ĐẠI HỌC CÔNG NGHIỆP TP.HCM

KHOA CÔNG NGHỆ THÔNG TIN

THỰC HÀNH KỸ THUẬT LẬP TRÌNH

Biên Soạn:

ThS. Võ Quang Hoàng Khang

HƯỚNG DẪN

MÔ TẢ MÔN HỌC

Môn Thực hành Kỹ thuật Lập trình cung cấp cho sinh viên những kỹ năng thực hành cơ bản về ngôn ngữ lập trình C. Môn học này là nền tảng để tiếp thu hầu hết các môn học khác trong chương trình đào tạo. Mặt khác, nắm vững môn này là cơ sở để phát triển tư duy và kỹ năng lập trình để giải các bài toán và các ứng dụng trong thưc tế.

Học xong môn này, sinh viên phải nắm được các vấn đề sau:

- Thao tác trên kiểu dữ liệu có cấu trúc
- Thao tác trên kiểu dữ liệu con trỏ với mảng 1 chiều, 2 chiều và mảng cấu trúc.
- Kỹ thuật viết hàm đệ quy.
- Thao tác trên tập tin.
- Kỹ thuật sắp xếp, tìm kiếm cơ bản.

NỘI DUNG MÔN HỌC

- Bài 1. KIỂU CON TRỎ
- Bài 2. KIỂU CHUỖI
- Bài 3. KIỂU DỮ LIỆU CẤU TRÚC
- Bài 4. KỸ THUẬT LẬP TRÌNH ĐỆ QUY
- Bài 5. TẬP TIN (FILE)

CÁCH TIẾP NHẬN NỘI DUNG MÔN HỌC

Thực hành Kỹ thuật lập trình là môn học giúp sinh viên làm quen với phương pháp lập trình trên máy tính, giúp sinh viên có khái niệm cơ bản về cách tiếp cận và giải quyết các bài toán tin học, giúp sinh viên có khả năng tiếp cận với cách tư duy của người lập trình, và là tiền đề để tiếp cận với các học phần quan trọng tiếp theo của ngành Công nghệ Thông tin. Vì vậy, yêu cầu người học phải dự học đầy đủ các buổi thực hành tại

lớp, thực hành lại các bài tập ở nhà, nghiên cứu tài liệu trước khi đến lớp và gạch chân những vấn đề không hiểu khi đọc tài liệu để đến lớp trao đổi.

BÀI 1: KIỂU CON TRỎ

Sau khi thực hành xong bài này, sinh viên có thể nắm được:

- *Biết cách khai báo và sử dụng biến kiểu con trỏ;*
- *Biết xử lý các thao tác trên mảng theo kiểu con trỏ;*
- *Biết xử lý các thao tác trên chuỗi theo kiểu con trỏ;*
- *Đi sâu vào các giải thuật trên mảng như tìm kiếm, sắp xếp, thêm phần tử, xóa phần tử...theo kiểu con trỏ;*

1.1 TÓM TẮT LÝ THUYẾT

1.1.1 Khai báo biến con trỏ

Cú pháp: <Kiểu> *<Tên con trỏ>

Ý nghĩa: Khai báo một biến có tên là ***Tên con trỏ*** dùng để chứa địa chỉ của các biến có kiểu ***Kiểu.***

Ví dụ 1: Khai báo 2 biến a,b có kiểu int và 2 biến pa, pb là 2 biến con trỏ int

```
a, b, *pa, *pb;
```

Ví dụ 2: Khai báo biến f kiểu float và biến pf là con trỏ float

```
float  f, *pf;
```

1.1.2 Các thao tác trên con trỏ

Gán địa chỉ của biến cho biến con trỏ

Toán tử **&** dùng để định vị con trỏ đến địa chỉ của một biến đang làm việc.

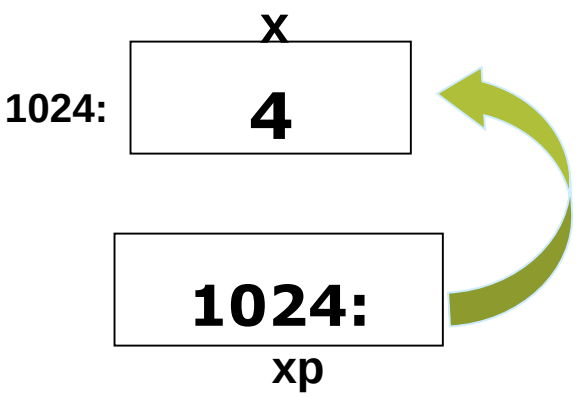
Cú pháp: **<Tên biến con trỏ> = &<Tên biến>**

Giải thích: Ta gán địa chỉ của biến ***Tên biến*** cho con trỏ ***Tên biến con trỏ***.

Ví dụ: khai báo biến int *xp, x = 4 ;

Gán địa chỉ của biến x cho con trỏ xp. xp = &x

Lúc này, hình ảnh của các biến trong bộ nhớ được mô tả:



Lấy giá trị của biến con trỏ chỉ tới

Để truy cập đến nội dung của ô nhớ mà con trỏ chỉ tới, ta sử dụng cú pháp:

***<Tên biến con trỏ>**

Với cách truy cập này thì ***<Tên biến con trỏ>** có thể coi là một biến có kiểu được mô tả trong phần khai báo biến con trỏ.

Ví dụ sau đây cho phép khai báo, gán địa chỉ cũng như lấy nội dung vùng nhớ của biến con trỏ: int x=100; *ptr; ptr= &x; int y= *ptr;

Lưu ý: Khi gán địa chỉ của một biến cho một biến con trỏ, mọi sự thay đổi trên nội dung ô nhớ con trỏ chỉ tới sẽ làm giá trị của biến thay đổi theo (thực chất nội dung ô nhớ và biến chỉ là một).

