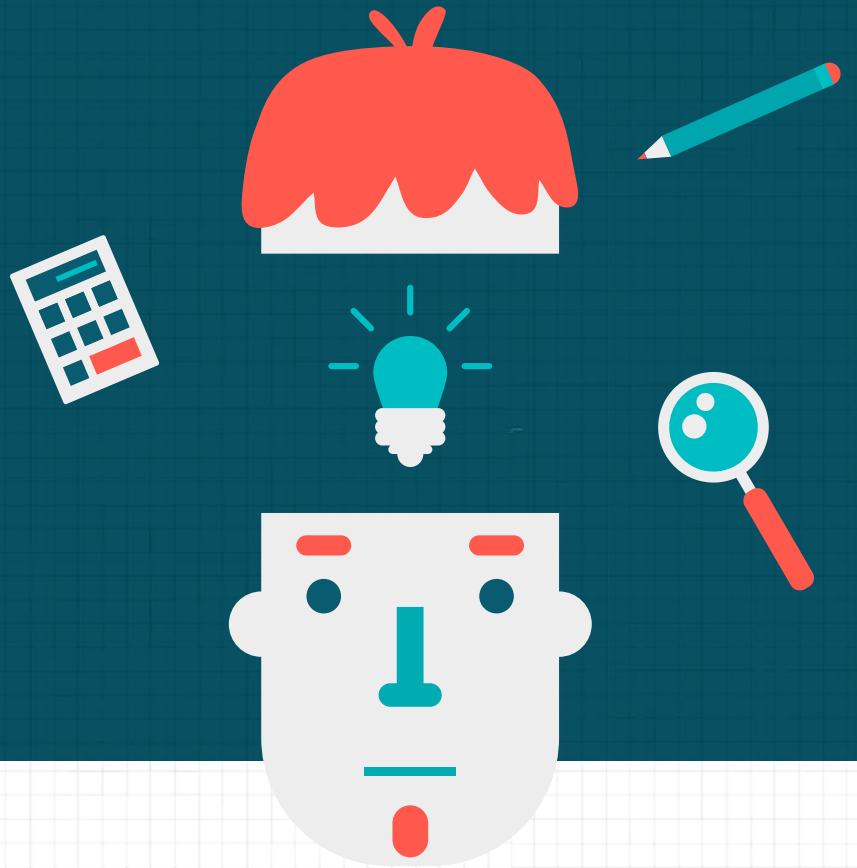
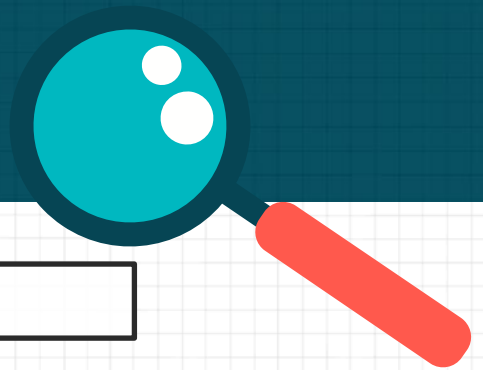
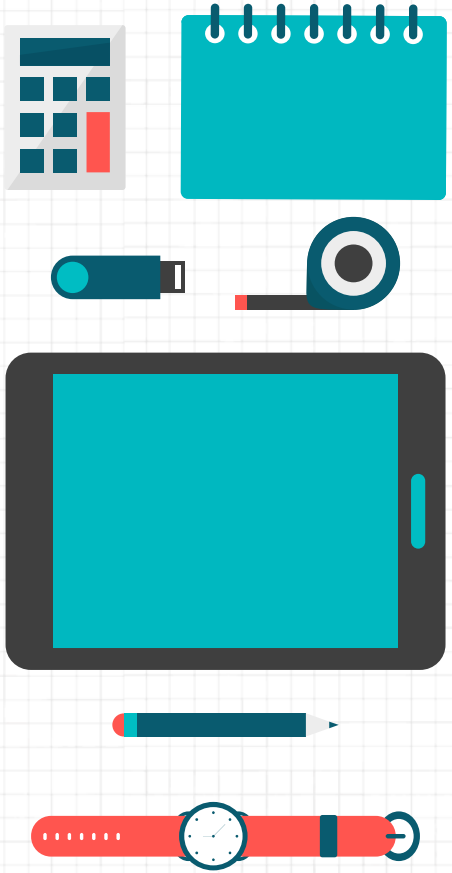
Bài 1
Con trỏ



