
HUST
ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
ONE LOVE. ONE FUTURE.

LẬP TRÌNH C CƠ BẢN

Kiểu dữ liệu cơ bản, vào ra file

ONE LOVE. ONE FUTURE.

Nội dung

- Ôn tập kiểu cấu trúc
- Cấp phát bộ nhớ động
- Thao tác với tập tin nhị phân
- Các bài tập lập trình

Cấp phát bộ nhớ động

Cấp phát bộ nhớ động

- Các mảng thông thường có kích thước cố định được sử dụng để lưu trữ được tối đa một số lượng các phần tử được biết trước tại thời điểm biên dịch.
- Kích thước này không thể thay đổi sau khi chương trình được tạo ra
- Tuy nhiên chúng ta không thể luôn biết trước thực tế sử dụng chương trình sẽ cần lưu trữ làm việc với bao nhiêu phần tử.
- Kỹ thuật cấp phát bộ nhớ động (dynamic memory allocation)
  - Xin hệ thống cấp phát số lượng bộ nhớ theo yêu cầu tại thời điểm chạy chương trình
  - Bộ nhớ cấp phát được quản lý bởi một con trỏ.

Hàm `malloc`

void * malloc(unsigned int nbytes);

- Xin hệ thống cấp phát một vùng (khối) bộ nhớ có kích thước **nBytes**
- **malloc** trả về một con trỏ tới vùng nhớ nếu việc xin cấp phát thành công, trả về con trỏ **NULL** nếu thất bại.
- Lưu ý: Luôn luôn kiểm tra xem bộ nhớ có được cấp phát thành công hay không.
- Thuộc thư viện **stdlib.h: #include <stdlib.h>**

Ví dụ & bài tập: Điền đoạn mã nguồn còn thiếu

```c
#include <stdlib.h>
int main() {
   int i, n, * p;
   printf("How many numbers do you want to enter?\n");
   scanf("%d", &n);
   p = (int*)malloc(n * sizeof(int));
   if (p == NULL) {
      printf("Memory allocation failed!\n");
      return 1;
   }
   /* Nhập các số nguyên */
   ...
      /* Hiển thị chúng theo chiều ngược lại */
      ...
      free(p); /* Giải phóng bộ nhớ */
   return 0;
}
```

Hướng dẫn

```
int main()
{
   . . .
      /* Nhập liệu các số nguyên – phần tử mảng động */
      printf("Please enter numbers now:\n");
   for (i = 0; i < n; i++)
      scanf("%d", &p[i]);

   /* Hiển thị theo chiều ngược */
   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;
}
```

Vì sao cần ép kiểu?

Phép ép kiểu trong câu lệnh

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

là cần thiết vì hàm **malloc** trả về **void *** :

void * malloc(unsigned int nbytes);

Kiểu (**void ***) đặc tả một con trỏ khái quát, có thể được ép kiểu về bất cứ kiểu con trỏ nào.