1.1.3 Con trỏ và mảng 1 chiều

Truy cập từng phần tử đang được quản lý bởi con trỏ theo dạng mảng:

<Tên biến>[<Vị trí>] *tương đương với* ***(<Tên biến> + <Vị trí>)**

&<Tên biến>[<Vị trí>] *tương đương với* **(<Tên biến> + <Vị trí>)**

Trong đó:

<Tên biến> là biến con trỏ,

<Vị trí> là 1 biểu thức số nguyên.

1.2 MỘT SỐ ỨNG DỤNG CHÍNH CỦA CON TRỎ

- Passing Arguments by Reference

```c
void swap(int *x, int *y) {
    int temp = *x;
    *x = *y;
    *y = temp;
}
```

- Accessing Array Elements

```c
int arr[] = {1, 2, 3, 4, 5};
int *ptr = arr;  // Pointer to the first element

printf("Accessing elements using pointer:\n");
for (int i = 0; i < 5; i++) {
    printf("%d ", *(ptr + i));  // Dereference and access element
}
```

- Dynamic Memory Allocation

```c
int main() {
    int *ptr = (int *)malloc(5 * sizeof(int));  // Allocate memory for 5 integers
    if (ptr == NULL) {
        printf("Memory allocation failed!\n");
        return 1;
    }

    // Use the allocated memory
    for (int i = 0; i < 5; i++) {
        ptr[i] = i * 10;
    }

    // Free the memory when done
    free(ptr);
    return 0;
}
```

- Implementing Data Structures (Linked lists, Trees, Queues, Stacks, Graphs)

```c
struct Node {
    int data;
    struct Node *next;
};
```

- Handling Strings

```c
char str1[] = "Hello";
char str2[] = "World";
char *ptr = strcat(str1, " ");  // Concatenate strings
ptr = strcat(ptr, str2);
printf("Concatenated string: %s\n", str1);
```

- Returning Multiple Values from Functions

```c
void getMinMax(int arr[], int n, int *min, int *max) {
    *min = arr[0];
    *max = arr[0];
    for (int i = 1; i < n; i++) {
        if (arr[i] < *min) {
            *min = arr[i];
        } else if (arr[i] > *max) {
            *max = arr[i];
        }
    }
}
```

- Function Pointers
  - Purpose: Pointers can point to functions, enabling dynamic function calls and techniques.
  - Techniques: Callback functions, function tables.

1.3 THỰC HÀNH CƠ BẢN

Câu 1. Chương trình thực hiện các xử lý cơ bản trên con trỏ.

```c
#include<stdio.h>
int main(){
      int* a;
      int z;
      z = 10;
      printf("How works pointer and how to handle pointers:\n\n");
      printf("Value of z = %d\n",z);
      printf("Address of z = %p\n",&z);
      a = &z;
      //Now, a is assigned with the address of z.
```

```
    printf("\nNow a is assigned with the address of z:\n");
    printf("Address of pointer a = %p\n",a);
    printf("content of pointer a = %d\n\n",*a);
    z = 30;//now value of z is 30
   printf("Now, The value of z assigned to 30:\n");
   printf("Address of pointer a = %p\n",a);
   printf(" Content of pointer a = %d\n\n",*a);
   *a = 5;
   printf("Now, pointer variable a is assigned the value 5\n");
   printf("Address of z = %p\n",&z);
   /*here, a contain the address of z, so *a changed the value of z (now z=7)*/
   printf("Value of z = %d",z);
   return 0;
}
```

Câu 2. Dùng con trỏ, viết chương trình để thực hiện các phép tính số học.

Câu 3. Dùng con trỏ, viết chương trình để tìm giá trị lớn nhất giữa hai số.

Câu 4. Dùng con trỏ, viết hàm hoán đổi hai số (swap).

Câu 5. Viết chương trình thực hiện các chức năng sau:

a) Nhập mảng số nguyên.

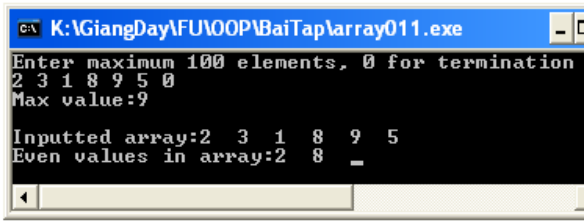
b) Xuất ra mảng số nguyên vừa nhập.

c) Xuất các số chẵn có trong mảng.

d) Tìm giá trị lớn nhất trong mảng.

Hướng dẫn:

```c
#include <stdio.h>
#define MAXN 100
/* Input an array, number of elements is stored at pn
   User will terminate inputting when 0 is entered.*/
void input(int*a, int *pn);
int max(int a[], int n);
void print (int* a, int n);
void printEven (int* a, int n);
int main()
{  int a[MAXN]; /* static array of 100 integers */
   int n; /* real used number of elements */
   int maxVal;
   input(a,&n);
   maxVal  = max (a,n);
   printf("Max value:%d\n", maxVal);
   printf("\nInputted array:");
   print(a,n);
   printf("\nEven values in array:");
   printEven(a,n);
   while (getchar()!='\n');getchar();
   return 0;
}
```




```c
void input(int*a, int *pn)
{ *pn=0; /* reset the number of elements */
  printf ("Enter maximum %d elements, 0 for termination\n", MAXN);
  int x; /* inputted value */
  do
  { scanf("%d", &x);
    if (x!=0) a[(*pn)++] = x;
  }
  while (x!=0 && *pn < MAXN);
}
```

```c
int max(int a[], int n)
{
    /* Do yourself */
}
void print (int* a, int n)
{  /* Do yourself */
}
void printEven (int* a, int n)
{  /* Do yourself */
}
```

Câu 6. Viết chương trình sử dụng vùng bộ nhớ cấp phát động (HEAP SEGMENT) cho phép người dùng nhập 2 số thực. Tính tổng, hiệu, tích, thương của chúng.