Ths. Phạm Minh Hoàng

Nội dung



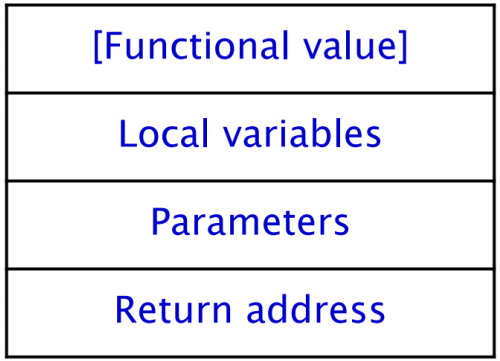


04

Con trỏ và hàm

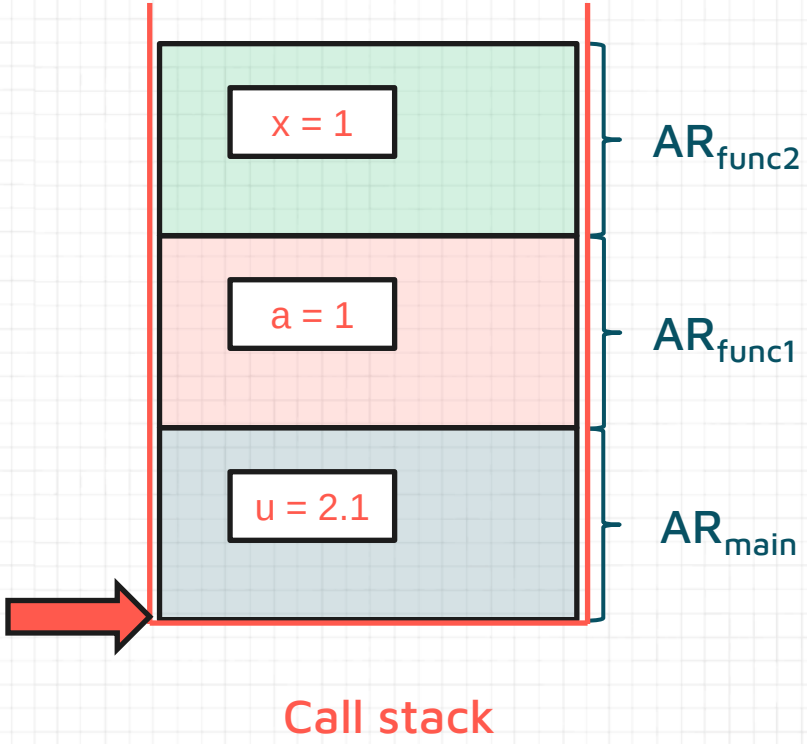
Call stack

- Ngăn xếp (stack): danh sách các phần tử được thiết lập theo nguyên tắc "vào trước, ra sau") (LIFO – Last In First Out)
- Call stack: ngăn xếp gồm các bản ghi hoạt động (active record-AR). Mỗi AR tương ứng một lời gọi hàm
- Một AR bao gồm
  - Vùng nhớ lưu các tham số của hàm
  - Vùng nhớ lưu các biến cục bộ
  - Giá trị trả về của hàm (có thể có hoặc không)
  - Địa chỉ trả về
  - v..v



Call stack

```
void func2(int x){
    x = x + 1;
}
void func1(){
    int a = 1;
    func2(a);
}
void main(){
    float u = 2.1;
    func1();
}
```



Truyền tham số cho hàm

- Ba cách truyền tham số cho hàm
  - Truyền tham trị

```
<return_type> func(<type1> param1, <type2> param2);
```

