TRƯỜNG ĐẠI HỌC THỦY LỢI
Khoa CNTT – Bộ môn CNPM

NGUYÊN LÝ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
Mã số: CSE 224

Giảng viên: Trần Thị Ngân

Email: ngantt@tlu.edu.vn

ĐT: 0989040454

Thông tin môn học

- Tên học phần: **Nguyên lý lập trình hướng đối tượng**
- Mã học phần: CSE224
- Số tín chỉ: 03
- Phân bổ thời gian:
  - Giờ giảng lý thuyết: **30** tiết
  - Giờ thực hành: **15** tiết
  - Giờ tự học của sinh viên: **90** tiết

Mục tiêu học phần

- Cung cấp cho sinh viên các kiến thức về nguyên lý cũng như kỹ năng lập trình hướng đối tượng, minh họa trên ngôn ngữ lập trình C++ .
  - **Nguyên lý** lập trình hướng đối tượng: lớp, thuộc tính, phương thức, hàm tạo (constructor), hàm hủy (destructor), kế thừa, nạp chồng (overload), ghi đè (override), hàm ảo, đa hình.
  - Không gian tên, khuôn mẫu (template), xử lý ngoại lệ

Các qui định trong lớp học

- Không làm việc riêng
- Tham gia đầy đủ các buổi học, trao đổi, đóng góp ý kiến tích cực trong buổi học
- Thảo luận/đặt câu hỏi cho các kiến thức chưa hiểu
- Hoàn thành các nội dung và bài kiểm tra theo đúng thời gian qui định
- **Không được vắng mặt hôm kiểm tra**

Tính điểm học phần

- Điểm chuyên cần (10%)
- Điểm giữa kì (20% mỗi bài)

Điểm quá trình: 50%

- Thi cuối kì (Thực hành)

Điểm thi kết thúc học phần: 50%

Nội dung học phần

- **Mở đầu**: Giới thiệu học phần
- **Chương 1:** Nhắc lại về C++
- **Chương 2:** Lớp và đối tượng
- **Chương 3:** Nạp chồng toán tử
- **Chương 4:** Nguyên lý kế thừa
- **Chương 5:** Khuôn mẫu (Template) và thư viện chuẩn (STL)
- **Chương 6:** Hàm ảo và đa hình