Câu 7. Viết chương trình sử dụng bộ nhớ cấp phát động cho phép người dùng nhập 2 ký tự, sau đó chương trình sẽ in ra các ký tự giữa 2 ký tự này theo thứ tự tăng dần.

Example:
Input: DA
Output:
A 65 81 41
B 66 82 42
C 67 83 43
D 68 84 44

Hướng dẫn:

- `char* pc1, *pc2;`
- Cấp bộ nhớ pc1, pc2;
- Nhập 2 ký tự vào pc1, pc2
- Nếu (*pc1>*pc2)
  Hoán vị *pc1, *pc2;
- `char c;`
- `For (c= *pc1; c<=*pc2; c++)`
  `printf("%c, %4d, %4o, %4X\n", c, c, c, c);`

Câu 8. Viết chương trình thực hiện các thao tác trên mảng 1 chiều, sử sụng kỹ thuật cấp phát động (HEAP).

a) Nhập mảng số thực.
b) Xuất các phần tử của mảng.
c) Tính giá trị trung bình của mảng: $m = (x_0 + x_1 + x_2 + ... + x_{n-1}) / n$
d) Tính tổng bình phương: $ss = x_0^2 + x_1^2 + x_2^2 + ... + x_{n-1}^2$
e) Tính phương sai (Variance): $d^2 = (ss / n) - m^2$
f) Tính độ lệch chuẩn (Standard deviation): $d = sqrt((ss / n) - m^2)$.
g) Thêm giá trị X vào vị trí P (X, P nhập từ bàn phím)
h) Xóa phần tử tại ví trí P (P nhập từ bàn phím)
i) Tìm kiếm giá trị X trong mảng (X nhập từ bàn phím)
j) Sắp xếp mảng tăng/giảm

BÀI 2: CHUỖI KÝ TỰ

Chuỗi ký tự là trường hợp đặc biệt của mảng một chiều. Chuỗi ký tự là một dãy các phần tử, mỗi phần tử có kiểu ký tự.

Chuỗi ký tự được kết thúc bằng ký tự '\0'. Do đó khi khai báo độ dài của chuỗi luôn luôn khai báo dư 1 phần tử để chứa ký tự '\0'.

2.1 TÓM TẮT LÝ THUYẾT

- **Nhập chuỗi:**

Cách 1: **gets(biến_chuỗi);**

Nhận các ký tự nhập từ phím cho đến khi nhấn phím Enter và đưa vào biến chuỗi.

Ví dụ:

```
char s[30];  fflush(stdin); printf("Nhap chuoi: ");
gets(s);
```

Cách 2: **scanf("%s", biến_chuỗi);**

Nhận các ký tự nhập từ phím cho đến khi gặp phím **space**, tab, new line, Enter thì dừng, cho nên chỉ dùng **hàm scanf để nhập chuỗi không có khoảng trắng**.

- **Xuất chuỗi:**

Cách 1: **puts (biến_chuỗi);** //Xuất chuỗi xong tự động xuống dòng

Cách 2: **printf ("%s", biến_chuỗi);**

- **Một số hàm thư viện thường dùng (thư viện <string.h>)**

TÊN HÀM	CHỨC NĂNG
int strlen(char s[]);	Trả về độ dài của chuỗi s.
strcpy(char des[], char src[]);	Sao chép nội dung chuỗi **src** vào chuỗi **des**.
strncpy(char dest[], char src[], int n);	Chép n ký tự từ chuỗi **src** sang chuỗi **dest**. Nếu chiều dài src <n thì hàm sẽ điền khoảng trắng cho đủ n ký tự vào dest.

strcat(char s1[],char s2[]);	Nối chuỗi s2 vào chuỗi s1.
strncat(char s1[],char s2[],int n)	Nối n ký tự đầu tiên của chuỗi s2 vào chuỗi s1.
Int strcmp(chars1[],char s2[])	So sánh 2 chuỗi s1 và s2 theo nguyên tắc thứ tự từ điển. Phân biệt chữ hoa và thường. Trả về: • 0 : nếu s1 bằng s2. • >0: nếu s1 lớn hơn s2. • <0: nếu s1 nhỏ hơn s2.
int stricmp(char s1[],char s2[])	Tương tự như strcmp(), nhưng không phân biệt hoa thường.
char *strstr(char s1[], char s2[]);	Tìm sự xuất hiện đầu tiên của chuỗi s2 trong chuỗi s1. Trả về: • NULL: nếu không có. • Ngược lại: Địa chỉ bắt đầu chuỗi s2 trong s1.

2.2 THỰC HÀNH CƠ BẢN

Câu 1. Viết chương trình thực hiện:

a. Nhập vào 2 chuỗi dữ liệu s1 và s2.
b. Xuất ra màn hình hai chuỗi vừa nhập
c. Xuất ra màn hình độ dài của các chuỗi vừa nhập.
d. So sánh hai chuỗi s1 và s2
e. Nối chuỗi s2 vào chuỗi s1
f. Kiểm tra chuỗi s1 có chứa chuỗi s2 hay không?
g. Kiểm tra chuỗi s2 có chứa chuỗi s1 hay không?

Câu 2. Viết hàm xóa các khoảng trắng thừa (đầu chuỗi, cuối chuỗi, khoảng trắng thừa giữa chuỗi); hàm in hoa đầu mỗi từ trong chuỗi.

