

Chapter 12
Structure



Structures

- Collect related variables (members) under one name
  - Example: Maintain the information of a student such as
    - ID
    - Name
    - Grade
    - Rank
- Contain members of different data types
- Create link lists by combining with pointers



Structure Definitions

```
struct struct-varname {
        data-type member-name-1;
        data-type member-name-2;
        ...
        data-type member-name-n;
};
```

Example

```
struct student {
        int     id;
        char    name[256];
        float   avggrade;
        int     rank;
};
```

int id
char name[256]
float avggrade
int rank

4 members



Declare Structure Variables (I)

struct struct-var-name var-1, var-2, ..., var-k;

Example

```
struct student {
    int id;
    char name[256];
    float avggrade;
    int rank;
};

int main() {
    struct student tom, amy, paul; /* three students */
    struct student studentarr[50]; /* an array of 50 students */
    return 0;
}
```

Declare Structure Variables (II)

- Declare structure variable when defining the structure

```
struct struct-var-name {
        data-type member-name-1;
        data-type member-name-2;
        ...
        data-type member-name-n;
} var-1, var-2, ..., var-k;
```

Example

```
int main() {
        struct student {
                int     id;
                char    name[256];
                float   avggrade;
                int     rank;
        } tom, amy, paul;
        return 0;
}
```



Accessing Members of Structures

struct-var-name.member-name

Example

```
struct student {
    int id;
    char name[256];
    float avggrade;
    int rank;
};
int main() {
    struct student a;
    a.id = 1;
    strcpy(a.name, "tom");
    a.avggrade = 88.9;
    a.rank = 2;
    printf("id %d, name %s, grade %f, rank %d\n",
         a.id, a.name, a.avggrade, a.rank);
    return 0;
}
```

Accessing Members of Structure Array

struct-var-name[index].member-name

Example

```
struct student {
    int id;
    char name[256];
    float avggrade;
    int rank;
};
int main() {
    struct student a[2];
    a[0].id = 1;
    strcpy(a[0].name, "tom");
    a[0].avggrade = 88.9;
    a[0].rank = 2;
    printf("id %d, name %s, grade %f, rank %d\n",
         a[0].id, a[0].name, a[0].avggrade, a[0].rank);
    return 0;
}
```



Example

```
/* OUTPUT---
請輸入姓名: Tom Lee
請輸入成績: 89
姓名:Tom Lee
成績:89
--------------------*/
```

```
01  /* 結構變數的輸入與輸出 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     struct data          /* 定義結構 data */
07     {
08        char name[10];
09        int math;
10     } student;           /* 宣告 data 型態的結構變數 student */
11     printf("請輸入姓名: ");
12     gets(student.name);                /* 輸入學生姓名 */
13     printf("請輸入成績 :");
14     scanf("%d",&student.math);         /* 輸入學生成績 */
15     printf("姓名:%s\n", student.name);
16     printf("成績:%d\n", student.math);
17     system("pause");
18     return 0;
19  }
```

Using sizeof()

- Using sizeof() to calculate the size of struct.

```
01  /* 結構的大小 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     struct data    /* 定義結構 */
07     {
08        char name[10];
09        int math;
10     } student;
11     printf("sizeof(student)=%d\n",sizeof(student));
12
13     system("pause");
14     return 0;
15  }
```

```
/* OUTPUT--

sizeof(student)=16
----------------------*/
```

Alignment to 4



Structure Initialization

```
struct student {
    int id;
    char name[256];
    float avggrade;
    int rank;
};

int main() {
    struct student a = {1, "tom", 88.9, 2};
    struct student b = a;

    printf("id %d, name %s, grade %f, rank %d\n",
         b.id, b.name, b.avggrade, b.rank);
    return 0;
}
```



Structure Initialization (Cont.)

- Type 1:

```
struct data      /* 定義結構 data */
{
   char name[10];
   int math;
};
struct data student={"Jenny",78};   /* 設定結構變數的初值 */
```

- Type 2:

```
struct data      /* 定義結構 data */
{
   char name[10];
   int math;
} student={"Jenny",78};   /* 宣告結構變數,並設定初值 */
```



Structure Initialization: Example

```
/* OUTPUT--
學生姓名: Mary Wang
數學成績: 74
----------------------*/
```

```
01  /* 結構變數的初值設定 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     struct data    /* 定義結構 data */
07     {
08        char name[10];
09        int math;
10     };
11     struct data student={"Mary Wang",74};  /* 設定結構變數初值 */
12     printf("學生姓名: %s\n",student.name);
13     printf("數學成績: %d\n",student.math);
14
15     system("pause");
16     return 0;
17  }
```



Structure Initialization: Example (Cont.)

