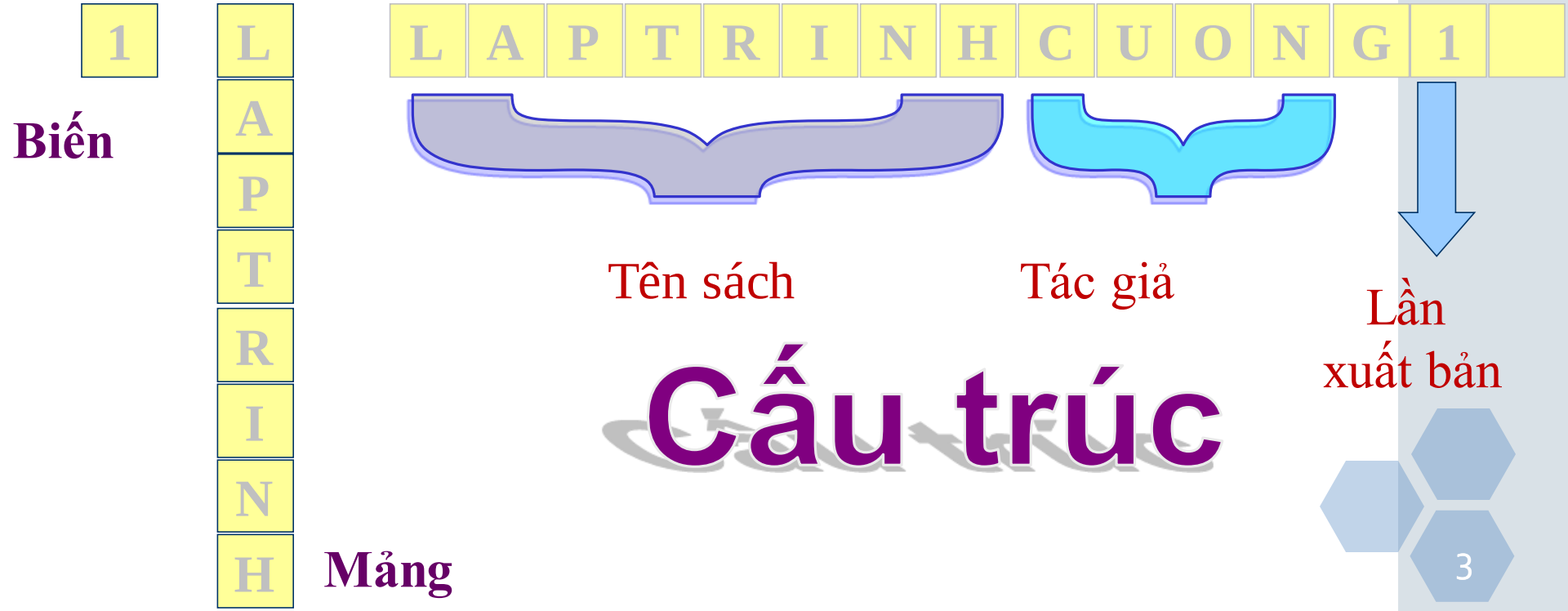
KIỂU CẤU TRÚC
VÀ
VÀO RA FILE

Nội dung

1. Kiểu dữ liệu cấu trúc
2. Khai báo và sử dụng cấu trúc
3. Mảng cấu trúc và con trỏ cấu trúc
4. File trong ngôn ngữ C
5. Các hàm vào ra file cơ bản

Cấu Trúc

- Một cấu trúc bao gồm các mẫu dữ liệu, không nhất thiết cùng kiểu, được nhóm lại với nhau.
- Một cấu trúc có thể bao gồm nhiều mẫu dữ liệu như vậy.



Định Nghĩa Cấu Trúc

- Việc định nghĩa cấu trúc sẽ tạo ra kiểu dữ liệu mới cho phép người dùng sử dụng chúng để khai báo các biến kiểu cấu trúc .
- Các biến trong cấu trúc được gọi là các phần tử của cấu trúc hay thành phần của cấu trúc
- Ví dụ:

```
struct cat {
        char bk_name [25];
        char author [20];
        int edn;
        float price;
};
```

Khai Báo Biến Cấu Trúc

- Khi một cấu trúc đã được định nghĩa, chúng ta có thể khai báo một hoặc nhiều biến kiểu này.
- Ví dụ: **struct cat books1;**
- Câu lệnh này sẽ dành đủ vùng nhớ để lưu trữ tất cả các mục trong một cấu trúc.

```
struct cat {
        char bk_name[25];
        char author[20];
        int edn;
        float price;
} books1, books2;
```

Cách khác

```
struct cat books1, books2;
```

hoặc

```
struct cat books1;
struct cat books2;
```

Truy Cập Phần Tử của Cấu Trúc

- Các phần tử của cấu trúc được truy cập thông qua việc sử dụng **toán tử chấm** (.), toán tử này còn được gọi là **toán tử thành viên - membership**.
- Cú pháp:

  structure_name.element_name
- Ví dụ:

  scanf(“%s”, books1.bk_name);