CHƯƠNG 1: NHẮC LẠI VỀ C++

Giảng viên: Trần Thị Ngân
Email: ngantt@tlu.edu.vn

Nội dung

1.1 Tổng quan về C++

1.2 Cấu trúc một chương trình C++

1.3 Hàm và nạp chồng hàm

1.4. Nhắc lại về con trỏ

1.5. Nhắc lại về kiểu dữ liệu cấu trúc

1.1 Tổng quan về C++

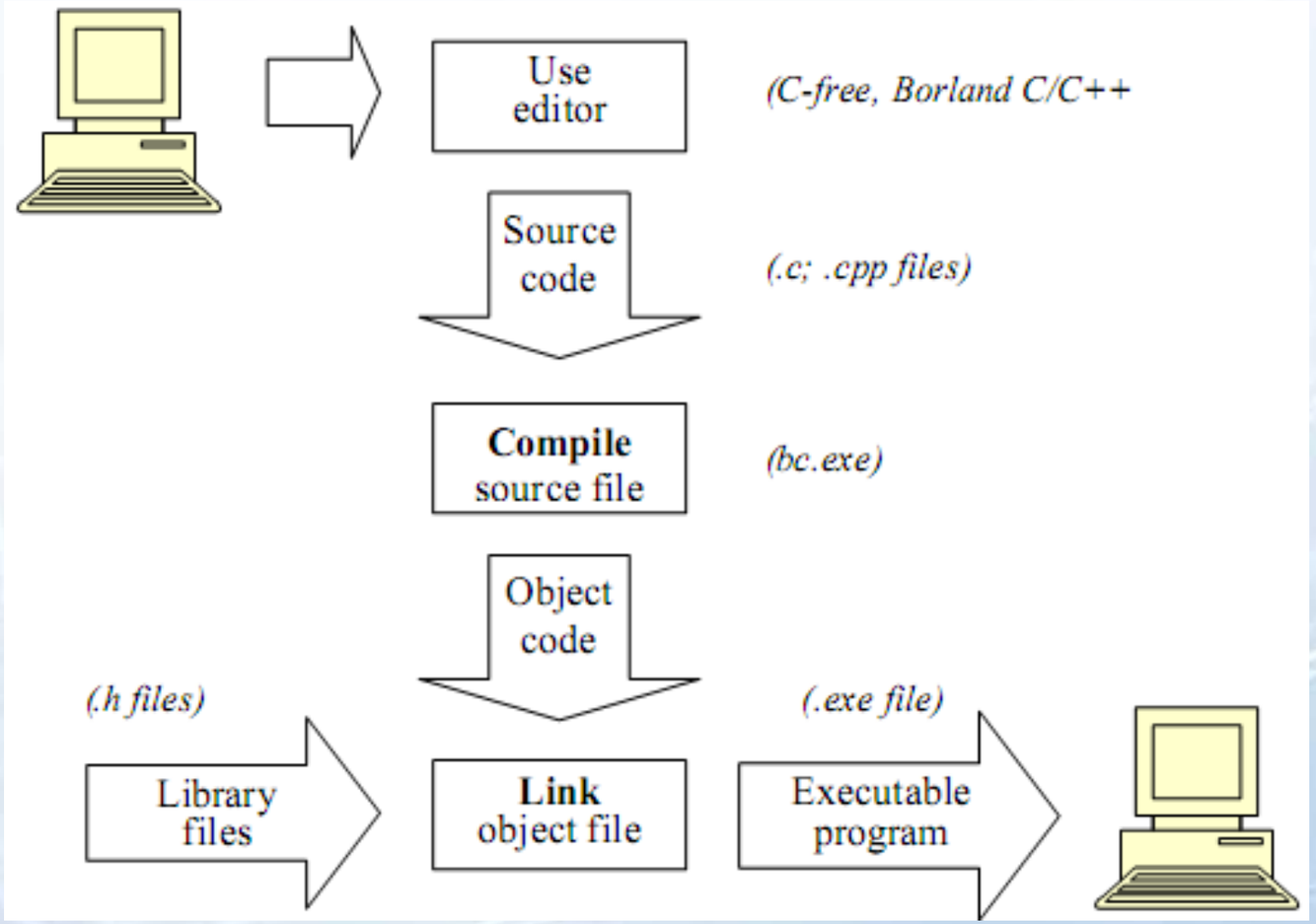
- Tác giả: Bjarne Stroustrup (Mỹ)
- Ý tưởng bắt đầu từ năm 1979
- Được giới thiệu năm 1985
- Phiên bản C++ 2.0 năm 1989
- Phiên bản mới nhất: C++17

- Môn học này chỉ học khoảng 10% kiến thức về C++ và các thư viện của nó
- Cần 3-5 năm để trở thành lập trình viên C++ ở mức độ chuyên nghiệp