Hàm calloc

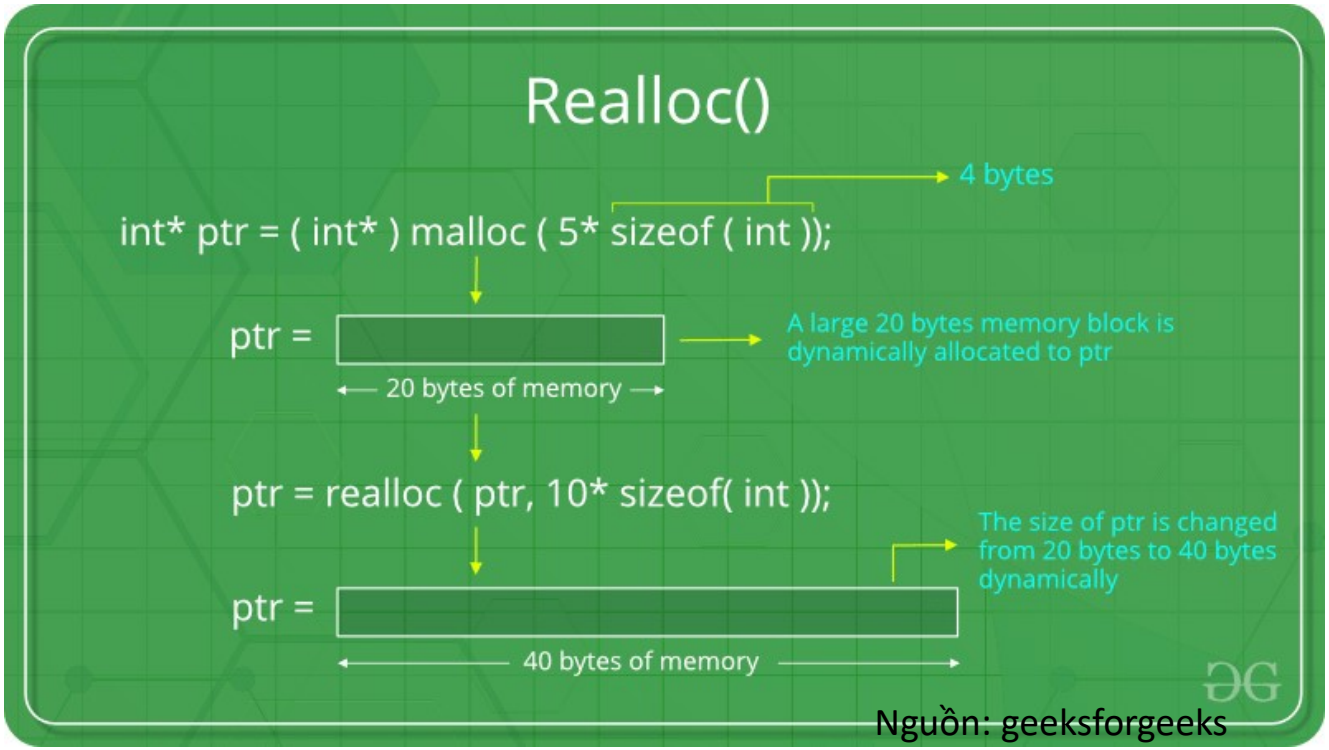
void *calloc(size_t nitems, size_t size);

- Cấp phát động vùng nhớ gồm một số xác định nitems các phần tử cùng kiểu, mỗi phần tử có kích thước size byte.
- Khởi tạo các phần tử với giá trị mặc định là 0, trong khi hàm malloc không tiến hành khởi tạo cho mảng cấp phát.
- Trả về một con trỏ tới vùng nhớ nếu việc xin cấp phát thành công, trả về con trỏ **NULL** nếu thất bại.
- **ptr = (float*) calloc(25,sizeof(float));**

Tái cấp phát bộ nhớ với hàm Reallocate

- Đôi khi chương trình cần cấp phát thêm bộ nhớ sau lần xin cấp phát đầu tiên.
- void *realloc(void *ptr, size_t size)
  - Thay đổi kích thước vùng nhớ trỏ bởi con trỏ **ptr** đã được cấp phát trước đó với hàm **malloc** hoặc **calloc**
- Các tham số
  - **ptr** – là con trỏ tới vùng nhớ đã được cấp phát và cần cấp phát lại. Nếu con trỏ này là NULL, một vùng nhớ mới sẽ được cấp phát và trả về bởi hàm.
  - **size** – Kích thước mới của vùng nhớ tính theo đơn vị byte. Nếu nó là 0 and ptr đang trỏ tới một vùng nhớ đã được cấp phát, vùng nhớ trỏ bởi ptr này được giải phóng và hàm trả về NULL.
- Giá trị trả về
  - Trả về một con trỏ tới vùng nhớ nếu yêu cầu cấp phát lại thành công, trả về con trỏ **NULL** nếu thất bại.