Purpose	Prototype
Trim blanks at the beginning of a string: " Hello" → "Hello"	char* **lTrim**(char s[])
Trim blanks at the end of a string: "Hello " → "Hello"	char* **rTrim**(char s[])
Trim extra blanks in a string: " I am a student " → "I am a student"	char* **trim** (char s[])
Convert a string to a name: " hoang thi hoa " → "Hoang Thi Hoa"	char* **nameStr**(char s[])

Câu 3. Viết các hàm:

a. Mô phỏng hàm **LEFT** (trả về n ký tự bên trái chuỗi) của Microsoft Excel.

b. Mô phỏng hàm **RIGHT** (trả về n ký tự bên phải chuỗi) của Microsoft Excel.

```
Chuoi goc: [Kernighan and Ritchie]
left( str, 9 ) : [Kernighan]
right( str, 7 ): [Ritchie]
```

BÀI 3: KIỂU DỮ LIỆU CẤU TRÚC

3.1 TÓM TẮT LÝ THUYẾT

❖ **Định nghĩa cấu trúc**

Cấu trúc (struct) thực chất là một kiểu dữ liệu do người dùng định nghĩa bằng cách gom nhóm các kiểu dữ liệu cơ bản có sẵn trong C thành một kiểu dữ liệu phức hợp nhiều thành phần.

Cú pháp định nghĩa kiểu cấu trúc

```
struct <tên cấu trúc>
{
        <Kiểu> <Trường 1> ;
        <Kiểu> <Trường 2> ;
        …
        <Kiểu> <Trường n> ;
};
```

//dùng từ khoá typedef để định nghĩa một tên mới cho kiểu dữ liệu đã có.

typedef struct <tên cấu trúc> <**tên mới**>;

Khai báo biến kiểu cấu trúc

<tên mới> <tên biến>;

Truy xuất

```
<Tên biến cấu trúc>.<tên thành phần>;
```

❖ **Mảng cấu trúc**

Cách khai báo tương tự như mảng một chiều (Kiểu dữ liệu bây giờ là kiểu dữ liệu có cấu trúc).

Cách truy cập phần tử trong mảng cũng như truy cập trên mảng một chiều. Nhưng do từng phần tử có kiểu cấu trúc nên phải chỉ định rõ cần lấy thành phần nào.

Nguyên tắc viết chương trình có mảng cấu trúc:

Do kiểu dữ liệu có cấu trúc thường chứa rất nhiều thành phần nên khi viết chương trình loại này ta cần lưu ý:

- Xây dựng hàm xử lý cho một cấu trúc.

- Muốn xử lý cho mảng cấu trúc, ta gọi lại hàm xử lý cho một cấu trúc đã được xây dựng bằng cách dùng vòng lặp.

3.2 THỰC HÀNH CƠ BẢN

Câu 1. Định nghĩa kiểu dữ liệu sinh viên, thông tin của mỗi sinh viên gồm:

- mã sinh viên
- họ tên
- giới tính (x: nữ, y: nam)
- ngày sinh (gồm ngày, tháng và năm)
- lớp (chuỗi 7 kí tự với 2 ký tự đầu là năm học, 1 kí tự tiếp theo là bậc học (D: đại học, C: cao đẳng), 2 kí tự tiếp theo là ngành học
- điểm trung bình.

Viết chương trình gồm các hàm thực hiện:

a. Nhập danh sách sinh viên
b. Xuất danh sách sinh viên
c. Xuất các sinh viên có điểm trung bình >5.
d. Xuất danh sách sinh viên thuộc ngành công nghệ thông tin
e. Đếm số lượng sinh viên nữ.
f. Xuất các sinh viên có điểm trung bình cao nhất.
g. Thêm một sinh viên mới vào cuối danh sách.
h. Tìm sinh viên có mã là X. Nếu tìm thấy hãy xoá sinh viên đó khỏi danh sách.
i. Sắp xếp danh sách tăng theo điểm trung bình.

<u>Hướng dẫn:</u>

- Trước hết ta phải định nghĩa kiểu dữ liệu ngày tháng năm (đặt tên kiểu là DATE), định nghĩa kiểu dữ liệu sinh viên (đặt tên kiểu là SV).
- Xây dựng hàm nhập và xuất ngày tháng năm.
- Muốn nhập được danh sách sinh viên ta phải xây dựng hàm nhập cho 1 sinh viên.
- Sau đó mới xây dựng hàm nhập cho danh sách sinh viên.
- Tham khảo đoạn code sau minh hoạ cho câu a và b, các câu còn lại học viên tự làm tiếp.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#define MAX 100
//định nghĩa kiểu dữ liệu
struct DATE{
  unsigned char ngay;
  unsigned char thang;
  int nam;
};

typedef struct SinhVien{
  char ma[11];
  char hoten[31];
  struct DATE ns; //ngày sinh
  char gt; //giới tính lưu đại diện bởi 1 ký tự x: nữ, y: nam
  char lop[8];
  float dtb;
}SV;
//khai báo các nguyên mẫu hàm sẽ dùng trong chuong trình
void NhapNgaySinh(DATE &d);
void XuatNgaySinh(DATE d);
void Nhap1sv (SV &x);
void Xuat1sv (SV x);
void Nhapsl(int &n);
void Nhapds(SV a[], int n);
void Xuatds(SV a[], int n);
//-------------------------------------------------------
//hàm chính của chương trình
int main(){
  SV a[MAX];
  int n; //n lưu số lượng sinh viên của lớp
  Nhapsl(n); //nhập số lượng sinh viên
  Nhapds(a, n); //nhập danh sách sinh viên
  Xuatds(a, n);
  return 0;
}
//-------------------------------------------------------
//Code chi tiết các hàm con
void NhapNgaySinh(DATE &d){
  printf("\nNhap vao ngay: ");
  scanf("%u", &d.ngay);
```