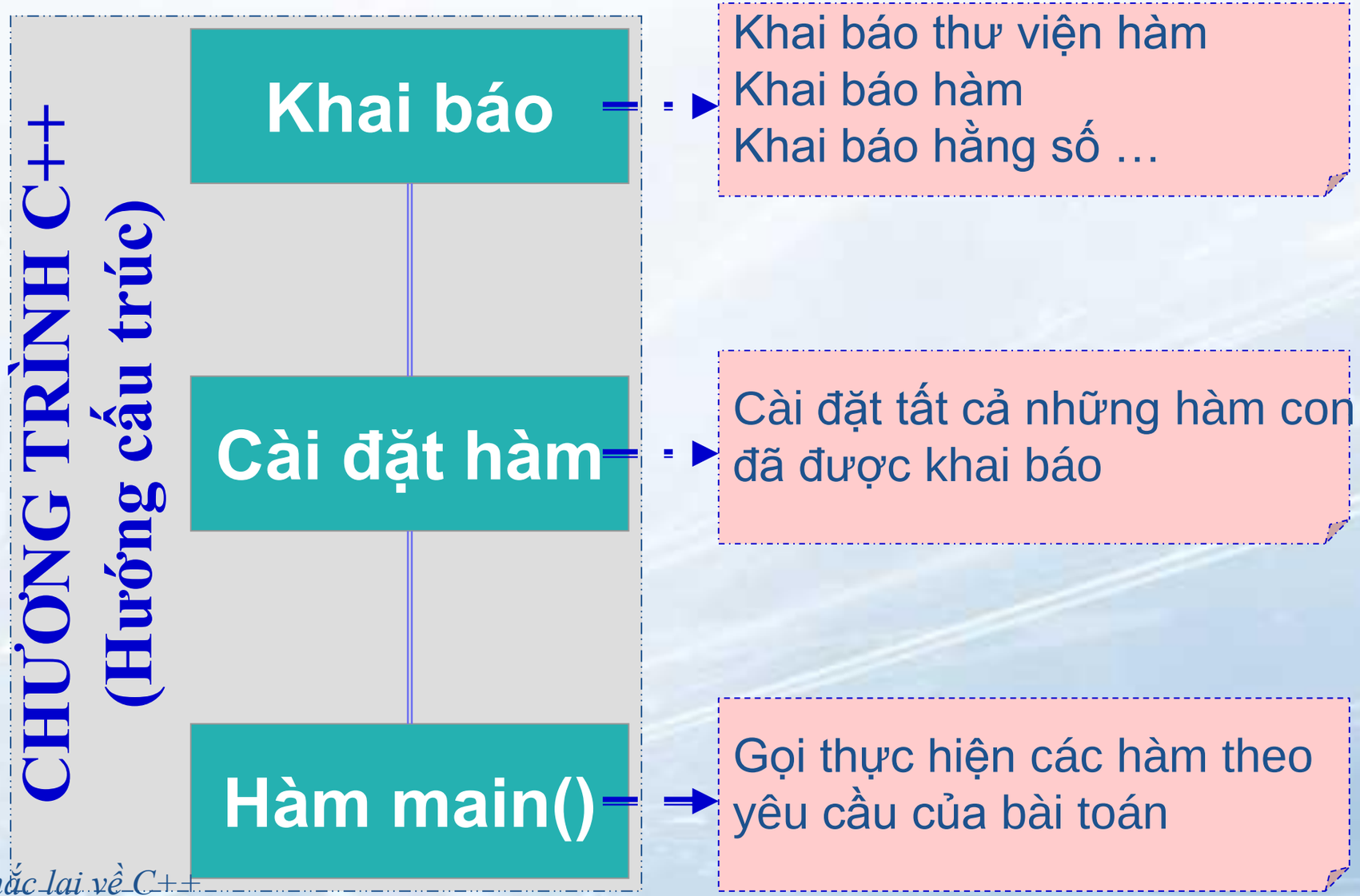
Các môi trường hỗ trợ lập trình (IDE)

- Borland C++
- Microsoft Visual Basic
- Microsoft Visual C++
- JBuider
- Eclipse SDK
- Visual .Net
- …

Các bước thực thi chương trình C++



1.2 Cấu trúc một chương trình C++



1.2 Cấu trúc một chương trình C++

```
/*Chuong trinh C++ */
#include <iostream>

using namespace std;

int main()
{
   //Lenh cout<< de xuat ra man hinh
   cout<<"Xin chao cac ban\n";
   return 0;
}
```

Chú thích
(Comment)

Chỉ thị tiền xử lý
(Preprocessor directive)

Chú thích
(Comment)

Lệnh
(Statement)

Ví dụ khởi động:

Xây dựng chương trình nhập vào 1 số nguyên dương n (yêu cầu nhập lại nếu n âm). Kiểm tra n có là số nguyên tố.

Ví dụ: Nhập số n = 42142

Số 42142 không là số nguyên tố

Chỉ thị tiền xử lý (Preprocessor directive)

- Các chỉ thị tiền xử lý là những dòng được đưa vào trong mã của chương trình phía sau dấu **#**
- Những dòng này không phải là lệnh của chương trình nhưng chỉ thị cho tiền xử lý
- Tiền xử lý kiểm tra mã lệnh trước khi biên dịch thực sự và ***thực hiện tất cả các chỉ thị trước*** khi thực thi mã lệnh của các câu lệnh thông thường

Chỉ thị tiền xử lý (Preprocessor directive)