Minh họa hàm realloc



Nguồn: geeksforgeeks

Ví dụ

```c
#include <stdio.h>
#include <stdlib.h>
int main() {
   char* str;
   str = (char*)malloc(15);
   strcpy(str, "tutorialspoint");
   printf("String = %s,  Address = %u\n", str, str);
   /* cấp phát lại bộ nhớ */
   str = (char*)realloc(str, 25);
   strcat(str, ".com");
   printf("String = %s,  Address = %u\n", str, str);
   free(str);
   return(0);
}
```

Giải phóng bộ nhớ cấp phát

void free(void *ptr);

- Hàm **free(p)** giải phóng bộ nhớ cấp phát trỏ bởi **p**
- Nếu **p** không trỏ tới một vùng nhớ nào, một lỗi thực thi chương trình sẽ xuất hiện
- Luôn nhớ giải phóng các vùng nhớ cấp phát động khi không còn cần đến chúng
  - VD: Khi thoát chương trình

Ví dụ

- Ví dụ 1.
- Cài đặt hàm **my_strcat** :
  - Đầu vào là hai xâu ký tự **s1** và **s2**
  - Đầu ra: một con trỏ - trỏ tới vùng nhớ động chứa nội dung là xâu kết quả của phép nối hai xâu s1 và s2
  - Ví dụ: Phép nối “hello_” và “world!” trả về “hello_world!”
- Sử dụng kỹ thuật cấp phát bộ nhớ động
- Kiểm tra lại các hàm đã viết

Đáp án – hàm my_strcat

```
char* my_strcat(char* str1, char* str2) {
    int len1, len2;
    char* result;
    len1 = strlen(str1);
    len2 = strlen(str2);
    result = (char*)malloc((len1 + len2 + 1) * sizeof(char));
    if (result == NULL) {
        printf("Allocation failed! Check memory\n");
        return NULL;
    }
    strcpy(result, str1);
    strcpy(result + len1, str2);
    return result;
}
```

Đáp án: hàm main()

```
int main() {
   char str1[MAX_LEN + 1], str2[MAX_LEN + 1];
   char* cat_str;
   printf("Please enter two strings\n");
   scanf("%100s", str1);
   scanf("%100s", str2);
   cat_str = my_strcat(str1, str2);
   if (cat_str == NULL) {
      printf("Problem allocating memory!n");
      return 1;
   }
   printf("The concatenation of %s and %s is %s\n", str1, str2, cat_str);
   free(cat_str);
   return 0;
}
```

Bài tập

- Bài tập 1. Xây dựng và chạy minh họa (demo) hàm sau
  - char* subStr(char* s1, int offset, int number)
- Hàm tách xâu con từ xâu s1 bắt đầu từ ký tự tại chỉ số **offset** (tính từ 0) và có độ dài **number** ký tự.
- Chú ý kiểm tra tính hợp lệ của các đối số. Trong trường hợp giá trị number lớn hơn độ dài phần còn lại của xâu s1 tính từ vị trí offset, trả về xâu con là phần còn lại của s1 tính từ vị trí offset.

Struct – Cấu trúc

Cấu trúc – Kiểu dữ liệu do người dùng định nghĩa

- Kết hợp nhiều biến trong một thực thể phức hợp – dưới một tên.
- Là một cách thuận tiện để nhóm các thông tin có liên quan (đến chung 1 thực thể trong bài toán) lại với nhau.
- Các biến góp phần tạo ra một **struct** được gọi là các thành phần hay trường.

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

Ví dụ: Kiểu số phức

Ví dụ định nghĩa kiểu số phức

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

Kết hợp với typedef để đặt 1 tên kiểu khác

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


complex_t num1, num2;
```

Ví dụ

Ví dụ. Cho hai cấu trúc sau:

```
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 điểm p nằm trong đường tròn c.
- Kiểm tra hàm bằng một chương trình.

Hướng dẫn

```
int is_in_circle(point_t* p, circle_t* c) {
   double x_dist, y_dist;
   x_dist = p->x - c->center.x;
   y_dist = p->y - c->center.y;
   return (x_dist * x_dist + y_dist * y_dist <= c->radius * c->radius);
}
int main() {
   point_t  p;
   circle_t c;
   printf("Enter point coordinates\n"); scanf("%lf%lf", &p.x, &p.y);
   printf("Enter circle center coordinates\n");
   scanf("%lf%lf", &c.center.x, &c.center.y);
   printf("Enter circle radius\n"); scanf("%lf", &c.radius);
   if (is_in_circle(&p, &c))
      printf("point is in circle\n");
   else
      printf("point is out of circle\n");
   return 0;
}
```

Bài tập

- Bài tập 1. Viết hàm kiểm tra xem hai hình tròn có giao nhau hay không. Viết một chương trình cho phép tạo ra một mảng động các phần tử là các hình tròn, số lượng phần tử do người dùng cung cấp tại thời điểm chạy chương trình.
- Người dùng có thể lựa chọn:
  - Nhập thủ công thông tin cho mỗi hình tròn
  - Tự động sinh thông tin ngẫu nhiên cho mỗi hình tròn.