```
  printf("\nNhap vao thang: ");
  scanf("%u", &d.thang);
  printf("\nNhap vao nam: ");
  scanf("%d", &d.nam);
}
void XuatNgaySinh(DATE d){
  printf("%02u / %02u / %4d", d.ngay, d.thang, d.nam);
}
//-----------------------------------------------------------
//nhập thông tin cho 1 sinh viên
void Nhap1sv (SV &x){
  printf("Nhap ma sinh vien: "); scanf("%s", &x.ma);
  printf("Nhap ho ten: ");
  fflush(stdin); //Xoa vung dem
  gets(x.hoten);
  printf("Nhap ngay thang nam sinh: "); NhapNgaySinh(x.ns);
  printf("Nhap lop: "); scanf("%s", &x.lop);
  do{
       printf("Nhap gioi tinh x-nu, y-nam: ");
       x.gt=getche();
  }while (x.gt!='x' && x.gt!='y');
  printf("\nNhap vao diem trung binh: "); scanf("%f", &x.dtb);
}
//-----------------------------------------------------------
//xuất thông tin của 1 sinh viên ra màn hình
void Xuat1sv (SV x){
  printf("\n%-11s\t-30s\t", x.ma, x.hoten);
  if (x.gt=='x') printf("nu\t");
  else printf("nam\t");
  XuatNgaySinh(x.ns);
  printf("\t%-8s\t%.1f\n", x.lop, x.dtb);
}
//-----------------------------------------------------------
//Nhập số lượng sinh viên của danh sách
void Nhapsl(int &n)
{
  //nhập số lượng n>0, nếu nhập sai bắt nhập lại
  // sinh viên tự code
}
//Nhập thông tin của từng sinh viên trong danh sách
void Nhapds(SV a[], int n){
  printf("\NHAP DANH SACH SINH VIEN---------- \n");
```

```
  for(int i=0; i<n; i++){
       printf("\nNhap sinh vien thu %d:\n", i+1);
       Nhap1sv(a[i]); //Gọi hàm nhập thông tin cho 1 sinh viên thứ i.
  }
}
//-------------------------------------------------------
//xuất danh sách sinh viên ra màn hình
void Xuatds(SV a[], int n){
  printf("\nDANH SACH SINH VIEN----------------\n");
  for(int i=0; i<n; i++)
       Xuat1sv(a[i]); // Gọi hàm xuất thông tin cho 1 sinh viên thứ i.
}
```

Câu 2. Viết chương trình quản lý các bưu kiện của bưu điện, sử dụng mảng một chiều để lưu danh sách các bưu kiện. Thông tin mỗi bưu kiện gồm: mã bưu kiện, tên người gửi, tên người nhận, trọng lượng, ngày gửi (ngày, tháng, năm), nội dung bưu kiện, đơn giá gửi.

Chương trình có các chức năng sau:

a. Nhập thông tin của các bưu kiện
b. Xuất thông tin của các bưu kiện
c. Thêm một bưu kiện vào danh sách
d. Sắp xếp danh sách các bưu kiện theo mã bưu kiện
e. Tính giá trị của mỗi bưu kiện biết giá trị = trọng lượng x đơn giá gửi.
f. Đếm số lượng bưu kiện có trọng lượng lớn nhất

Câu 3. Viết chương trình quản lý các thuê bao điện thoại, sử dụng mảng 1 chiều để lưu danh sách các thuê bao. Thông tin mỗi thuê bao gồm: mã thuê bao, họ tên chủ thuê bao, ngày đăng ký (ngày tháng năm), số điện thoại, loại thuê bao (TT: thuê bao trả trước, TS: thuê bao trả sau), thời gian gọi nội mạng, thời gian gọi ngoại mạng (đơn vị là phút).

Chương trình có các chức năng sau:

a. Nhập thông tin của các thuê bao
b. Xuất thông tin của các thuê bao
c. Thêm một thuê bao vào danh sách
d. Sắp xếp danh sách các thuê bao theo mã thuê bao
e. Tìm thuê bao theo họ tên chủ thuê bao

f. Xuất các thuê bao theo loại (loại nào là do người dùng chọn)
g. Xuất các thuê bao đăng kí sau năm 2020
h. Tính cước phí của mỗi thuê bao biết giá cước gọi nội mạng là 1500đ, ngoại mạng là 3000đ
i. Đếm số lượng thuê bao trả trước.

Câu 4. Viết chương trình quản lý sách cho một cửa hàng sách, sử dụng mảng 1 chiều để lưu các cuốn sách; thông tin của mỗi cuốn sách gồm: mã sách, tên sách, tên tác giả, loại sách (gồm 2 loại Tự nhiên và Xã hội), năm xuất bản, giá tiền, số lượng.

Chương trình có các chức năng sau:

a. Nhập thông tin các cuốn sách.
b. Xuất thông tin các cuốn sách.
c. Thêm 1 cuốn sách.
d. Tính tổng thành tiền tất cả cả các cuốn sách.
e. Sắp xếp các cuốn sách theo mã sách.
f. Tìm sách theo tên sách.
g. Xuất các cuốn sách có năm xuất bản trước năm 2000.
h. Đếm số lượng sách có giá lớn hơn 50000.
i. Xuất các cuốn sách theo loại (xuất loại nào là do người dùng chọn).

Câu 5. Viết chương trình cài đặt một mảng các cấu trúc lưu trữ thông tin sách trong thư viện, bao gồm: **tựa sách, ISBN, tên tác giả, tên nhà xuất bản, ngày tháng năm nhập sách** (là ngày viết phiếu). Sau đó, nhập vào một chuỗi ISBN, tìm và in ra thông tin sách tương ứng nếu có.

Hướng dẫn:

Dùng `struct tm` của `time.h` để lưu trữ ngày cập nhật phiếu.

