạo biến str lưu chuỗi “hello class”, sau đó tạo biến con trỏ p lưu trữ địa chỉ đầu tiên của chuỗi. Hãy thực hiện chuyển chuỗi str thành chuỗi ký tự in hoa “HELLO CLASS” thông qua sử dụng con trỏ p.

Lời giải

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

int main() {
	char str[20] = "hello class";
	char *p;
	int n = strlen(str);
	p = str;
	for (int i = 0; i<=n; i++)
		p[i] = toupper(p[i]);
	cout << p;
}
```

Một số lưu ý

- Không thực hiện các phép toán nhân, chia, lấy phần dư.
- Tăng/giảm con trỏ n đơn vị có nghĩa là tăng/giảm giá trị của nó n*sizeof (<kiểu dữ liệu mà nó trỏ đến>)
- Không thể tăng/giảm biến mảng. Hãy gán một con trỏ đến địa chỉ đầu của mảng và tăng/giảm nó.

Câu hỏi lý thuyết

- Bài 1: Toán tử nào dùng để xác định địa chỉ của một biến?
- Bài 2: Toán tử nào dùng để xác định giá trị của biến do con trỏ trỏ đến?
- Bài 3: Phép lấy giá trị gián tiếp là gì?
- Bài 4: Cho biến daa kiểu int. Khai báo và khởi tạo con trỏ pdaa trỏ đến biến này. Sau đó gán giá trị 100 cho biến daa sử dụng hai cách trực tiếp và gián tiếp.
- Bài 5: Các phần tử trong mảng được sắp xếp trong bộ nhớ như thế nào?
- Bài 6: Cho mảng một chiều data. Trình bày 2 cách lấy địa chỉ phần tử đầu tiên của mảng này.
- Bài 7: Cho con trỏ p1 trỏ đến phần tử thứ 3 còn con trỏ p2 trỏ đến phần tử thứ 4 của mảng int. Tính p2 – p1 ?

Câu hỏi lý thuyết

- Bài 8: Tìm lỗi trong đoạn code sau:

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

int main() {
    int *x, y = 2;
    *x = y;
    *x += y++;
    cout << *x << y;
}
```

- Bài 9: Cho đoạn chương trình sau:

```
float pay;
float *ptr_pay;
pay = 2313.54;
ptr_pay = &pay;
```

Hãy cho biết giá trị của:

a. pay

b. *ptr_pay

c. *pay

d. &pay

Bài tập bắt buộc

1. Cho biết ý nghĩa của các khai báo và câu lệnh; Tìm lỗi sai trong đoạn code và giải thích
2. Cho mảng 1 chiều a có 10 phần tử, biến con trỏ p trỏ tới mảng 1 chiều a. Hãy dùng con trỏ p để gán giá trị 100 cho phần tử thứ 5 của mảng. Hãy viết chương trình nhập và xuất mảng 1 chiều thông qua con trỏ p.
3. Tạo biến str lưu chuỗi "hello class", sau đó tạo biến con trỏ p lưu trữ địa chỉ đầu tiên của chuỗi. Hãy thực hiện chuyển chuỗi str thành chuỗi ký tự in hoa "HELLO CLASS" thông qua sử dụng con trỏ p.
4. Viết hàm nhập một dãy số thực A tùy ý trong đó có sự cấp phát động. Viết hàm sao chép dãy số thực A (được nhập bởi hàm trên) sang một dãy B trong đó có sự giải phóng vùng nhớ cấp phát động và cấp phát lại ở dẫy B.
5. Làm lại các bài tập về mảng dùng con trỏ

Chúc các em học tốt!