

CHƯƠNG 7: KIỂU CẤU TRÚC - FILE

7.1 Kiểu cấu trúc

Khai báo :

```
struct TÊN_KIỂU {  
    T1  x1;  
    T2  x2;  
    T3  x3;  
};
```

- Qui tắc đặt tên biến : Alphabet + Alphabet / Ký số / dấu gạch nối
- TÊN_KIỂU : tên của kiểu cấu trúc, đặt theo qui tắc tên biến
- x1, x2, x3 các thành phần (thuộc tính, trường) ,có chức năng là biến , có kiểu tương ứng T1, T2, T3.

7.2 Truy xuất thành phần thứ i :

```
struct TÊN_KIỂU {
```

```
    T1  x1;
```

```
    T2  x2;
```

```
    T3  x3;
```

```
};
```

```
void main( )
```

```
{ TÊN_KIỂU V;
```

```
}
```



V.xi

Ví dụ 1 :

```
struct VECTOR {  
    int x ;  
    int y ;  
} ;  
  
int main(int argc, char* argv[])  
{ VECTOR V;  
    printf("x ="); scanf("%d", &V.x);  
    printf("y ="); scanf("%d", &V.y);  
    printf("V=(%d, %d)\n",V.x, V.y);  
    return 0;  
}
```

Ví dụ 2 :

```
struct VECTOR {  
    int x ;  
    int y ;  
} ;  
  
typedef VECTOR MY_VECTOR;  
  
int main(int argc, char* argv[])  
{ MY_VECTOR V;  
    printf("x ="); scanf("%d", &V.x);  
    printf("y ="); scanf("%d", &V.y);  
    printf("V=(%d, %d)\n",V.x, V.y);  
    return 0;  
}
```

Ví dụ 3 :

```
struct SINH_VIEN {  
    char hoten[30] ;  
    int diem ;  
} ;  
  
int main(int argc, char* argv[])  
{ SINH_VIEN SV;  
    printf("Ho ten :"); gets(SV.hoten);  
    printf("diem :"); scanf("%d", &SV.diem);  
    printf("Sinh vien : %s , diem : %d \n", SV.hoten, SV.diem);  
}
```

Ví dụ 4 :

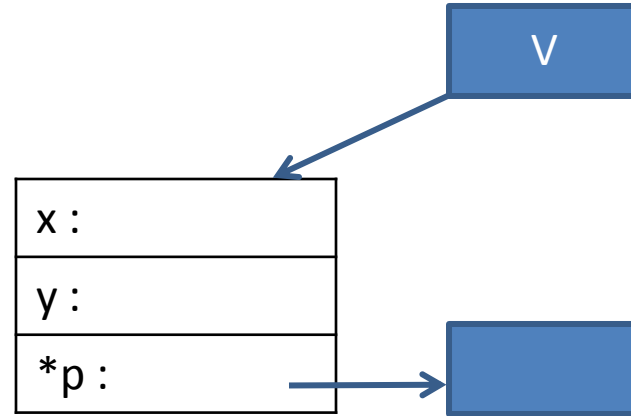
```
struct CON_TRO {  
    int x ;  
    int *p ;  
}  
;  
int main(int argc, char* argv[])  
{CON_TRO V;  
    V.p = (int*)malloc(sizeof(int));  
    printf("x :"); scanf("%d", &V.x);  
    printf("*p :"); scanf("%d",V.p);  
    printf("Gia tri : %d , %d \n",V.x, *(V.p));  
}
```

Ví dụ 5 :

```
struct MANG {  
    int N ;  
    int M[100] ;  
} ;  
  
int main(int argc, char* argv[])  
{ MANG V;  
  V.N=2;  
  V.M[0]= 10; V.M[1]= 20;  
  printf("So phan tu : %d , Tong mang = %d \n",V.N, V.M[0]+V.M[1]);  
  return 0;  
}
```

Ví dụ 6 :

```
struct CONTRO_CAUTRUC {  
    int x ;  
    int y ;  
    int *p;  
};
```



```
int main(int argc, char* argv[])
```

```
{CONTRO_CAUTRUC *V;
```

```
1. V = (CONTRO_CAUTRUC*)malloc(sizeof(CONTRO_CAUTRUC));
```

```
2. (*V).p = (int*)malloc(sizeof(int)); // V→p = (int*)malloc(sizeof(int));
```

```
3. (*V).x=2;      // V→x = 2;
```

```
4. (*V).y=5;      // V→y = 5;
```

```
printf("*p = "); scanf("%d", V→p);
```

```
printf(" %d , %d, %d ", (*V).x, (*V).y, *(V→p) );
```

```
return 0;
```

```
}
```