- Assign value of one struct variable to another.

```
01  /* 結構的設值 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     struct data
07     {
08        char name[10];
09        int math;
10     } s1={"Lily Chen",83};        /* 宣告結構變數 s1，並設定初值 */
11     struct data s2;               /* 宣告結構變數 s2 */
12     s2=s1;                      /* 把結構變數 s1 的值設定給結構變數 s2 */
13     printf("s1.name=%s, s1.math=%d\n",s1.name,s1.math);
14     printf("s2.name=%s, s2.math=%d\n",s2.name,s2.math);
15     system("pause");
16     return 0;
17  }
```

```
/* OUTPUT----------

s1.name=Lily Chen, s1.math=83
s2.name=Lily Chen, s2.math=83
---------------------------*/
```

Nested Structure

- A structure contains other structures

```
struct struct-varname1 {
        data-type member-name-1;
        data-type member-name-2;
        ...
        data-type member-name-n;
};
struct struct-varname2 {
        data-type member-name-1;
        struct struct-var-name1 member-name-n;
};
```



Example: Nested Structure

```
struct contact {
    char email[50];
    char phone[50];
};
struct student {
    int id;
    char name[256];
    float avggrade;
    struct contact info;
};
int main() {
    struct student a = {1, "tom", 88.9, {"test@gmail.com", "22334444"}};

    printf("id %d, name %s, grade %f, email %s, phone %s\n",
         a.id, a.name, a.avggrade, a.info.email, a.info.phone);
    return 0;
}
```



Example: Nested Structure (Cont.)

```
01  /* 巢狀結構的使用 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     struct date          /* 定義結構 date */
07     {
08        int month;
09        int day;
10     };
11     struct data          /* 定義巢狀結構 data */
12     {
13        char name[10];
14        int math;
15        struct date birthday;
16     } s1={"Mary Wang",74,{10,2}};   /* 設定結構變數 s1 的初值 */
17     printf("學生姓名: %s\n",s1.name);
18     printf("生日: %d月%d日\n",s1.birthday.month,s1.birthday.day);
19     printf("數學成績: %d\n",s1.math);
20
21     system("pause");
22     return 0;
23  }
```

```
/* OUTPUT---

學生姓名: Mary Wang
生日: 10月2日
數學成績: 74
----------------------*/
```



Structure Array

Declaration

```
struct structure-type structure-var-name[Num];
```

```
struct data s1[10];              /* 宣告結構陣列 s1 */
s1[2].math=12;                   /* 設定 s1[2].math=12 */
strcpy(s1[2].name,"Peggy");      /* 設定 s1[2].name 的值為"Peggy" */
```



sizeof() for Structure Array

```
01  /* 結構陣列的大小 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     struct data          /* 定義結構 */
07     {
08        char name[10];
09        int math;
10     }student[10];
11
12     printf("sizeof(student[3])=%d\n",sizeof(student[3]));
13     printf("sizeof(student)=%d\n",sizeof(student));
14     system("pause");
15     return 0;
16  }
```

```
/* OUTPUT----

sizeof(student[3])=16
sizeof(student)=160
-----------------------*/
```



sizeof() for Structure Array (Cont.)

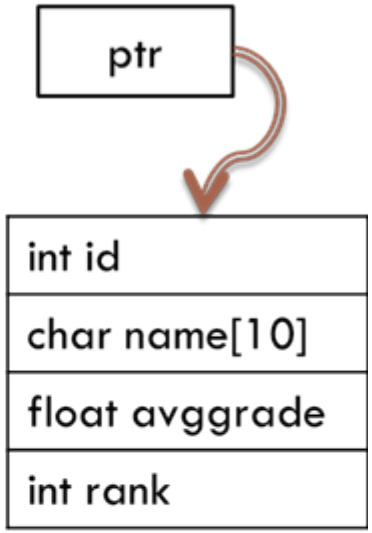
```
01  /* 結構陣列的使用 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  #define MAX 2
05  int main(void)
06  {
07     int i;
08     struct data
09     {
10        char name[10];
11        int math;
12     } student[MAX];          /* 宣告結構陣列 student */
13     for(i=0;i<MAX;i++)
14     {
15        printf("學生姓名: ");
16        gets(student[i].name);              /* 輸入學生姓名 */
17        printf("數學成績: ");
18        scanf("%d",&student[i].math);       /* 輸入學生數學成績 */
19        fflush(stdin);                      /* 清空緩衝區內的資料 */
20     }
21     for(i=0;i<MAX;i++)                     /* 輸出結構陣列的內容 */
22        printf("%s 的數學成績=%d\n",student[i].name,student[i].math);
23     system("pause");
24     return 0;
25  }
```

```
/* OUTPUT--

學生姓名: Jenny
數學成績: 65
學生姓名: Teresa
數學成績: 88
Jenny 的數學成績=65
Teresa 的數學成績=88
----------------------*/
```