Khởi Tạo Cấu Trúc

- Giống như các biến khác và mảng, các biến kiểu cấu trúc có thể được khởi tạo tại thời điểm khai báo

```
struct employee
{       int no;
 char name [20];
};
```

- Các biến **emp1** và **emp2** có kiểu **employee** có thể được khai báo và khởi tạo như sau:

```
struct employee emp1 = {346, “Abraham”};
struct employee emp2 = {347, “John”};
```

Câu Lệnh Gán Sử Dụng Các Cấu Trúc - 1

- Có thể sử dụng câu lệnh gán đơn giản để gán giá trị của một biến cấu trúc cho một biến khác có cùng kiểu
- Chẳng hạn, nếu **books1** và **books2** là các biến cấu trúc có cùng kiểu, thì câu lệnh sau là hợp lệ

books2 = books1;

Câu Lệnh Gán Sử Dụng Các Cấu Trúc - 2

- Trong trường hợp không thể dùng câu lệnh gán trực tiếp, thì có thể sử dụng hàm tạo sẵn **memcpy()**
- Cú pháp:

  memcpy (char * destn, char &source, int nbytes);
- Ví dụ:

  memcpy (&books2, &books1, sizeof(struct cat));

Cấu Trúc Lồng Trong Cấu Trúc

- Một cấu trúc có thể lồng trong một cấu trúc khác. Tuy nhiên, một cấu trúc không thể lồng trong chính nó.

```
struct issue      {
    char borrower [20];
    char dt_of_issue[8];
    struct cat books;
}issl;
```

- Việc truy cập vào các phần tử của cấu trúc này tương tự như với cấu trúc bình thường khác,

```
issl.borrower
```

- Để truy cập vào phần tử của cấu trúc cat là một phần của cấu trúc issl ,

```
issl.books.author
```

Truyền tham số kiểu cấu trúc

- Tham số của hàm có thể là một cấu trúc.
- Kiểu của tham số thực sự phải trùng với kiểu của tham số hình thức.
- Ví dụ: …

Mảng Cấu Trúc

- Mảng cấu trúc được khai báo tương tự mảng thông thường
- Một kiểu cấu trúc phải được định nghĩa trước, sau đó một biến mảng có kiểu đó mới được khai báo
- Ví dụ: **struct cat books[50];**
- Để truy cập vào thành phần author của phần tử thứ tư của mảng **books**:

  books[4].author

Khởi Tạo Các Mảng Cấu Trúc

- Mảng cấu trúc được khởi tạo bằng cách liệt kê danh sách các giá trị phần tử của nó trong một cặp dấu móc
- Ví dụ:

```
struct unit {
        char ch;
        int i;
};
struct unit series[3] =
        {{‘a’, 100}{‘b’, 200}{‘c’, 300}};
```

Con Trỏ Đến Cấu Trúc

- Con trỏ cấu trúc được khai báo bằng cách đặt dấu * trước tên của biến cấu trúc.
- Toán tử -> được dùng để truy cập vào các phần tử của một cấu trúc sử dụng một con trỏ
- Ví dụ:

```
struct cat *ptr_bk;
ptr_bk = &books;
printf("%s",ptr_bk->author);
```

- Con trỏ cấu trúc được truyền vào hàm, cho phép hàm thay đổi trực tiếp các phần tử của cấu trúc.

KIỂU FILE

Khái niệm

- File lưu dạng text thông thường(text stream)
- File lưu dạng nhị phân (binary stream)

KIỂU FILE

- Dạng text (text stream)

+ Các chuỗi lưu dạng text

+ Có thể ở xem bình thường.

+ Mỗi lần ghi một chuỗi phải thêm ký tự xuống dòng '\n'

- Dạng nhị phân(binary stream)

+ Các chuỗi lưu dưới dạng được mã hóa binary.

+ Không mở xem dạng thông thường.

+ Mỗi lần ghi tùy thuộc dữ liệu thông thường dùng struct.

THAO TAC TRÊN FILE

1. Mở file (xem có nhiều mode để mở)
2. Thao tác (đọc, ghi)
3. Đóng file.

Mở File Text Stream

Syntax

FILE *fopen(const char *filename, const char *mode);

Ví dụ:

```
FILE *fp;
   fp=fopen("INPUT.TXT","w");// w có nghĩa mở để ghi
```

CÁC MODE MỞ FILE

Mode Meaning

+ "r" Open a text file for reading
+ "w" Create a text file for writing
+ "a" Append to a text file
+ "rb" Open a binary file for reading
+ "wb" Create a binary file for writing
+ "ab" Append to a binary file
+ "r+" Open a text file for read/write
+ "w+" Create a text file for read/write
+ "a+" Open a text file for read/write
+ "rb+" Open a binary file for read/write
+ "wb+" Create a binary file for read/write
+ "ab+" Open a binary file for read/write

Opening Modes in Standard I/O

File Mode	Meaning of Mode	During Inexistence of file
r	Open for reading.	If the file does not exist, fopen() returns NULL.
w	Open for writing.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
a	Open for append. i.e, Data is added to end of file.	If the file does not exists, it will be created.
r+	Open for both reading and writing.	If the file does not exist, fopen() returns NULL.
w+	Open for both reading and writing.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
a+	Open for both reading and appending.	If the file does not exists, it will be created.