Ví dụ 6 :

```
struct CONTRO_CAUTRUC {  
    int x ;  
    int y ;  
} ;
```

```
int main(int argc, char* argv[])
```

```
{CONTRO_CAUTRUC *V;
```

```
V = (CONTRO_CAUTRUC*)malloc(sizeof(CONTRO_CAUTRUC));
```

```
printf("x = "); scanf("%d", &V→x);
```

```
printf("y = "); scanf("%d", &V→y);
```

```
printf(" %d , %d ",(*V).x, (*V).y );
```

```
return 0;
```

```
}
```

Ví dụ 7 :

```
struct CONTRO_CAUTRUC {  
    int x ;  
    int y ;  
} ;  
  
int main(int argc, char* argv[])  
{CONTRO_CAUTRUC *V;  
V = (CONTRO_CAUTRUC*)malloc(2*sizeof(CONTRO_CAUTRUC));  
V[0].x=10; V[0].y=20;  
V[1].x=100; V[1].y=200;  
printf("Tong V[0] = %d, Tong V[1] = %d \n", V[0].x + V[0].y,  
                                             V[1].x + V[1].y );  
}
```

V[0]	V[1]
x :	x :
y :	y :

Ví dụ 8 :

```
struct VECTOR {
```

```
    int x ;   int y ;
```

```
} ;
```

```
void nhap_vector(VECTOR *p)
```

```
{ printf("x = "); scanf("%d", &p→x);
```

```
    printf("y = "); scanf("%d", &p→y); // cin>> p→y;
```

```
}
```

```
void viet_vector(VECTOR Vp)
```

```
{ printf("vector = (%d , %d) ",Vp.x, Vp.y); }
```

```
-----
```

```
int main(int argc, char* argv[])
```

```
{ VECTOR V;
```

```
    nhap_vector(&V); viet_vector(V); }
```

Ví dụ 8 :

```
struct VECTOR {  
    int x ;   int y ;  
}  
;  
  
void nhap_vector(VECTOR &p)  
{ printf("x = "); scanf("%d", &p.x); // cin>>p.x ;  
  printf("y = "); scanf("%d", &p.y);  
}  
  
void viet_vector(VECTOR p)  
{ printf("vector = (%d , %d) ",p.x, p.y); }  
  
-----  
  
int main(int argc, char* argv[])  
{ VECTOR V;  
  nhap_vector(V); viet_vector(V); }
```

7.3 Mảng cấu trúc :

Khai báo (mảng tĩnh):

KIỂU_CẤU_TRÚC *BIẾN[N]* ;

Truy xuất một thành phần trong *BIẾN[i]* :

BIẾN[i].Thành phần

7.3 Mảng cấu trúc :

Mảng động :

KIỂU_CẤU_TRÚC **BIẾN*;

Cấp phát mảng động :

***BIẾN* = (KIỂU_CẤU_TRÚC*) malloc(N*sizeof(KIỂU_CẤU_TRÚC));**

Truy xuất mảng động : Mảng động (sau khi được cấp phát vùng nhớ) được xem như mảng tĩnh (\equiv *BIẾN*[*N*]).

***BIẾN*[*i*].Thành phần**

7.4 Kiểu file :

File :

- Dữ liệu được lưu ở biến của chương trình, và nó sẽ biến mất khi chương trình kết thúc.
- Sử dụng file để lưu trữ dữ liệu cần thiết để đảm bảo dữ liệu của chúng ta không bị mất ngay cả khi chương trình của chúng ta ngừng chạy.

VD : file được lưu trên USB.

Các kiểu file :

- File văn bản – text files :
 - File văn bản là file thường có đuôi là **.txt**.
 - File có thể dễ dàng tạo ra bằng cách dùng các text editor thông dụng như Notepad.
- File nhị phân – Binary files :
 - File nhị phân thường có đuôi mở rộng là **.bin**,
 - Dữ liệu trong file này được lưu dưới dạng nhị phân, chỉ bao gồm các số 0 và 1.

Các thao tác với file :

- Tạo mới một file
- Mở một file đã có
- Đóng file đang mở
- Đọc thông tin từ file / Ghi thông tin ra file

Khai báo :

```
FILE *fptr;
```

Mở file (ngôn ngữ C) : Dùng ghi hay đọc dữ liệu từ **file**.

```
fptr = fopen("fileopen","mode");
```

- fileopen : Đường dẫn tới file.
- mode : Chỉ định mở file để đọc, ghi, . . .