  - Truyền con trỏ

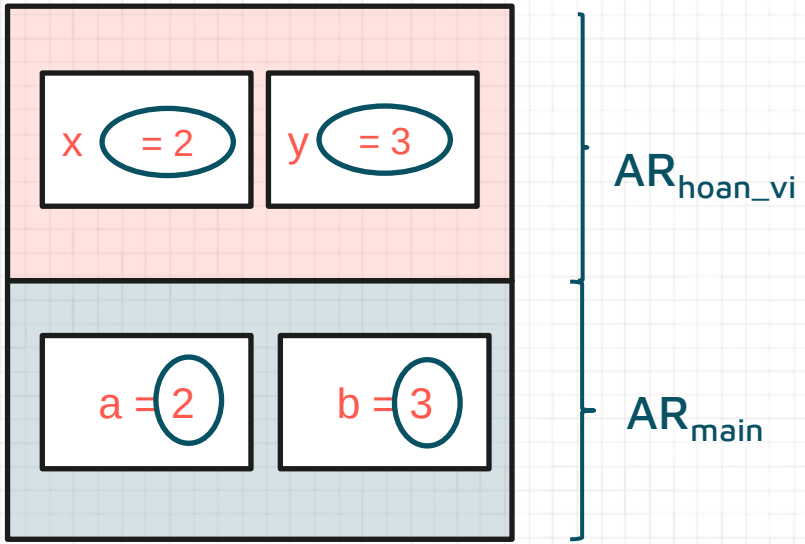
```
<return_type> func(<type1>* param1, <type2>* param2);
```

  - Truyền tham chiếu

```
<return_type> func(<type1>& param1, <type2>& param2);
```

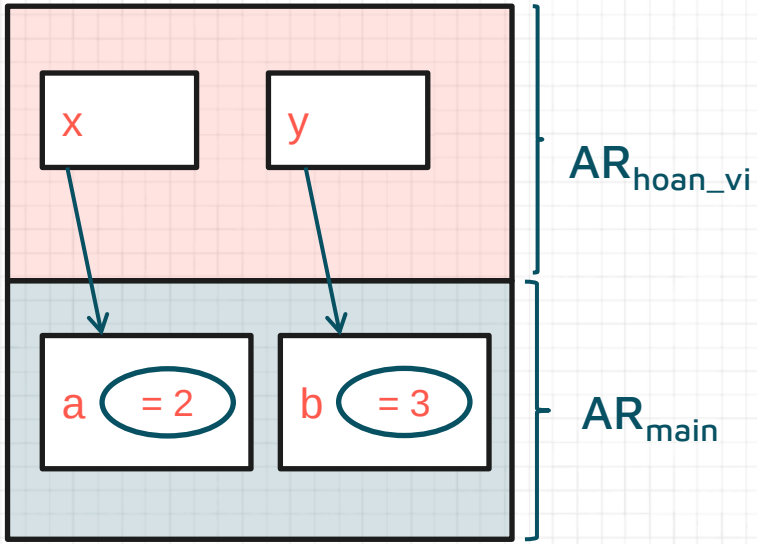
Truyền tham trị

```
void hoan_vi(int x, int y){
    int t = x;
    x = y;
    y = t;
}

void main(){
    int a = 2, b = 3;
    hoan_vi(a, b);
    cout << a << b;
}
```



Truyền con trỏ

```
void hoan_vi(int* x, int* y){
    int t = *x;
    *x = *y;
    *y = t;
}

void main(){
    int a = 2, b = 3;
    hoan_vi(&a, &b);
    cout << a << b;
}
```



Truyền tham chiếu

- Khái niệm tham chiếu: là tên gọi khác của một đối tượng
- Khai báo tham chiếu

```
<kiểu dữ liệu>& <tên tham chiếu> = <đối tượng tham chiếu>;
```

- Ví dụ:

```
int a = 1;
int& ref1 = a;
int &ref2 = a;
```

- Tham chiếu phải được gán giá trị ngay khi khởi tạo

```
int a = 1;
int& ref1 = a;
int& ref2; //error
```

Truyền tham chiếu

- Tương tác với tham chiếu cũng giống như tương tác với đối tượng được tham chiếu tới và ngược lại

