C Programming Basic – week 2

Nội dung

- Cấu trúc
- Cấp phát bộ nhớ động
- Thao tác tệp nhị phân
- Bài tập

Cấp phát bộ nhớ động

- Mạng có kích thước **cố định**, sử dụng để lưu trữ số lượng biến xác định – **biết khi biên dịch**
- Kích thước này không thể thay đổi sau khi biên dịch
- Tuy nhiên, ta thường không biết trước số lượng biến
- Giải quyết dựa trên cấp phát bộ nhớ động

Hàm malloc

```
void * malloc(unsigned int nbytes);
```

- Hàm **malloc** sử dụng để cấp phát **nBytes** trong bộ nhớ
- **malloc** trả về con trỏ tới vùng bộ nhớ nếu thành công, con trỏ **NULL** nếu thất bại
- Cần kiểm tra kết quả trả về của malloc trước khi tiếp tục sử dụng
- **#include <stdlib.h>**

VD

```
int main(void)
{
    int i, n, *p;

    printf("How many numbers do you want to enter?\n");
    scanf("%d", &n);

    /* Allocate an int array of the proper size */
    p = (int *)malloc(n * sizeof(int));
    if (p == NULL)
    {
        printf("Memory allocation failed!\n");
        return 1;
    }
    /* Get the numbers from the user */
    ...
    /* Display them in reverse */
    ...
   /* Free the allocated space */
    free(p);
    return 0;
}
```

VD (2)

```
int main(void)
{
   . . .
    /* Get the numbers from the user */
    printf("Please enter numbers now:\n");
    for (i = 0; i < n; i++)
        scanf("%d", &p[i]);

    /* Display them in reverse */
    printf("The numbers in reverse order are - \n");
    for (i = n - 1; i >= 0; --i)
        printf("%d ",p[i]);
    printf("\n");
    free(p);
    return 0;
}
```

Ép kiểu

Ép kiểu

p = (int *)malloc(n*sizeof(int));

vì **malloc** trả về **void *** :

void * malloc(unsigned int nbytes);

Kiểu (**void ***) là con trỏ tổng quát có thể ép về các kiểu con trỏ với từng kiểu dữ liệu

Giải phóng bộ nhớ

```
void free(void *ptr);
```

- Sử dụng `free(p)` để giải phóng bộ nhớ cấp phát cho `p`
- Nếu `p` không trỏ đến một vùng được cấp phát bởi `malloc` thì trả về lỗi run-time
- Luôn giải phóng bộ nhớ sau khi không sử dụng

Exercise 2.1

- Cài đặt hàm **`my_strcat`** :
  - Input – Hai chuỗi **`s1`** và **`s2`**
  - Output – con trỏ trỏ đến chuỗi ghép nối
  - VD: Ghép nối “hello_” và “world!” tạo thành chuỗi “hello_world!”
- Viết chương trình để kiểm tra hàm

Cấu trúc

- Một tập các biến khác kiểu
- Kết hợp các thông tin liên quan đến một đối tượng
- Các biến trong **struct** được gọi là các biến thành viên hay các trường

Định nghĩa một `struct`

```
struct struct-name
{
      field-type1 field-name1;
      field-type2 field-name2;
      field-type3 field-name3;
   ...
};
```

VD:

```
struct complex {
        int real;
        int img;
  };
struct complex num1, num2,
  num3;
```

Typedef

- Kết hợp `typedef` với định nghĩa cấu trúc

```
typedef struct complex {
        int real;
        int img;
    } complex_t;


complex_t num1, num2;
```

Exercise 2.2

- Cho các cấu trúc

```
typedef struct point
{
    double x;
    double y;
} point_t;


typedef struct circle
{
    point_t center;
    double  radius;
} circle_t;
```

- Viết hàm is_in_circle trả về 1 nếu một điểm p ở trong c. Kiểm tra hàm bằng một chương trình

Con trỏ trong cấu trúc

- Nếu một thành viên của **`struct`** là một con trỏ, khi thực hiện việc sao chép thì chỉ địa chỉ của con trỏ được sao chép

Đọc/ghi tệp nhị phân

mode	Mô tả
"rb"	mở một tệp nhị phân để đọc
"wb"	tạo một tệp nhị phân để ghi
"ab"	mở một tệp nhị phân có sẵn để thêm vào
"r+b"	mở một tệp nhị phân có sãn để đọc/ghi
"w+b"	tạo một tệp nhị phân để đọc/ghi
"a+b"	mở tệp có sẵn hoặc tạo mới để thêm vào

Làm việc với khối dữ liệu

- Hai hàm vào/ra: fread() và fwrite(), được sử dụng để thực hiện thao tác vào ra
- Làm việc với con trỏ tệp

fread()