Đóng file :

```
fclose(fptr);
```

Ví dụ :

```
int main(int argc, char* argv[])
{
    FILE *fptr;
    fptr = fopen("D:\\TESTFILE\\myfile.txt","w");
    if(fptr == NULL)
    {
        printf("Error!");
        return 0;
    }
    . . .
}
```

Các tham số của “mode” :

`fptr = fopen("fileopen", "mode");`

Mode	Ý nghĩa	Nếu file không tồn tại
r	Mở file chỉ cho phép đọc dưới dạng text.	Nếu file không tồn tại, fopen() trả về NULL.
rb	Mở file chỉ cho phép đọc dưới dạng nhị phân.	Nếu file không tồn tại, fopen() trả về NULL.
w	Mở file chỉ cho phép ghi dưới dạng text	Nếu file đã tồn tại, nội dung sẽ bị ghi đè. Nếu file không tồn tại, nó sẽ được tạo tự động.
wb	Mở file chỉ cho phép ghi dưới dạng nhị phân.	Nếu file đã tồn tại, nội dung sẽ bị ghi đè. Nếu file không tồn tại, nó sẽ được tạo tự động.

Ghi file văn bản :

```
int main(int argc, char* argv[])  
{ int num1, num2;  
  FILE *fptr;  
  
  fptr = fopen("D:\\TESTFILE\\myfile.txt","w");
```

Kiểm tra lỗi mở file;

```
printf("Num 1: "); scanf("%d",&num1);  
printf("Num 2: "); scanf("%d",&num2);
```

```
fprintf(fptr,"%d□□□%d", num1, num2);
```

```
fclose(fptr);  
}
```

Thực hiện chương trình :

Num1 : 12

Num2 : 45

Nội dung file :

12□□□45

Ghi file văn bản :

```
int main(int argc, char* argv[])
```

```
{ int num1, num2;
```

```
FILE *fptr;
```

```
fptr = fopen("D:\\TESTFILE\\myfile.txt","w");
```

Kiểm tra lỗi mở file;

```
printf("Num 1: "); scanf("%d",&num1);
```

```
printf("Num 2: "); scanf("%d",&num2);
```

```
fprintf(fptr,"%d\n%d",num1,num2);
```

```
fclose(fptr);
```

```
}
```

Thực hiện chương trình :

Num1 : 12

Num2 : 45

Nội dung file :

12

45

Đọc file văn bản :

```
int main(int argc, char* argv[])
```

```
{ int num1, num2;
```

```
FILE *fptr;
```

```
fptr = fopen("D:\\TESTFILE\\myfile.txt", "r");
```

Kiểm tra lỗi mở file;

```
fscanf(fptr, "%d",&num1);
```

```
fscanf(fptr, "%d",&num2);
```

(hay

fscanf(fptr, "%d%d",&num1, &num2);

)

```
printf("%d□□□%d", num1, num2);
```

```
fclose(fptr);
```

```
}
```

Nội dung file :

12□□□45

hay

12

45

Đọc file văn bản : Tính tổng các số nguyên trong file

```
int main(int argc, char* argv[])
```

```
{ int num, x, T = 0;
```

```
FILE *fptr;
```

```
fptr = fopen("D:\\TESTFILE\\myfile.txt", "r");
```

Kiểm tra lỗi mở file;

```
fscanf(fptr, "%d",&num);
```

```
T=0; i=1;
```

```
while (i<=num) {
```

```
    fscanf(fptr, "%d",&x);    T=T+x;
```

```
    i++;
```

```
}
```

```
printf("T=%d",T);
```

```
fclose(fptr);
```

```
}
```

Nội dung file :

5

1 3 6 7 9

Ghi file nhị phân :

Ví dụ 1:

VECTOR **V**;

FILE *fptr;

fptr = fopen("D:\\TESTFILE\\myfile.dat", "**wb**");

Kiểm tra mở file;

printf("Nhap x, y :");scanf("%d%d", &V.x,&V.y);

fwrite(&V**, sizeof(**VECTOR**), **1**, fptr);**

fclose(fptr);



1 giá trị kiểu VECTOR được
viết vào file

Ghi file nhị phân :

Ví dụ 2 :

VECTOR **V[2]**;

FILE *fptr;

fptr = fopen("D:\\TESTFILE\\myfile.dat", "**wb**");

Kiểm tra mở file;

printf("Nhap x[0], y[0] :"); scanf("%d%d", &V[0].x,&V[0].y);

printf("Nhap x[1], y[1] :"); scanf("%d%d", &V[1].x,&V[1].y);

fwrite(V, sizeof(VECTOR), 2, fptr);

fclose(fptr);