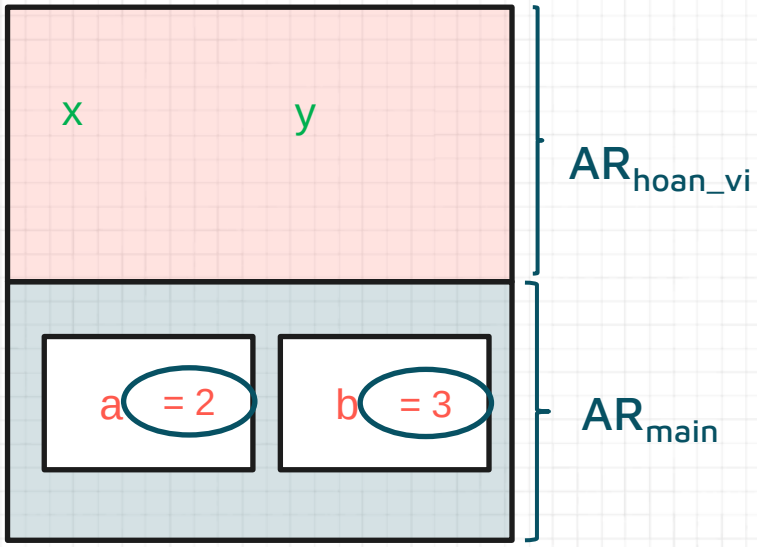
```
int a = 1;
int& ref = a;
a = 8;
cout << ref;
++ref;
cout << a;
```

- Tham chiếu sẽ không thay đổi sau khi được khai báo

```
int a = 1, b = 2;
int& ref = a;
ref = b;//chỉ thay đổi giá trị của a, không tham chiếu vào b
```

Truyền tham chiếu

```
void hoan_vi(int& x, int& y){
    int t = x;
    x = y;
    y = t;
}

void main(){
    int a = 2, b = 3;
    hoan_vi(a, b);
    cout << a << b;
}
```



Bài tập

- Kết quả sẽ in ra bao nhiêu nếu hàm func() là hàm func1(), func2(), func3()

```
void main(){
    int a = 5;
    int* ptr = &a;
    func(ptr);
    if (ptr)
        cout << *ptr;
    else
        cout << " ptr is null";
}
```

```
void func1(int* p){
    p = NULL;
}
```

```
void func2(int*& p){
    p = NULL;
}
```

```
void func3(int* p){
    *p = 6;
}
```

Trả về giá trị cho hàm

- Ba cách trả về giá trị cho hàm
  - Trả về giá trị

```
<return_type> func(<param>);
```

  - Trả về con trỏ

```
<return_type>* func(<param>);
```

  - Trả về tham chiếu

```
<return_type>& func(<param>);
```

Trả về giá trị cho hàm

- Tìm số nhỏ nhất trong mảng

Trả về giá trị

```
int getMin(int a[], int n){
  int minIdx = 0;
  for(int i = 0; i < n; i++){
     if(a[i] < a[minIdx])
        a[minIdx] = a[i];
  }
  return a[minIdx];
};
```

Trả về con trỏ

```
int* getMin(int a[], int n){
  int minIdx = 0;
  for(int i = 0; i < n; i++){
    if(a[i] < a[minIdx])
      a[minIdx] = a[i];
  }
  return &a[minIdx];
};
```

Trả về tham chiếu

```
int& getMin(int a[], int n){
   int minIdx = 0;
   for(int i = 0; i < n; i++){
       if(a[i] < a[minIdx])
           a[minIdx] = a[i];
   }
   return a[minIdx];
};
```

Trả về giá trị cho hàm

- Chuyện gì xảy ra trong 2 đoạn code dưới?