Đặc điểm:

1. Mô tả trên một dòng, không có dấu ;
2. Trường hợp cần mô tả trên nhiều dòng dùng dấu \ ở cuối mỗi dòng

Khai báo thư viện #include

#include *<header>*
hoặc
#include *"file"*

Mục đích: Chỉ thị này sẽ thay thế toàn bộ nội dung của ***header*** hay một tập tin ***"file"*** *tự định nghĩa thêm.*

Các tập tin này thường chứa định nghĩa các hàm được sử dụng trong chương trình

Hàm main()

- Hàm main() là bắt buộc và được thực hiện đầu tiên khi thực thi chương trình C++
- Tập các lệnh trong hàm main() phải được đặt trong cặp dấu { }
- Chương trình sẽ thực hiện những lệnh theo thứ tự trong hàm main()

Lệnh/ khối lệnh

- **Lệnh**:Lệnh thực hiện một chức năng nào đó (khai báo, gán, xuất, nhập, …) và được kết thúc bằng dấu chấm phẩy (;)
- **Khối lệnh**: Khối lệnh gồm nhiều lệnh và được đặt trong cặp dấu ngoặc { }

Lệnh/ khối lệnh

- Lệnh cout<<“Xin chao cac ban\n”; dùng để xuất ra màn hình dòng chữ “Xin chao cac ban”
- Lệnh return 0; dùng để kết thúc hàm main() – Kết thúc chương trình và trả về giá trị mã là 0

!!! Mỗi lệnh đều được kết thúc bằng dấu ;

Chú thích (comment)

```
/*Chuong trinh C++ */
//Lenh cout<< de xuat ra man hinh
```

- Được lập trình viên ghi chú hay diễn giải trong chương trình
- Đây không phải là lệnh
- Chú thích một dòng: dùng // trước chú thích
- Chú thích cho nhiều dòng: dùng cặp dấu /* và */ để bao nội dung chú thích

1.3 Hàm và nạp chồng hàm

Hàm (chương trình con - subroutine) là một khối lệnh, **thực hiện trọn vẹn một công việc nhất định** (module), được đặt tên và được gọi thực thi nhiều lần tại nhiều vị trí

Khi nào sử dụng hàm?

1. Khi có một công việc giống nhau cần thực hiện ở nhiều vị trí
2. Khi cần chia nhỏ chương trình để dễ quản lý

Hàm

- Hàm có thể được gọi từ chương trình chính (hàm main) hoặc từ 1 hàm khác
- Hàm có giá trị trả về hoặc không
- Nếu hàm không có giá trị trả về gọi là thủ tục *(procedure)*

Hàm

- *Hàm thư viện*: là những hàm đã được xây dựng sẵn. Muốn sử dụng các hàm thư viện phải khai báo thư viện chứa nó trong phần khai báo #include
- *Hàm do người dùng định nghĩa*

Hàm thư viện (C++ định nghĩa sẵn)

- Khai báo thư viện (cmath, stdlib,…)
- Sử dụng các hàm cần thiết

Hàm do người dùng định nghĩa

- Khai báo hàm (nguyên mẫu)
- Định nghĩa hàm
- Gọi hàm

Cấu trúc chung của hàm

KDL trả về của hàm TênHàm([ds tham số]);

KDL trả về của hàm (kết quả của hàm/ đầu ra), gồm:

- **void**: Không trả về giá trị
- **float / int / long / char */ kiểu cấu trúc / …** : Trả về kết quả tính được với KDL tương ứng

Định nghĩa hàm

```
Kiểu dữ liệu trả về của hàm TênHàm([ds tham số])
{
        //Thân hàm
}
```

Cấu trúc chung của hàm

- ***TênHàm*** : Đặt tên theo qui ước sao cho phản ánh đúng chức năng thực hiện của hàm
- ***ds tham số*** (nếu có): đầu vào của hàm

Trong một số trường hợp có thể là đầu vào và đầu ra của hàm nếu kết quả đầu ra có nhiều giá trị

Ví dụ khởi động:

Viết hàm đếm **số chữ số** có trong số nguyên n.

Viết hàm main() nhập vào 1 số nguyên dương n (yêu cầu nhập lại nếu không thỏa mãn). Hiển thị số chữ số có trong n.

