CHƯƠNG 7: KIỂU CẤU TRÚC

7.1 Khai báo và truy xuất:

Khai báo :

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

- TÊN_KIỂU : tên của kiểu cấu trúc, đặt theo qui tắc tên biến
- x1, x2, x3 các thành phần có chức năng là biến có kiểu tương ứng T1, T2, T3.

CHƯƠNG 7: KIỂU CẤU TRÚC

Truy xuất thành phần thứ i :

```
struct TÊN_KIỂU {
        T1   x1;
        T2   x2;
        T3   x3;
};
void main( )
{ TÊN_KIỂU V;
}
```

V.xi

CHƯƠNG 7: KIỂU CẤU TRÚC

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;
}
```

CHƯƠNG 7: KIỂU CẤU TRÚC

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;
}
```

CHƯƠNG 7: KIỂU CẤU TRÚC

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);
}
```

CHƯƠNG 7: KIỂU CẤU TRÚC

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));
}
```

CHƯƠNG 7: KIỂU CẤU TRÚC

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;
}
```

CHƯƠNG 7: KIỂU CẤU TRÚC

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));
 (*V).x=2;        // V→x = 2;
 (*V).y=5;        // V→y = 5;
  printf("(*V).x = %d , (*V).y = %d \n",(*V).x, (*V).y );
 return 0;
}
```

CHƯƠNG 7: KIỂU CẤU TRÚC

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("(*V).x = %d , (*V).y = %d \n",(*V).x, (*V).y );
 return 0;
}
```

CHƯƠNG 7: KIỂU CẤU TRÚC

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 );
}
```

CHƯƠNG 7: KIỂU CẤU TRÚC

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