- Chương trình sử dụng hàm nói trên và hiển thị:
  - Thông tin về các hình tròn
  - Thông tin về hình tròn giao với nhiều hình tròn khác nhất (cùng với thông tin chi tiết, ví dụ giao với các hình tròn nào..)

Bài tập

- Bài tập 2. Cải tiến bài tập tuần trước về chủ đề danh sách sinh viên – bảng điểm như sau:
- Thay vì mảng thông thường, chương trình cấp phát động đúng số bộ nhớ cần thiết để lưu trữ dữ liệu đọc từ file danh sách lớp.
- Bổ sung tính năng nhập thêm: chương trình hỏi số lượng sinh viên cần nhập bổ sung, sau đó tái cấp phát bộ nhớ động để có đủ bộ nhớ cho dữ liệu mới, sử dụng hàm realloc.
- Chú ý: Nhập file danh sách lớp ít nhất 10 dòng, dữ liệu đúng thực tế.

Làm việc với file nhị phân

Các tham số mode cho tập tin nhị phân

mode	Ý nghĩa
"rb"	Mở tập tin nhị phân đã có chỉ để đọc.
"wb"	Mở tập tin nhị phân chỉ để ghi.
"ab"	Mở tập tin nhị phân đã có để ghi thêm vào cuối.
"r+b"	Mở tập tin nhị phân đã có cho phép cả đọc và ghi.
"w+b"	Mở tập tin nhị phân cho phép cả đọc và ghi. Nếu file đã tồn tại, nội dung sẽ bị ghi đè. Nếu file không tồn tại, nó sẽ được tạo tự động.
"a+b"	Mở hoặc tạo tập tin nhị phân cho phép cả đọc và ghi vào cuối.

C cung cấp hai hàm vào ra: fread() và fwrite(), cho phép thực hiện các thao tác vào ra theo các khối các byte. Tương tự như các hàm khác, các hàm trên làm việc với đối số là các con trỏ file.

fread()

- Nguyên mẫu của hàm fread

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

- ptr là con trỏ trỏ tới một mảng được sử dụng để lưu trữ dữ liệu đọc từ tập tin.
- size: kích thước mỗi phần tử mảng (theo byte).
- n: số phần tử dữ liệu đọc từ tập tin.
- stream: con trỏ file gắn với file đang được đọc hay ghi.
- Hàm trả về số phần tử thực sự được đọc thành công từ tập tin.

fwrite()

- Nguyên mẫu hàm fwrite

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

- ptr là con trỏ trỏ tới một mảng được sử dụng để lưu trữ dữ liệu sẽ được ghi ra tập tin.
- size: kích thước mỗi phần tử mảng (theo byte).
- n: số phần tử dữ liệu đọc từ tập tin.
- stream: con trỏ file gắn với file đang được đọc hay ghi.
- Hàm trả về số phần tử thực sự được ghi thành công vào tập tin.

Hàm feof

- int feof(FILE *stream);
- Kiểm tra xem vị trí con trỏ file đã tới cuối tập tin hay chưa.
  - Hàm trả về 0 nếu chưa tới vị trí cuối file; ngược lại trả về giá trị khác không.

Ví dụ

- Ví dụ 1. Đọc 80 byte từ tập tin haiku.txt.

```
#define 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, fptr2);
buff[num * sizeof(char)] = '\0';
printf("%s", buff);
```

Ví dụ

- Ví dụ 2. Viết chương trình sao chép tập tin (từ lab1.txt sang lab1a.txt) tương tự các bài tập trước nhưng sử dụng các thao tác đọc ghi tập tin theo khối dữ liệu.
- Sử dụng các hàm: fread, fwrite, feof

```
#include <stdio.h>
enum { SUCCESS, FAIL };
#define MAX_LEN  80

void BlockReadWrite(FILE* fin, FILE* fout) {
   int num;
   char buff[MAX_LEN + 1];

   while (!feof(fin)) {
      num = fread(buff, sizeof(char), MAX_LEN, fin);
      buff[num * sizeof(char)] = `\0';

         printf("%s", buff);
      fwrite(buff, sizeof(char), num, fout);
   }
}
```