Structure Pointer

```
struct student {
    int id;
    char name[256];
    float avggrade;
    int rank;
};

int main() {
    struct student a = {1, "tom", 88.9, 2};
    struct student *ptr = &a;
    ptr->rank = 3;
    printf("id %d, name %s, grade %f, rank %d\n",
         ptr->id, ptr->name, ptr->avggrade, ptr->rank);
    return 0;
}
```



Access member by pointer

```
struct-pointer->member-name
```



Structure Pointer (Cont.)

```
01  /* 使用指向結構的指標 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     struct data    /* 定義結構 */
07     {
08        char name[10];
09        int math;
10        int eng;
11     } student,*ptr;     /* 宣告結構變數 student 及指向結構的指標 ptr */
12     ptr=&student;          /* 將 ptr 指向結構變數 student 的位址 */
13     printf("學生姓名: ");
14     gets(ptr->name);       /* 輸入字串給 student 的 name 成員存放 */
15     printf("數學成績: ");
16     scanf("%d",&ptr->math); /* 輸入整數給 student 的 math 成員存放*/
17     printf("英文成績: ");
18     scanf("%d",&ptr->eng); /* 輸入整數給 student 的 eng 成員存放*/
19     printf("數學成績=%d, ",ptr->math);
20     printf("英文成績=%d, ",ptr->eng);
21     printf("平均分數=%.2f\n",(ptr->math + ptr->eng)/2.0);
22     system("pause");
23     return 0;
24  }
```

```
/* OUTPUT-----------------

學生姓名: Jenny
數學成績: 78
英文成績: 89
數學成績=78, 英文成績=89, 平均分數=83.50
---------------------------------*/
```

&ptr->eng = (*ptr).eng



Using Pointer for Structure Array

Pointer pointing to structure array

```
(StructureName+i)->Member;
```

```
01  /* 以指標來表示結構陣列 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  #define MAX 3
05  int main(void)
06  {
07     int i,m,index=0;
08     struct data              /* 定義結構 data */
09     {
10        char name[10];
11        int math;
12     } student[MAX]={{"Mary",87},{"Flora",93},{"Jenny",74}};
13
```



Using Pointer for Structure Array (Cont.)

```
14      m=student->math;          /* 將 m 設值為 student[0].math */
15      for(i=1;i<MAX;i++)        /* 輸出結構陣列的內容 */
16      {
17         if((student+i)->math > m)
18         {
19            m=(student+i)->math;
20            index=i;
21         }
22      }
23      printf("%s 的成績最高, ",(student+index)->name);
24      printf("分數為%d 分\n",(student+index)->math);
25      system("pause");
26      return 0;
27  }
```

```
/* OUTPUT ------

Flora 的成績最高, 分數為 93 分
--------------------------*/
```



Passing Structure to Function

```
return-type function-name (struct struct-name var-name)
{
        /* function body */
}
```



Example

```
struct student {
    int id;
    char name[256];
    float avggrade;
    int rank;
};

void display (struct student stu) {
    printf("id %d, name %s, grade %f, rank %d\n",
         stu.id, stu.name, stu.avggrade, stu.rank);

}

void main() {
    struct student stu[2] = {{1, "tom", 88, 2}, {2, "amy", 92, 1}};
    int i;
    for (i = 0; i < 2; i++)
         display(stu[i]);
}
```

output

```
id 1, name tom, grade 88.000000, rank 2
id 2, name amy, grade 92.000000, rank 1
```



Passing Structure: Call by Value

```
struct student {
    int id;
    char name[256];
    float avggrade;
    int rank;
};

void modify (struct student stu) {
    stu.rank = 0;
}

void main() {
    struct student stu[2] = {{1, "tom", 88, 2}, {2, "amy", 92, 1} };
    int i;
    for (i = 0; i < 2; i++) {
        modify(stu[i]);
        printf("id %d, name %s, grade %f, rank %d\n",
                stu[i].id, stu[i].name, stu[i].avggrade, stu[i].rank);
    }
}
```

output

```
id 1, name tom, grade 88.000000, rank 2
id 2, name amy, grade 92.000000, rank 1
```