Ví dụ: Nhập số n = 42142

Số 42142 vừa nhập có 5 chữ số.

Ví dụ áp dụng

Nạp chồng hàm cho các hàm tính tổng các giá trị tuyệt đối của 2 số nguyên, 2 số thực, 3 số nguyên cùng một chương trình C++.

Viết hàm main() gọi các hàm đã xây dựng.

Nạp chồng hàm

- Đặc điểm
  - ✓ Các hàm cùng tên
  - ✓ Khác nhau về danh sách tham số (khác nhau về **số lượng** tham số hoặc **kiểu dữ liệu** của tham số).
- Nạp chồng hàm tạo ra các định nghĩa hàm riêng biệt
- Cho phép thực hiện cùng một công việc trên các dữ liệu khác nhau

Bài tập áp dụng

Viết các hàm sau:

1. Đếm số ước số của số một số nguyên dương n.
2. Kiểm tra số nguyên n có phải là số nguyên tố không,
3. Kiểm tra xem số nguyên dương n có bao nhiêu chữ số là số nguyên tố,

Hàm **main()** nhập vào một số nguyên dương A gồm k chữ số, gọi các hàm cần thiết để đếm xem A có bao nhiêu chữ số là số nguyên tố.

Ví dụ: Nhập A = 5678

A có 2 chữ số là số nguyên tố

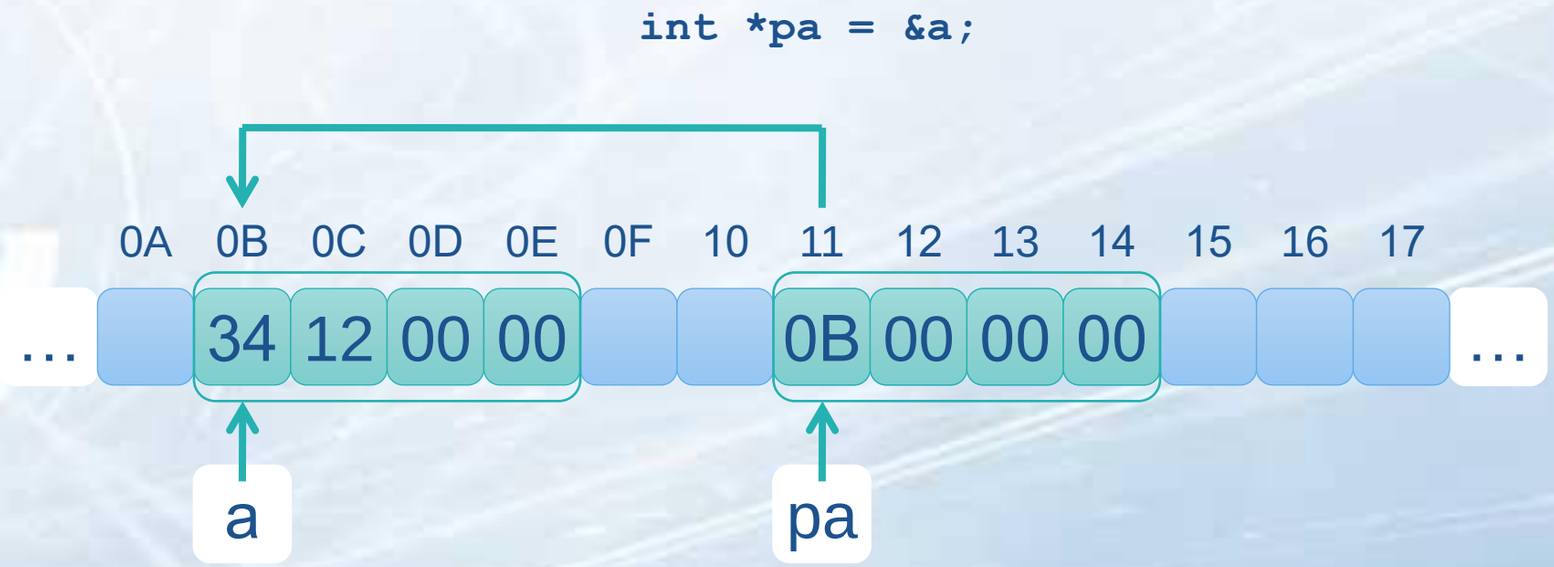
1.4. Nhắc lại về con trỏ

- Khái niệm con trỏ
- Khai báo con trỏ
- Con trỏ NULL
- Khởi tạo con trỏ
- Sử dụng con trỏ
- Kích thước con trỏ

Khái niệm con trỏ

Khái niệm

- Địa chỉ của biến là một con số.
- Ta có thể tạo biến khác để lưu địa chỉ của biến này ➔ Con trỏ.