1.
```
int* doSth(){
  int a = 10;
  int* p = &a;
  return p;
}

void main(){
  int* p = doSth();
  int a = 12 + *p;
}
```

2.
```
int& doSth(){
  int a = 10;
  int& p = a;
  return p;
}

void main(){
  int& p = doSth();
  int a = 12 + p;
}
```

Không trả về con trỏ hay tham chiếu với các biến cục bộ trong hàm

So sánh

- So sánh các cách truyền tham số và trả về giá trị cho hàm

	Truyền/trả về giá trị	Truyền/trả về con trỏ	Truyền/trả về tham chiếu
Giống nhau	Sao chép dữ liệu		
Khác nhau	Dữ liệu sao chép là giá trị	Dữ liệu sao chép là địa chỉ	Dùng chung vùng nhớ

Bài tập

- Cho biết kết quả trong đoạn code dưới

```
int& get(int* p, int i){
    return p[i];
}

void main(){
    int a[3]={1, 2, 3};
    int& b = get(a, 1);
    b = 10;
    cout << a[0] << a[1] << a[2];
}
```

Con trỏ hàm

- Hàm cũng được cấp phát vùng nhớ → cũng có địa chỉ
- Con trỏ giữ địa chỉ vùng nhớ của hàm → con trỏ hàm
- Con trỏ hàm có thể trỏ đến bất kỳ hàm nào có cùng prototype
- Khai báo con trỏ hàm

```
<return_type> (*func_pointer)(<param>);
```

  - <return_type>: kiểu dữ liệu trả về
  - func_pointer: con trỏ hàm
  - <param>: danh sách tham số

Con trỏ hàm

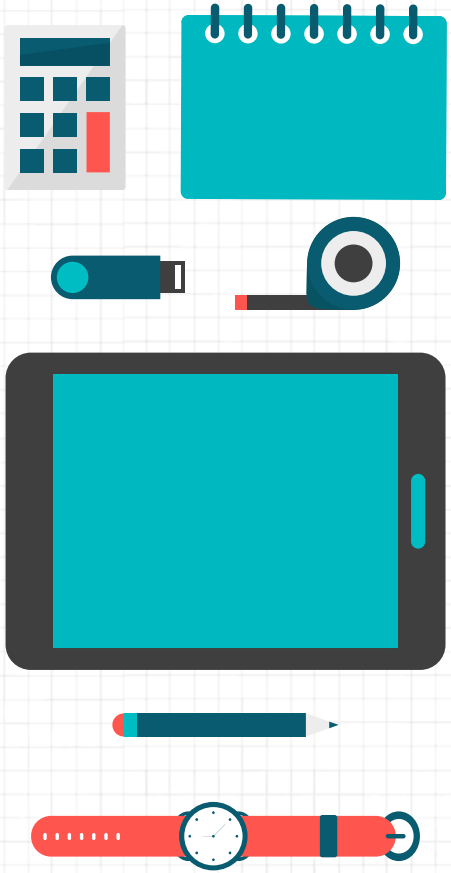
Dùng con trỏ hàm

```
int add(int a, int b){
    return a + b;
}
int sub(int a, int b){
    return a - b;
}
void main(){
    int (*funp)(int, int) = nullptr;
    funp = add;
    int c = funp(3, 4);
    funp = sub;
    int d = funp(10, 2);
    cout << c << d;
}
```

Con trỏ hàm

Truyền con trỏ hàm như tham số của hàm

```
int add(int a, int b);
int sub(int a, int b);

void dump(int x, int y, int (*p)(int, int)){
    int r = p(x, y);
    cout << r;
}

void main(){
    int a = 32, b = 6;
    dump(a, b, add);
    dump(a, b, sub);
}
```



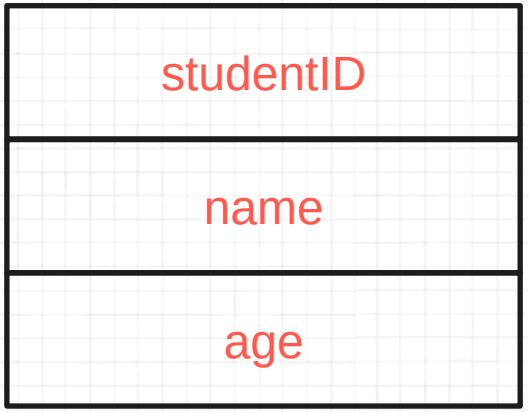
05

Con trỏ cấu trúc

Cấu trúc

- Cấu trúc dùng để lưu trữ một đối tượng với nhiều thuộc tính
- Các thuộc tính của cấu trúc được lưu trong những vùng nhớ liên tiếp nhau

```
struct Student{
    char studentID[20];
    char name[50];
    int age;
}
void main(){
    Student a;
}
```