```
Nhap thong tin sach:
 Tua     > The C Programming Language ↵
 ISBN    > 0-131-10370-9 ↵
 Tac gia > Kernighan, Brian W. & Ritchie, Dennis M. ↵
 NXB     > Prentice Hall ↵
Tiep ( y/n )? y ↵
Nhap thong tin sach:
 Tua     > Applications Programming in ANSI C ↵
 ISBN    > 0-023-61141-3
 Tac gia > Johnsonbaugh, R. & Kalin M. ↵
 NXB     > MacMillan Pub. Co. ↵
Tiep ( y/n )? n ↵
ISBN ? 0-023-61141-3 ↵
Ket qua tim:
Applications Programming in ANSI C
 Johnsonbaugh, R. & Kalin M.
 MacMillan Pub. Co.
 [update: 15-04-2006]
```

BÀI 4: LẬP TRÌNH ĐỆ QUY

4.1 TÓM TẮT LÝ THUYẾT

Một hàm được gọi có tính đệ quy nếu trong thân của hàm đó có lệnh gọi lại chính nó.

Kỹ thuật giải bài toán bằng đệ quy:

1. Tham số hóa bài toán.
2. Tìm các điều kiện biên (chặn, dừng), tìm giải thuật cho các tình huống này.
3. Tìm giải thuật tổng quát theo hướng đệ quy lui dần về tình huống bị chặn.

Lưu ý

Đệ quy cung cấp cho ta cơ chế giải quyết các bài toán phức tạp một cách đơn giản hơn.

- Xây dựng hàm đệ qui thông qua việc xác định điều kiện dừng và bước thực hiện tiếp theo.
- Chỉ nên cài đặt bằng phương pháp đệ quy khi không còn cách giải quyết bằng cách lặp thông thường.

4.2 THỰC HÀNH CƠ BẢN

Câu 1. Viết chương trình thực hiện (dùng đệ quy):

a. Nhập 1 mảng số nguyên.

b. Xuất mảng số nguyên

c. Tính tổng các phần tử của mảng

d. Tính tổng các phần tử chẵn của mảng

e. Đếm số lượng phần tử dương

f. Tìm phần tử lớn nhất (nhỏ nhất) của mảng

g. Tìm phần tử chẵn cuối cùng có trong mảng

h. Nhập 1 trị x, tìm vị trí có x cuối cùng trong mảng.

Hướng dẫn:

```
void Nhap(int a[], int n)
{
   if(n==0)
     return;
   Nhap(a, n-1); //nhập cho n-1 phần tử đầu
   printf("\nNhap phan tu thu %d: ", n-1);
   scanf("%d", &a[n-1]); //nhập cho phần tử cuối trong mảng
}
void Xuat(int a[], int n)
{
   if(n==0)
     return;
   Xuat(a, n-1);
   printf("%d\t", a[n-1]);
}
```

Câu 2. Giả sử dân số thế giới năm 2018 là 8 tỷ người, với mức tăng dân số hàng năm không đổi là 2,5%. Viết hàm đệ quy tính dân số thế giới năm 2028

Câu 3. Viết hàm đệ quy tính tổng các chữ số của một số nguyên n.

Ví dụ: n=1980 => Sum = 1 + 9 + 8 + 0 = 18

Câu 4. Tính dãy Fibonaci dùng kỹ thuật đệ quy.

Câu 5. Viết chương trình thực hiện (dùng đệ quy):

a. Tìm USCLN của 2 sô nguyên dương

b. Tìm BSCNN của 2 sô nguyên dương.

Câu 6. Cho hàm Ackermann với n và m không âm:

$$A(n,m)=\begin{cases} m+1 & n=0 \\ A(n-1,1) & m=0 \\ A(n-1,A(n,m-1)) & n,m>0 \end{cases}$$

Viết hàm đệ quy tính Ackermann của n, m. Cho biết số lần gọi đệ quy khi tính Ackermann của n, m.

Ví dụ:

Tính `A(3, 6)`. Cho biết số lần gọi đệ quy khi tính `A(3, 6)`.

```
A( 3, 6 ) = 509
Goi de quy 172233 lan
```

Câu 7. Bài toán Tháp Hà Nội (Tower of Hanoi) sử dụng đệ quy

Bài toán tháp Hanoi (Towers of Hanoi): có 3 cọc A, B, C; khởi đầu cọc A có n đĩa xếp sao cho đĩa lớn hơn luôn nằm bên dưới, 2 cọc kia trống. Hãy chuyển tất cả đĩa từ cọc A sang cọc C, được dùng cọc phụ B. trong quá trình chuyển đĩa phải đảm bảo đĩa lớn hơn luôn nằm bên dưới.
Minh họa với 2 đĩa:

Xem thêm về bài này trong nhiều sách về lập trình.