Đáp án

```
int main() {
   FILE* fptr1, * fptr2;
   char filename1[] = "lab1a.txt";
   char filename2[] = "lab1.txt";
   int reval = SUCCESS;
   if ((fptr1 = fopen(filename1, "w")) == NULL) {
      printf("Cannot open %s.\n", filename1);
      reval = FAIL;
   }
   else if ((fptr2 = fopen(filename2, "r")) == NULL) {
      printf("Cannot open %s.\n", filename2);
      reval = FAIL;
   }
   else {
      BlocReadWrite(fptr2, fptr1);
      fclose(fptr1);
      fclose(fptr2);
   }
   return reval;
}
```

Ví dụ

- Ví dụ 2. Viết chương trình mycat có chức năng tương tự lệnh cat trong Unix sử dụng kỹ thuật vào ra theo khối dữ liệu.
- Gợi ý:
  - Nhận đối số dòng lệnh
  - Dùng hàm fread

```
void BlockCat(FILE* fin) {
   int num;
   char buff[MAX_LEN + 1];

   while (!feof(fin)) {
      num = fread(buff, sizeof(char), MAX_LEN, fin);
      buff[num * sizeof(char)] = `\0';

         printf("%s", buff);
   }
}
```

Ví dụ

```
main(int argc, char* argv[]) {
   FILE* fptr1, * fptr2;
   int reval = SUCCESS;
   if (argc != 2) {
      printf("The correct syntax should be: cat1 filename \n");
      reval = FAIL;
   }
   if ((fptr1 = fopen(argv[1], "r")) == NULL) {
      printf("Cannot open %s.\n", argv[1]);
      reval = FAIL;
   }
   else {
      BlockCat(fptr1);
      fclose(fptr1);
   }
   return reval;
}
```

Bài tập

- Bài tập 1. Viết chương trình sao chép tập tin theo nhiều chế độ, hoạt động với giao diện menu dòng lệnh với các chức năng chính như sau:
  1. Copy by character
  2. Copy by line
  3. Copy by block - optional size
  4. Quit
- Ở mỗi chức năng sao chép, sau khi hoàn thành việc sao chép, hiển thị thời gian thực hiện theo đơn vị mili giây để so sánh.
  Chú ý: Tập tin nguồn phải là tập tin văn bản có kích thước tối thiểu là 640KB.

Bài tập

- Bài tập 2. Đọc và ghi tập tin nhị phân chứa các cấu trúc
  Giả sử bạn cần quản lý một danh bạ điện thoại của mình bằng chương trình. Định nghĩa một cấu trúc biểu diễn danh bạ gồm các trường "name," "telephone number," "e-mail address," và khai báo một mảng chứa tối đa 100 phần tử thuộc kiểu cấu trúc trên.
- Nhập liệu cho khoảng 10 phần tử mảng.
- Chương trình sau đó ghi nội dung mảng với các phần tử nói trên vào tập tin có tên phonebook.dat sử dụng hàm fwrite.
- Đọc lại dữ liệu từ tập tin vào mảng sử dụng hàm fread và in nội dung mảng ra màn hình để kiểm tra.

Hướng dẫn

```
enum { SUCCESS, FAIL };
#define MAX_ELEMENT  20

// the phone book structure
typedef struct phoneaddress_t {
   char name[20];
   char tel[11];
   char email[25];
}phoneaddress;

int main(void)
{
   FILE* fp;
   phoneaddress phonearr[MAX_ELEMENT];
   int i, n, irc; // return code
   int reval = SUCCESS;
```

Hướng dẫn

```
printf("How many contacts do you want to enter (<20)?");
scanf("%d", &n);
for (i = 0; i < n; i++) {
   printf("name:"); scanf("%s", phonearr[i].name);
   printf("tel:"); scanf("%s", phonearr[i].tel);
   printf("email:"); scanf("%s", phonearr[i].email);
}
if ((fp = fopen("phonebook.dat", "w+b")) == NULL) {
   printf("Can not open %s.\n", "phonebook.dat");
   reval = FAIL;
}
// write the entire array into the file
irc = fwrite(phonearr, sizeof(phoneaddress), n, fp);
printf(" fwrite return code = %d\n", irc);
fclose(fp);
```