GHI FILE DẠNG TEXT STREAM

Syntax:

```
int fputs(const char *str, FILE *fp);
```

Ví dụ:

```
//ghi chuoi n xuong file text
fputs("Nguyen Thi Le",fp);
fputc('\n',fp);// phải thêm xuống dòng
```

LẤY THÔNG TIN FILE DẠNG TEXT STREAM

Syntax:

```
char *fgets(char *str, int length, FILE *fp);
```

Ví dụ:

```
char chuoi_n2[100];
fgets(chuoi_n2,100,fp);
```

Đọc File Binary Stream

Syntax:

size_t fread(void *buffer, size_t numbytes, size_t count, FILE *fp);

Ví dụ:

```
struct NhanVien
{
    char MNV[10];
    char HoTen[MAX];
    char DiaChi[MAX] ;
    char Phai[4];//nhap Nam, Nu
    bool CBQL;
};
NhanVien nv[10];
```

fread(&nv[i], sizeof(nv[i]), 1, f)

Ghi File Binary Stream

Syntax:

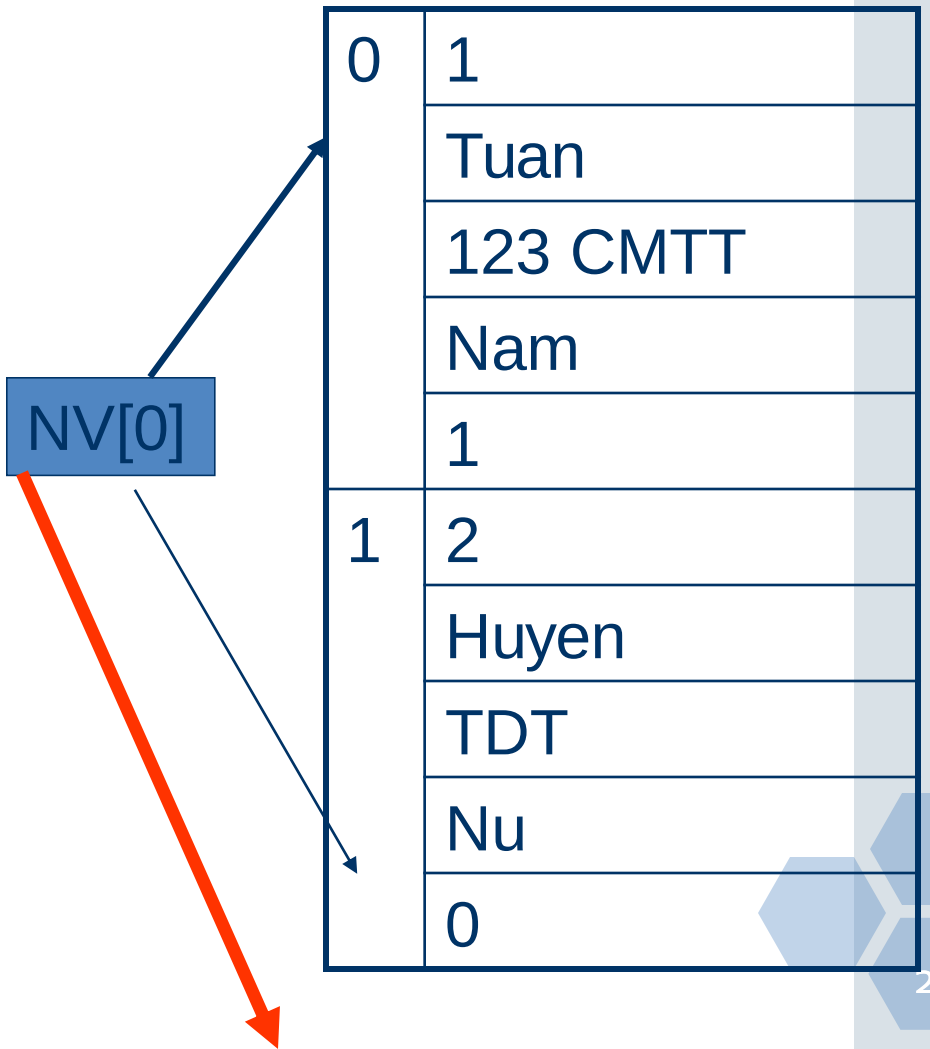
size_t fwrite(const void *buffer, size_t numbytes, size_t count, FILE *fp);

```
struct NhanVien
{
    char MNV[10];
    char HoTen[MAX];
    char DiaChi[MAX] ;
    char Phai[4];//nhap Nam, Nu
    bool CBQL;
};
NhanVien nv[10];
```

fwrite(&nv[i], sizeof(nv[i]), 1, f);

Đọc file

0	MNV
	HoTen
	DiaChi
	Phai
	CBQL
1	MNV
	HoTen
	DiaChi
	Phai
	CBQL



Bài tập áp dụng

Sinh viên chuyển các bài vào ra màn hình – bàn phím sang vào ra với file văn bản