```
Disk 1: [1] -> [2]
Disk 2: [1] -> [3]
Disk 1: [2] -> [3]
Disk 3: [1] -> [2]
Disk 1: [3] -> [1]
```

BÀI 5: TẬP TIN (FILE)

5.1 TÓM TẮT LÝ THUYẾT

Trong các chương trình trước thì các dữ liệu đưa vào chương trình chỉ được tồn tại trong RAM, khi thoát chương trình thì tất cả dữ liệu đều bị mất. Để khắc phục tình trạng này các

ngôn ngữ lập trình cung cấp cho chúng ta các hàm để lưu trữ và truy xuất tập tin, đó là kiểu FILE. Và ở đây ta chỉ đề cập đến 2 loại tập tin:

- Tập tin văn bản: là tập tin dùng để ghi các ký tự lên đĩa theo các dòng.
- Tập tin nhị phân: là tập tin dùng để ghi các cấu trúc dạng nhị phân (được mã hoá).

Quá trình thao tác trên tập tin thông qua 4 bước:

- Bước 1: Khai báo con trỏ trỏ đến tập tin.
- Bước 2: Mở tập tin.
- Bước 3: Các xử lý trên tập tin.
- Bước 4: Đóng tập tin.

Khai báo

```
FILE *  tên biến;
```

Ví dụ : FILE *f; *// Khai bao bien con tro file f*

Mở tập tin

```
fopen (đường dẫn tên tập tin, "kiểu truy cập");
```

Ví dụ : FILE *f = fopen ("C:\\VD1.txt" , "rt") ;

Các kiểu truy nhập tập tin thông dụng:

t là kiểu truy nhập tập tin đối với dạng tập tin văn bản (text). **b** là kiểu truy nhập tập tin đối với dạng tập tin nhị phân (binary). **r** mở ra để đọc (ready only). **w** mở ra để ghi (create / write). **a** mở ra để thêm vào (append). **r+** mở ra để đọc và ghi (modify).

Đóng tập tin

```
fclose ( <biến con trỏ tập tin> ) ;

hoặc fcloseall () ;
```

Các hàm đọc ghi tập tin văn bản

TÊN HÀM	Ý NGHĨA	VÍ DỤ

Đọc dữ liệu từ tập tin ra bi ến (nhập dữ liệu cho biến t ừ file)		
fscanf(biến file, "định dạng", các tham biến);	Đọc dữ liệu từ một tập tin theo định dạng.	int x; float y; fscanf(f, "%d%f", &x, &y);
fgets(biến chuỗi, kích thước tối đa, biến file);	Đọc một chuỗi ký tự từ một tập tin với kích thước tối đa cho phép, hoặc gặp ký tự xuống dòng.	char s[80]; fgets(s, 79, f);
biến kí tự = getc(biến file);	Đọc một ký tự từ tập tin đang mở.	char ch=getc(f);
Ghi tập tin (ghi dữ liệu (từ biến) ra tập tin		
fprintf(biến file, "chuỗi định dạng" [, <các tham biến nếu có>]);	Ghi dữ liệu theo một định dạng nào đó vào tập tin.	fprintf(f,"Ket qua la %d\n",x);
fputs(chuỗi ký tự, biến file);	Ghi một chuỗi ký tự vào tập tin đang mở.	fputs("Chuong trinh minh hoa", f);

Các hàm đọc ghi tập tin nhị phân

TÊN HÀM	Ý NGHĨA	VÍ DỤ
Đọc dữ liệu từ tập tin ra biến (nhập dữ liệu cho biến từ file)		
fread(&ptr, size, len, biến file);	• ptr: vùng nhớ để lưu dữ liệu đọc. • size: kích thước mỗi ô nhớ (tính bằng byte). • len: độ dài dữ liệu cần đọc.	int a[30], n; fread(a, sizeof(int), n , f); SV b; Fread(&b, sizeof(SV), 1 , f);
Ghi tập tin (ghi dữ liệu (t ừ biến) ra tập tin		
fwrite(&prt, size, len, biến file);	Tham số tương tự như hàm fread.	fwrite(a, sizeof(int), n , f);

5.2 MỘT SỐ VÍ DỤ

Example 1: Ghi dữ liệu vào file dùng hàm fprintf()

```
#include <stdio.h>
#include <stdlib.h>
int main(){
   int num;
   FILE *fptr;
   fptr = fopen("C:\\program.txt","w");
   if(fptr == NULL){
      printf("Error!");
      exit(1);
   }
   printf("Enter num: ");
   scanf("%d",&num);
   fprintf(fptr,"%d",num);
   fclose(fptr);
   return 0;
}
```

Example 2: Đọc dữ liệu trong file dùng hàm fscanf()

```
#include <stdio.h>
#include <stdlib.h>

int main(){
   int num;
   FILE *fptr;
   if ((fptr = fopen("program.txt","r")) == NULL){
       printf("Error! opening file");
       // Program exits if the file pointer returns NULL.
       exit(1);
   }
   fscanf(fptr,"%d", &num);
   printf("Value of n=%d", num);
   fclose(fptr);
   return 0;
}
```

Example 3: Write a C program to read name and marks of n number of students from user and store them in a file.

```
#include <stdio.h>
int main(){
   char name[10];
   int marks, i, num;
   printf("Enter number of students: ");
   scanf("%d", &num);
   FILE *fptr;
   fptr = (fopen("C:\\student.txt", "w"));
   if(fptr == NULL){
       printf("Error!");
       exit(1);
   }

   for(i = 0; i < num; ++i){
      printf("For student%d\nEnter name: ", i+1);
      scanf("%s", name);
      printf("Enter marks: ");
      scanf("%d", &marks);
      fprintf(fptr,"\nName: %s \nMarks=%d \n", name, marks);
   }

   fclose(fptr);
   return 0;
}
```

Example 4: Write a C program to read name and marks of n number of students from user and store them in a file. If the file previously exits, add the information of n students.

```
#include <stdio.h>
int main(){
   char name[50];
   int marks, i, num;
   printf("Enter number of students: ");
   scanf("%d", &num);
   FILE *fptr;
   fptr = (fopen("C:\\student.txt", "a"));
   if(fptr == NULL){
       printf("Error!");
```

```
        exit(1);
    }
    for(i = 0; i < num; ++i){
        printf("For student%d\nEnter name: ", i+1);
        gets(name);
        printf("Enter marks: ");
        scanf("%d", &marks);
        fprintf(fptr,"\nName: %s \nMarks=%d \n", name, marks);
    }
    fclose(fptr);
    return 0;
}
```

Example 5: Write a C program to write all the members of an array of structures to a file using fwrite(). Read the array from the file and display on the screen.