Toán tử truy xuất trường dữ liệu

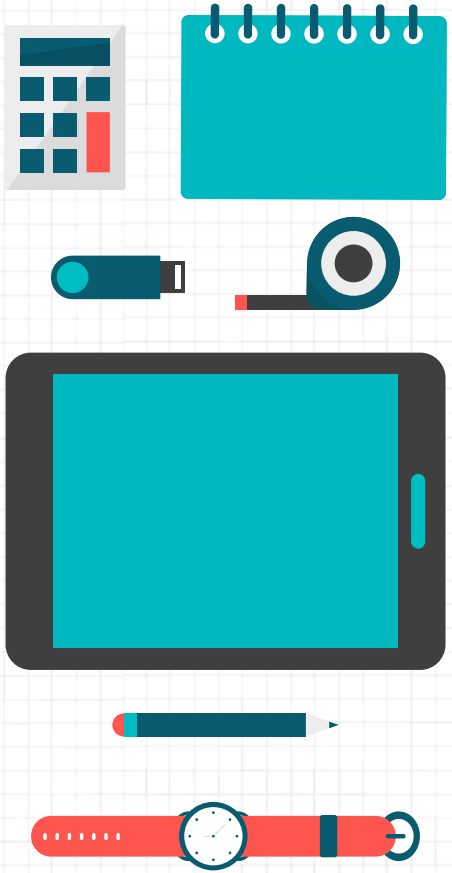
- Dùng toán tử .
- Cú pháp: `<biến cấu trúc>.<trường dữ liệu>;`
- Ví dụ

```
struct Student{
    char studentID[20];
    char name[50];
    int age;
}
void main(){
    Student a;
    cout << a.age;
}
```

Con trỏ cấu trúc

- Con trỏ cấu trúc trỏ đến vùng nhớ của trường dữ liệu đầu tiên trong cấu trúc
- Truy xuất đến trường dữ liệu qua con trỏ cấu trúc
  - Cú pháp: `<biến cấu trúc>-><trường dữ liệu>;`
  - Ví dụ

```
struct Student{
    char studentID[20];
    char name[50];
    int age;
}
void main(){
    Student a;
    Student* p = &a;
    cout << p->age;
}
```



06

Một số vấn đề khác

Con trỏ void

- Con trỏ void trỏ đến vùng nhớ không xác định kiểu dữ liệu
  - Không xác định kích thước vùng nhớ do con trỏ void trỏ đến
  - Không thể truy xuất nội dung vùng nhớ do con trỏ void trỏ đến

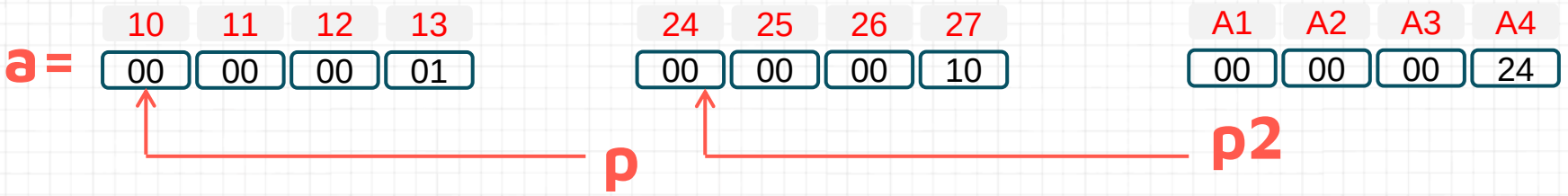
```
int a = 1;
float b = 3.14;

void* p = &a;
void* q = &b

cout << *p; //error
cout << *((float*)q); //ok
```