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 *ch1, *ch2;
int *p1, p2;
```

- ch1 và ch2 là biến con trỏ, trỏ tới vùng nhớ kiểu char (1 byte).
- p1 là biến con trỏ, trỏ tới vùng nhớ kiểu int (4 bytes) còn p2 là biến kiểu int bình thường.

Khai báo con trỏ

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

- Ví dụ

```
double *p;
int *p1;
char *p2;
```

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 varialbe
int *p3 = NULL;
```

Khởi tạo kiểu con trỏ

- Khởi tạo
  - Khi mới khai báo, biến con trỏ được đặt ở địa chỉ nào đó (không biết trước).

    ➔ Đặt địa chỉ của biến vào con trỏ (toán tử &)
- Ví dụ

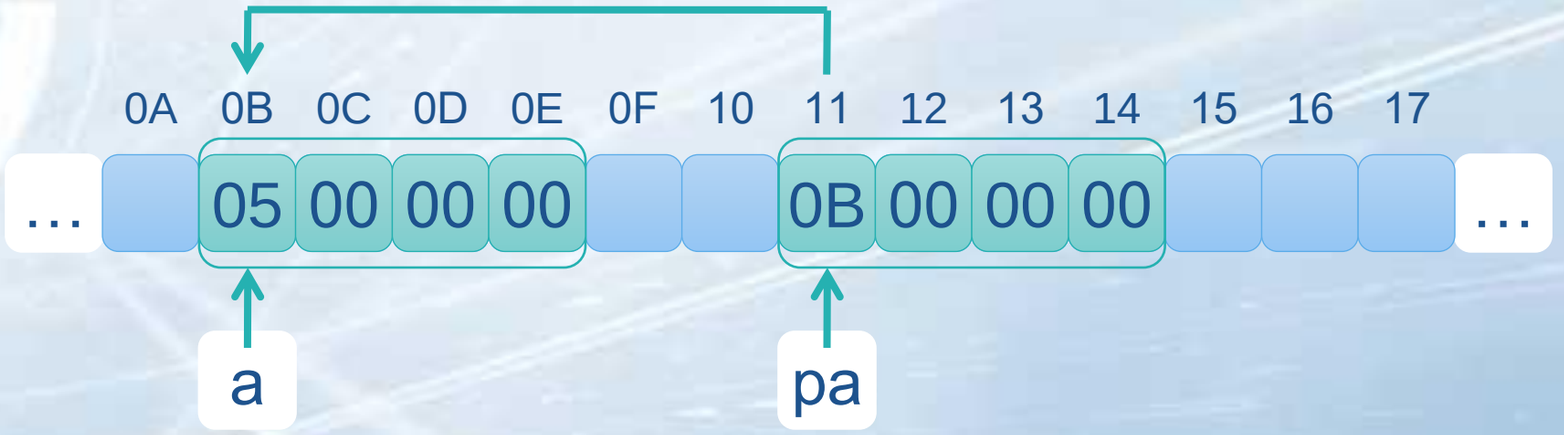
```
<tên biến con trỏ> = &<tên biến>;
```

```
int a, b;
int *pa = &a, *pb;
pb = &b;
```

Sử dụng con trỏ

- Truy xuất đến ô nhớ mà con trỏ trỏ đến
  - Con trỏ chứa một số nguyên chỉ địa chỉ.
  - Vùng nhớ mà nó trỏ đến, sử dụng toán tử *.
- Ví dụ

```
int a = 5, *pa = &a;
cout<< pa;  // Giá trị biến pa
cout<< *pa; // Giá trị vùng nhớ pa trỏ đến
cout<< &pa; // Địa chỉ biến pa
```