2 giá trị kiểu VECTOR được
viết vào file

Đọc file nhị phân : Giả sử đã có file ở Ví dụ 1 , ghi file nhị phân

Ví dụ 3:

VECTOR V, **Vr**;

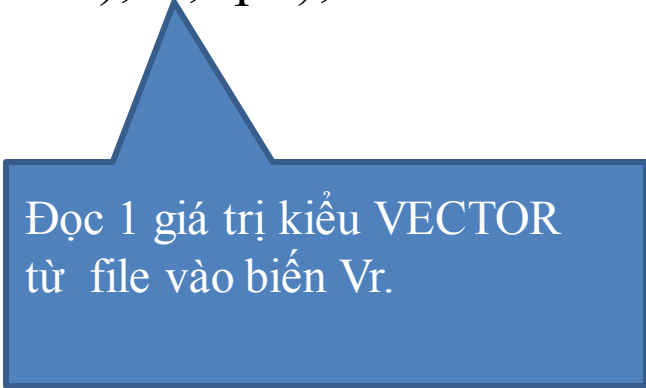
FILE *fptr;

fptr = fopen("D:\\TESTFILE\\myfile.dat", "**rb**");

Kiểm tra mở file;

fread(&**Vr**, sizeof(**VECTOR**), **1**, fptr);

fclose(fptr);



Đọc 1 giá trị kiểu VECTOR
từ file vào biến Vr.

Đọc file nhị phân :

Ví dụ 4 :

```
VECTOR V[2], Vr[2];
```

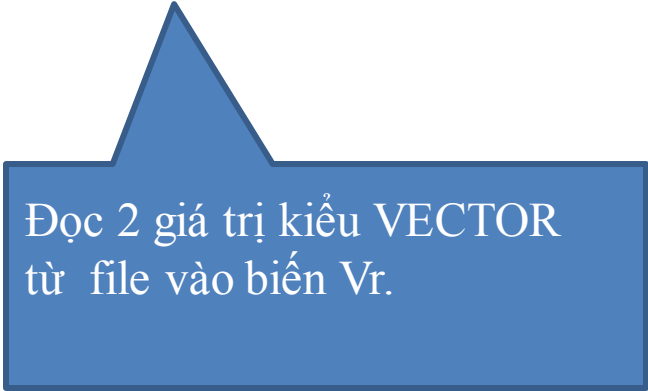
```
FILE *fptr;
```

```
fptr = fopen("D:\\TESTFILE\\myfile.dat", "rb");
```

```
Kiểm tra mở file;
```

```
fread(Vr, sizeof(VECTOR), 2, fptr);
```

```
fclose(fptr);
```



Đọc 2 giá trị kiểu VECTOR
từ file vào biến Vr.

Đọc file nhị phân :

Ví dụ 5 : Nhập n SV , một SV có các thuộc tính họ tên, điểm.

`SINH_VIEN SV; // Xem phần kiểu cấu trúc`

`FILE *fptr;`

Mở file để ghi ;

`for(i=1; i <= n; i++){`

`gets(SV.hoten); scanf("%d", &SV.diem); scanf("%c", &c);`

`fwrite(&SV, sizeof(SINH_VIEN), 1, fptr);`

`}`

`fclose(fptr);`

Mở file để đọc ;

`for(i=1; i <=n; i++){`

`fread(&SV, sizeof(SINH_VIEN), 1, fptr);`

`printf("%s %d\n",SV.hoten, SV.diem);`

`}`

`fclose(fptr);`

Đọc file nhị phân :

Ví dụ 6 : Nhập n (=5) SV. Viết ra màn hình SV thứ k (=4).

`SINH_VIEN SV; // Xem phần kiểu cấu trúc`

`FILE *fptr;`

Tạo file n SV; Mở file để đọc ;

`if(fseek(fptr, (k-1)*sizeof(SINH_VIEN), SEEK_SET)==0) {`

`fread(&SV, sizeof(SINH_VIEN), 1, fptr);`

`printf("%s %d\n",SV.hoten, SV.diem);`

`}`

`fclose(fptr);`

Hằng	Miêu tả
SEEK_SET	Tính từ đầu file
SEEK_CUR	Tính từ vị trí hiện tại của con trỏ file
SEEK_END	Tính từ cuối file

- $(k-1)*sizeof(SINH_VIEN), SEEK_SET \equiv$ Phần tử thứ k trong file