Hướng dẫn (tiếp)

```
    //read from this file to array again
    if ((fp = fopen("phonebook.dat", "rb")) == NULL) {
        printf("Can not open %s.\n", "phonebook.dat");
        reval = FAIL;
    }
    irc = fread(phonearr, sizeof(phoneaddress), n, fp);
    printf(" fread return code = %d\n", irc);
    for (i = 0; i < n; i++) {
        printf("%s-", phonearr[i].name);
        printf("%s-", phonearr[i].tel);
        printf("%s\n", phonearr[i].email);
    }
    fclose(fp);
    return reval;
}
```

Bài tập

- Bài tập 3. Viết chương trình đọc tập tin bảng điểm (bangdiem.txt kết quả của bài tập về danh sách sinh viên và bảng điểm đã cho) và lưu trữ dữ liệu sử dụng bộ nhớ động, sau đó ghi chúng ra tập tin nhị phân grade.dat (chứa mảng các phần tử kiểu cấu trúc về sinh viên).
- Chương trình có thể: đọc file grade.dat và hiển thị bảng điểm trên màn hình, tìm kiếm một sinh viên dựa trên mã số sinh viên và cập nhật điểm mới nhập từ người dùng và lưu vào tập tin.
- Chương trình nên được viết với tương tác menu dòng lệnh:

  1. TextToDat

  2. Display Dat file

  3. Search and Update.

  4 Quit.

Di chuyển trong tập tin

Truy cập ngẫu nhiên tập tin

- Sử dụng hai hàm: fseek() and ftell()
- fseek(): hàm chuyển dịch vị trí con trỏ file tới một điểm mong muốn trong tập tin.

  fseek(FILE *stream, long offset, int whence);
- Stream – con trỏ tập tin
- Offset : độ dài chuyển dịch tính theo byte.
- Whence: hằng số chỉ mốc và hướng dịch chuyển
  - SEEK_SET: từ đầu tập tin, dịch chuyển về phía cuối tập tin
  - SEEK_CUR: từ vị trí con trỏ file hiện tại, dịch chuyển về phía cuối tập tin
  - SEEK_END: từ cuối tập tin dịch chuyển về đầu tập tin.

Truy cập ngẫu nhiên tập tin

- Hàm ftell: cho biết giá trị vị trí hiện tại của con trỏ file
- Cú pháp: long ftell(FILE *stream);

- Hàm rewind(): Đặt lại vị trí con trỏ file ở đầu tập tin.
- Cú pháp: void rewind(FILE *stream);

Ví dụ: Đọc/Trích xuất một phần dữ liệu từ tập tin nhị phân

- Ví dụ 1. Viết chương trình đọc một phần cụ thể trong dữ liệu danh bạ lưu trữ trong tập tin phonebook.dat. Ví dụ từ dữ liệu danh bạ (bản ghi) thứ 2 đến thứ 3, hay từ thứ 3 đến thứ 6. Sau đó thay đổi giá trị trường email và ghi lại vào tập tin ở đúng vị trí đã trích xuất.
- Chương trình cần cấp phát bộ nhớ để lưu trữ đúng lượng dữ liệu đọc ra từ tập tin khi chạy chương trình. Ví dụ cần mảng động với 4 phần tử cấu trúc để lưu dữ liệu từ phần tử thứ 3 đến thứ 6.

Đáp án

```c
#include <stdio.h>
#include <stdlib.h>
enum { SUCCESS, FAIL };
#define MAX_ELEMENT  20
// the phone book structure
typedef struct phoneaddress {
   char name[20];
   char tel[11];
   char email[25];
}phoneaddress;
int main(void) {
   FILE* fp;
   phoneaddress* phonearr;

   int i, n, irc; // return code
   int reval = SUCCESS;
   printf("Read from 2sd data to 3rd data \n");
```