Kích thước của con trỏ

- Kích thước của con trỏ

```
char *p1;
int *p2;
float *p3;
double *p4;
…
```

- Con trỏ chỉ lưu địa chỉ nên kích thước của mọi con trỏ là như nhau

- Con trỏ hằng là 1 con trỏ, trỏ đến 1 ô nhớ, nhưng ko được quyền thay đổi giá trị của ô nhớ đó

Con trỏ hằng thường được dùng trong các hàm thao tác với mảng (chẳng hạn) mà hàm này chỉ đọc mảng không được thay đổi các giá trị trong mảng

- Hằng con trỏ là những con trỏ mà chỉ trỏ cố định vào 1 vùng nhớ, những con trỏ này ko có khả năng trỏ vào vùng nhớ khác

1.5. Nhắc lại về kiểu dữ liệu cấu trúc

- Kiểu struct
- Định nghĩa kiểu struct
- Truy nhập tới các trường (thành viên) của biến cấu trúc
- Gán giá trị cho biến cấu trúc

Kiểu struct

- **Ý nghĩa**: là cấu trúc cho phép bên trong chứa các trường dữ liệu mà có kiểu dữ liệu có thể khác nhau
- Các thao tác cơ bản:
  - Định nghĩa kiểu struct
  - Khai báo biến
  - Truy nhập vào các trường
  - Gán giá trị

Kiểu struct – Định nghĩa

```
struct  tên_kiểu {
    Khai báo các trường;
};
```

VD:

```
struct date {
    int  ngay;
    int thang;
    int nam;
//int ngay, thang, nam;
};
```

Kiểu struct – Định nghĩa

```
struct  tên_kiểu {
   Khai báo các trường;
};
```

VD: Định nghĩa một kiểu cấu trúc có tên là Nguoi với các trường tên (Ten), tuổi (Tuoi), giới tính (gioitinh), ngày sinh (date/ngaysinh).

Kiểu struct – Định nghĩa kiểu nguoi

```
struct  tên_kiểu {
   Khai báo các trường;
};
```

VD:

```
struct  nguoi {
   char  ten[30];
   int  tuoi;
   char  gioitinh; // ‘M’ cho nam, ‘F’ cho nữ
   date ngaysinh;
};
```

Truy nhập vào các trường

- Có 2 cách truy nhập:
  - ***Cách 1:*** dùng biến thông thường

    cú pháp "*tên_biến.tên_trường*"
  - ***Cách 2:*** dùng biến con trỏ,

    Cú pháp "*tên_biến->tên_trường*"

```
struct  lophoc{
    char  sohieu[30] ;
    char  chuyennganh;
    int  soluong;
} ;
lophoc  lh;
lophoc * p = & lh;

cin>>lh.sohieu;
cin>>p->chuyennganh;
cin>>p->soluong;
```

```
#include <iostream>
using namespace std;
int main ()
{
int *p1, *p2;
 p1 = new int;
 *p1 = 42;
p2 = p1;
cout << "*p1 = " << *p1 << endl;
cout << "*p2 = " << *p2 << endl;
p1 = new int;
*p1 = 88;
cout << "*p1 = " << *p1 << endl;
cout << "*p2 = " << *p2 << endl;
return 0;
}
```

Kiểu struct - Gán giá trị

- Hai biến cùng một kiểu struct có thể được gán cho nhau. Việc gán này sẽ thực hiện gán lần lượt tất cả các trường của hai biến này cho nhau (tương ứng).

VD:

```
lophoc lh1, lh2;
lh1=lh2;
```

⟺

```
lh1.sohieu = lh2.sohieu;
lh1.chuyennganh =   lh2.chuyennganh;
lh1.soluong = lh2.soluong;
```

Bài tập áp dụng

- Viết chương trình quản lý nhân viên với các chức năng:
  - Nhập thông tin cho n nhân viên (0<n<=5 nhập từ bàn phím các nhân viên không trùng mã)
  - In ra thông tin nhân viên có bậc lương cao nhất trong danh sách.

  Trong đó, các thông tin về nhân viên bao gồm: *mã nhân viên, họ tên, năm sinh, chuyên môn, bậc lương.*