Con trỏ đến con trỏ

- Con trỏ cũng là biến nên nó cũng có vùng nhớ để lưu trữ
- Cũng có thể lấy địa chỉ vùng nhớ của con trỏ để gán cho 1 con trỏ khác
- Con trỏ trỏ đến vùng nhớ của một con trỏ khác gọi là con trỏ đa cấp
- Khai báo con trỏ cấp 2

Con trỏ đến con trỏ

```
int a = 1;
int* p = &a;
int** p2 = &p;
```



Định nghĩa kiểu

- Các kiểu dữ liệu mới đôi khi khá phức tạp, đặc biệt nếu có con trỏ đi kèm
- Định nghĩa lại kiểu dữ liệu mới gọn hơn
  - Dùng typedef

```
typedef <old_type> <new_type>;
```

  - Dùng using

```
using <new_type> = <old_type>;
```

Dùng typedef

- Dùng typedef

```
struct Student{
    char studentID[20];
    char name[50];
    int age;
}

typedef Student* pStudent

void main(){
    pStudent a;
    cout << a->age;
}
```

Dùng using

- Dùng using

```
struct Student{
    char studentID[20];
    char name[50];
    int age;
}

using pStudent = Student*

void main(){
    pStudent a;
    cout << a->age;
}
```

Bài tập 1

- Cho đoạn chương trình sau

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

- Chương trình có lỗi gì?
- Hãy sửa lại cho đúng

Bài tập 2

- Cho đoạn chương trình sau

```
void main(){
    double m[100];
    double *p1, *p2;
    p1 = m + 2;
    p2 = &m[10];
}
```

- p1 cách p2 bao nhiêu byte?

Bài tập 3

- Cho biết đoạn chương trình xuất ra kết quả bằng bao nhiêu ? Giải thích

```
void main(){
    int x = 1023;
    char* p = (char*)&x;
    cout << p[0] << p[1] << p[2] << p[3];
}
```

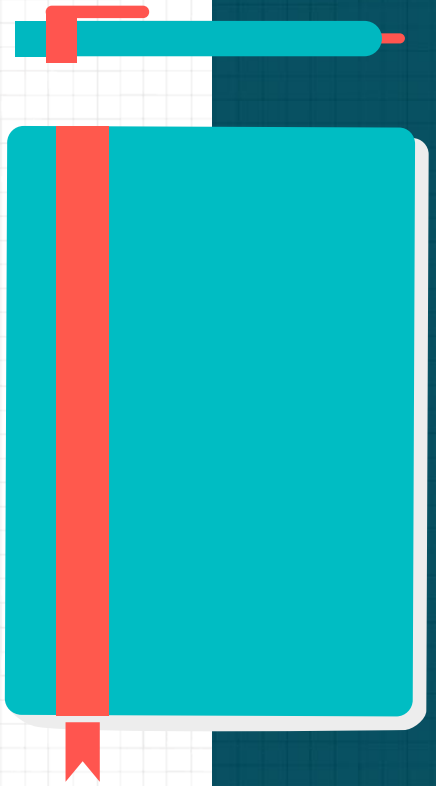
Bài tập 4

- Cho biết đoạn chương trình xuất ra kết quả bằng bao nhiêu ? Giải thích

```
void main () {
  int firstvalue = 5, secondvalue = 15;
  int * p1, * p2;
  p1 = &firstvalue;
  p2 = &secondvalue;
  *p1 = 10;
  *p2 = *p1;
  p1 = p2;
  *p1 = 20;
  cout << "firstvalue is " << firstvalue << '\n';
  cout << "secondvalue is " << secondvalue << '\n';
}
```

Bài tập 5

- Viết chương trình sử dụng con trỏ
  - Nhập mảng số nguyên của N phần tử
  - Tạo mảng mới gồm các số nguyên tố từ mảng đã cho
  - Sắp xếp mảng số nguyên tố theo thứ tự tăng dần hoặc giảm dần (dùng con trỏ hàm)
  - Xuất mảng số nguyên tố ra màn hình



The End!

Do you have any questions?