Đáp án

```
if ((fp = fopen("phonebook.dat", "r+b")) == NULL) {
    printf("Can not open %s.\n", "phonebook.dat");
    reval = FAIL;
}
// Memory allocation
phonearr =
    (phoneaddress*)malloc(2 * sizeof(phoneaddress));
if (phonearr == NULL) {
    printf("Memory allocation failed!\n");
    return FAIL;
}
if (fseek(fp, 1 * sizeof(phoneaddress), SEEK_SET) != 0)
{
    printf("Fseek failed!\n");
    return FAIL;
}
irc = fread(phonearr, sizeof(phoneaddress), 2, fp);
```

Đáp án

```
    for (i = 0; i < 2; i++) {
        printf("%s-", phonearr[i].name);
        printf("%s-", phonearr[i].tel);
        printf("%s\n", phonearr[i].email);
    }
    // Modifying some data
    strcpy(phonearr[1].name, "Lan Hoa");
    strcpy(phonearr[1].tel, "0923456");
    strcpy(phonearr[1].email, "lovelybuffalo@hut.edu.vn");
    fseek(fp, 1 * sizeof(phoneaddress), SEEK_SET);
    irc = fwrite(phonearr, sizeof(phoneaddress), 2, fp);
    printf(" fwrite return code = %d\n", irc);
    fclose(fp); free(phonearr);
    return reval;
}
```

Bài tập

- Bài tập 1. Viết chương trình chuyển đổi dữ liệu từ điển Việt Anh từ định dạng văn bản sang nhị phân.
- Dữ liệu có tại
- http://www.denisowski.org/Vietnamese/vnedict.txt
  - Miên Đất Hứa : the Promised Land
  - Miến : Burma (short for Miến Điện)
  - Miến Điện : Burma
  - Miền Trung : Central Vietnam
- Sau đó chương trình đọc dữ liệu từ tập tin nhị phân, hỏi người dùng vị trí bắt đầu và kết thúc của mục từ và hiển thị các từ nằm ở các vị trí này trong từ điển.

Bài tập : Quản lý thông tin mô tả điện thoại di động

- Bài tập 2. Quản lý thông tin mô tả điện thoại di động
  Từ các trang web của các showroom điện thoại di động, xây dựng một tập tin văn bản có tên PhoneDB.txt chứa thông tin về ít nhất 20 mẫu điện thoại gần đây như iPhone, Samsung, Oppo, Huawei, mỗi điện thoại một dòng.. theo định dạng sau
  - Model Memory Space (GB) Screen Size (inches) Price
- Viết chương trình có giao diện menu như sau:
  1. Import DB from text: Đọc file PhoneDB.txt và chuyển thành định dạng nhị phân PhoneDB.dat sử dụng kỹ thuật cấp phát động bộ nhớ.
  2. Import from DB: Đọc dữ liệu từ file PhoneDB.dat và nạp vào bộ nhớ chương trình. Cho phép người dùng lựa chọn hai chế độ đọc: Đọc toàn bộ và đọc một phần (chỉ định vị trí bản ghi bắt đầu và kết thúc).
  3. Print All Database: Hiển thị dữ liệu về các mẫu điện thoại trên màn hình, mỗi mẫu một dòng và căn thẳng các cột.
  4. Search by phone by Phone Model: Tìm kiếm điện thoại dựa trên model do người dùng nhập.
  5. Exit

Bài tập : Chia tách và ghép File

- Bài tập 3. Chia tách và ghép File
  Sử dụng tập tin phonebook.dat (kết quả của bài tập trên Lab) chứa ít nhất 20 số liên lạc. Viết các chương trình sau
- Chương trình **filesplit** nhận hai đối số: tên file nguồn (.dat) và một số nguyên N và tên hai file kết quả. Nó sẽ tách file nguồn thành 2 file, trong đó file đầu tiên chứa N số liên lạc đầu tiên và file thứ hai chứa các số liên lạc còn lại. Ví du
  - **filesplit phone.dat 10 phone1.dat phone2.dat**
- Chương trình **filemerge** ghép hai file đã tách thành một file:
  - **filemerge phone1.dat phone2.dat phone.dat**
- Chương trình **fileread** đọc và hiển thị danh sách các số liên lạc chứa trong một file .dat bất kỳ ra màn hình. Nó được sử dụng để kiểm tra kết quả thực hiện các chương trình filesplit and filemerge.