```
#include <stdio.h>
struct student{
    char name[50];
    int height;
};
int main(){
    struct student stud1[5], stud2[5];
    FILE *fptr;
    int i;
    fptr = fopen("file.txt","wb");
    for(i = 0; i < 5; ++i){
        fflush(stdin);
        printf("Enter name: ");
        gets(stud1[i].name);
        printf("Enter height: ");
        scanf("%d", &stud1[i].height);
    }
    fwrite(stud1, sizeof(stud1), 1, fptr);
    fclose(fptr);

    fptr = fopen("file.txt", "rb");
    fread(stud2, sizeof(stud2), 1, fptr);
    for(i = 0; i < 5; ++i){
        printf("Name: %s\nHeight: %d", stud2[i].name,stud2[i].height);
    }
    fclose(fptr);
```

```
}
```

5.3 THỰC HÀNH TẬP TIN

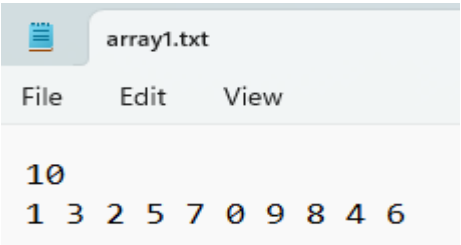
Câu 1. Cho file TXT có cấu trúc sau:

- Dòng đầu lưu giá trị một số nguyên dương n
- Dòng còn lại lưu giá trị của một dãy gồm n các số nguyên.

a. Đọc file trên, lưu dữ liệu đọc được vào mảng một chiều.

b. Xuất mảng ra màn hình

c. Ghi các số nguyên tố có trong mảng vào file (ghi tiếp theo vào file đã có).



Câu 2. Viết chương trình phát sinh ngẫu nhiên ma trận a kích thước 5x6, lưu ma trận này vào file test.inp. Đọc lại file test.inp đưa dữ liệu vào ma trận b và xuất ra màn hình xem kết quả lưu đúng không? Cấu trúc của file test.inp như sau:

- Dòng đầu lưu 2 số nguyên: d, c thể hiện số dòng và số cột của ma trận.
- d dòng tiếp theo, mỗi dòng gồm c phần tử là giá trị các phần tử trên một dòng của ma trận.

Hướng dẫn:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define MAX 100
#define duongdan "D:\\test.inp"
//sinh ma trận ngẫu nhiên----------------------------------------
void SinhMT(int a[MAX][MAX], int d, int c) {
   /*Do yourself*/
}
//ghi ma trận vào file-------------------------------------------
void LuuFile(int a[MAX][MAX], int d, int c) {
   FILE *f; f=fopen(duongdan, "wt");
```

```
   if(f==NULL) {
      printf("\nKhong tao duoc file.");
      getch();
      exit(0);
   }
   fprintf(f, "%d %d\n", d, c);
   for(int i=0; i<d; i++){
      for(int j=0; j<c; j++)
         fprintf(f, "%d\t", a[i][j]);
      fprintf(f, "\n");
   }
   fclose(f);
}

//đọc dữ liệu từ file ra biến mảng lưu ma trận----------------------------
void DocFile(int a[MAX][MAX], int &d, int &c) {
   /* mở file như trên
   Đọc số dònng, số cột
   Đọc giá trị các phần tủ của ma trận
   đóng file
   */
}
//xuất ma trận ------------------------------------------
void XuatMT(int a[MAX][MAX], int d, int c) {
   /*Do yourself*/
}

int main(){
   int a[MAX][MAX], d=5, c=6;
   int b[MAX][MAX], x, y;
   SinhMT(a, d, c);
   LuuFile(a, d, c); //ghi dữ liệu từ mảng a xuống file
   DocFile(b, x, y); //đọc dữ liệu từ file ra mảng b
   XuatMT(b, x, y);
   return 0;
}
```

Câu 3. Viết chương trình đọc một chuỗi tối đa 100 kí tự từ bàn phím. Lưu các ký tự là nguyên âm vào tập tin "NguyenAm.txt". Đọc các kí tự từ tập tin này và hiển thị lên màn hình.

Câu 4. Viết chương trình cho phép nhập từ bàn phím và ghi vào 1 tập tin tên DSHH.TXT với mỗi phần tử của tập tin là một cấu trúc bao gồm các trường:

- mh (mã hàng: char[5]).
- sl (số lượng: int).
- dg (đơn giá: float).
- st (Số tiền: float) = số lượng * đơn giá.

Sau khi nhập xong yêu cầu in toàn bộ danh sách hàng hóa ra màn hình.

Câu 5. Tạo tập tin văn bản PERSON.DAT lưu thông tin cá nhân thành các dòng có định dạng như sau: (trong () là yêu cầu của trường): (5 dòng)

code(unsigned int):firstname lastname(32),address(32):birthday(mm/dd/yy)

Viết chương trình đọc tập tin PERSON.DAT, lấy và hiển thị thông tin lưu trữ ứng với từng cá nhân.

Nội dung tập tin PERSON.DAT

```
1654:Jackie Chan,Hong Kong:7/22/54
4424:Tony Jaa,Thailand:5/12/76
```

Kết quả

```
Jackie Chan                    [code: 1654]
  Address : [Hong Kong]
  Birthday: [7/22/54]
Tony Jaa                       [code: 4424]
  Address : [Thailand]
  Birthday: [5/12/76]
```