- Cú pháp hàm fread()

```
size_t fread(void *ptr, size_t size,
  size_t n, FILE *stream);
```

- ptr là con trỏ trỏ đến mảng chứa dữ liệu
- size: kích thước của mỗi phần tử
- n: số lượng phần tử cần đọc
- stream: con trỏ tệp liên kết với tệp cần đọc
- Hàm fread() trả về số phần tử thực sự được đọc

fwrite()

- Cú pháp hàm fwrite()

```
size_t fwrite(const void *ptr, size_t
  size, size_t n, FILE *stream);
```

- ptr là con trỏ trỏ đến mảng chứa dữ liệu
- n: số lượng phần tử được ghi
- stream: con trỏ tệp liên kết với tệp được ghi vào
- Hàm fwrite() trả về số lượng phần tử thực sự được ghi

Hàm feof

- int feof(FILE *stream);

- trả về 0 nếu chưa kết thúc tệp; trả về khác 0 nếu ngược lại

VD

- Đọc 80 byte từ tệp

```
enum {MAX_LEN = 80};
int num;
FILE *fptr2;
char filename2[]= "haiku.txt";
char buff[MAX_LEN + 1];
if ((fptr2 = fopen(filename2, "r")) == NULL){
  printf("Cannot open %s.\n", filename2);
  reval = FAIL; exit(1);
}
. . . .
num = fread(buff, sizeof(char), MAX_LEN, fin);
buff[num * sizeof(char)] = `\0';
printf("%s", buff);
```

Exercise 2.3

- Viết chương trình sao chép tệp lab1.txt sang t lab1a.txt
- Sử dụng các hàm fread, fwrite và feof

Exercise 2.4

- Viết chương trình mycat có chức năng tương tự chương trình cat trong Unix
- Sử dụng hàm fread

Exercise 2.5

- A) Cải thiện Exercise 2.3 sử dụng tham số dòng lệnh

- VD: Nếu chương trình có tên "filecpy", có thể thực hiện lời gọi với cú pháp:
- ./filecpy haiku.txt haiku2.txt

- B) Viết chương trình có chức năng tương tự cat trong linux
- ./cat1 haiku.txt

Gợi ý

- Sử dụng argc[] và argv[]

```
if(argc<3) { printf("%s <file1> <file2>n",argv[0]); exit(1); }
```

- argv[1] và argv[2] là tên của tệp nguồn và tệp đích

```
if((fp=fopen(argv[1],"r"))==NULL) {
…
};
if((fp2=fopen(argv[2],"w"))==NULL) {
…
};
```

Exercise 2.6

- Xây dựng danh bạ điện thoại
- Định nghĩa cấu trúc chứa các trường "name," "telephone number," "e-mail address," và tạo một mảng chứa tối đa 100 bản ghi
- Nhập 10 bản ghi
- Viết chương trình ghi mảng ra tệp bằng hàm fwrite(), sau đó đọc trở lại mảng bằng hàm fread ().

Truy cập tệp ngẫu nhiên

- Hai hàm fseek() và ftell()
- fseek(): hàm di chuyển indicator của tệp tới vị trí mong muốn
- Cú pháp

```
fseek(FILE *stream, long offset, int whence);
```

- Stream là con trỏ tệp liên kết với tệp
- Offset là số byte tính từ vị trí xác định
- Tùy chọn: SEEK_SET, SEEK_CUR, và SEEK_END
  - SEEK_SET: tính từ đầu tệp
  - SEEK_CUR: tính từ vị trí hiện tại
  - SEEK_END: tính từ cuối tệp

Truy cập tệp ngẫu nhiên (2)

- ftell: trả về giá trị hiện thời của indicator
- Cú pháp:

  long ftell(FILE *stream);

- rewind(): khởi tạo lại giá trị của indicator về vị trí đầu tệp
- Cú pháp:

void rewind(FILE *stream);

Exercise 2.7

- Viết chương trình tải một phần danh bạ điện thoại từ tệp (ví dụ từ bản ghi thứ ba tới bản ghi thứ sáu), sửa dữ liệu, sau đó ghi lại vào trong tệp.
- Yêu cầu cấp phát bộ nhớ động sử dụng hàm malloc()

Homework 1

- Truy cập http://thegioididong.vn
- Lấy thông tin về các mẫu điện thoại Nokia và lưu vào tệp văn bản NokiaDB.txt với định dạng

Model Memory DisplaySize Price

- Viết chương trình với các chức năng sau:
  1. Nhập dữ liệu từ tệp vào bộ nhớ; lưu ra tệp nhị phân NokiaDB.dat
  2. Đọc dữ liệu từ tệp nhị phân NokiaDB.dat
  3. In ra danh sách các mẫu điện thoại Nokia trong CSDL
  4. Tìm kiếm trong danh sách theo model
  5. Thoát