Passing Structure: Call by Value (Cont)

```
/* OUTPUT---
學生姓名: Jenny
數學成績: 74
----------------------*/
```

```
01  /* 傳遞結構到函數裡 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  struct data
05  {
07     char name[10];
07     int math;
08  };
09  void display(struct data);          /* 宣告函數 display()的原型 */
10  int main(void)
11  {
12     struct data s1={"Jenny",74};    /* 設定結構變數 s1 的初值 */
13     display(s1);          /* 呼叫函數 display()，傳入結構變數 s1 */
14     system("pause");
15     return 0;
16  }
17  void display(struct data st)       /* 定義 display()函數 */
18  {
19     printf("學生姓名: %s\n",st.name);
20     printf("數學成績: %d\n",st.math);
21  }
```

將結構 data 定義在 main() 的外部，這個結構就成了全域的結構

Passing Structure: Call by Address

```
struct student {
    int id;
    char name[256];
    float avggrade;
    int rank;
};

void modify (struct student *stu) {
    stu->rank = 0;
}

void main() {
    struct student stu[2] = {{1, "tom", 88, 2}, {2, "amy", 92, 1} };
    int i;
    for (i = 0; i < 2; i++) {
        modify(&stu[i]);
        printf("id %d, name %s, grade %f, rank %d\n",
                stu[i].id, stu[i].name, stu[i].avggrade, stu[i].rank);
    }
}
```

output

```
id 1, name tom, grade 88.000000, rank 0
id 2, name amy, grade 92.000000, rank 0
```



Passing Structure: Call by Address (Cont.)

```
01  /* 傳遞結構的位址到函數裡 */
02  #include <stdio.h>
03  #include <stdlib.h>
04
05  struct data      /* 定義全域的結構 data */
06  {
07     char name[10];
08     int math;
09  };
10  void swap(struct data *,struct data *);   /* swap()的原型 */
11
12  int main(void)
13  {
14     struct data s1={"Jenny",74};     /* 宣告結構變數 s1，並設定初值 */
15     struct data s2={"Teresa",88};    /* 宣告結構變數 s2，並設定初值 */
16
```



Passing Structure: Call by Address (Cont.)

```
17      swap(&s1,&s2);          /* 呼叫 swap()函數 */
18      printf("呼叫 swap()函數後:\n");
19      printf("s1.name=%s, s1.math=%d\n",s1.name,s1.math);
20      printf("s2.name=%s, s2.math=%d\n",s2.name,s2.math);
21
22      system("pause");
23      return 0;
24  }
25  void swap(struct data *p1,struct data *p2)
26  {
27      struct data tmp;
28      tmp=*p1;
29      *p1=*p2;
30      *p2=tmp;
31  }
```

```
/*  OUTPUT -------
呼叫 swap()函數後:
s1.name=Teresa, s1.math=88
s2.name=Jenny, s2.math=74
-------------------------*/
```



Passing Structure Array: Call by Address

```
01  /* 傳遞結構陣列 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  #define MAX 3
05
06  struct data           /* 定義全域的結構 data */
07  {
08     char name[10];
09     int math;
10  };
11  int maximum(struct data arr[]);    /* 宣告函數 maximum()的原型 */
12  int main(void)
13  {
14     int idx;
15     struct data s1[MAX]={{"Mary",87},{"Flora",93},{"Jenny",74}};
16
```

Passing Structure Array: Call by Address (Cont.)

```
17      idx=maximum(s1);    /* 呼叫 maximum()函數 */
18      printf("%s 的成績最高, ",(s1+idx)->name);    /* 印出最高分的姓名 */
19      printf("分數為%d 分\n",(s1+idx)->math);    /* 印出最高分的成績 */
20
21      system("pause");
22      return 0;
23  }
24  int maximum(struct data arr[])     /* maximum()函數的定義 */
25  {
26      int m,i,index;
27      m=arr->math;              /* 將 m 設值為 arr[0].math */
28      for(i=0;i<MAX;i++)
29        if((arr+i)->math>m)
30        {
31            m=(arr+i)->math;
32            index=i;
33        }
34      return index;             /* 傳回陣列的索引值 */
35  }
```

```
/* OUTPUT ------
Flora 的成績最高, 分數為 93 分
--------------------------*/
```

Enumeration

- User-defined type consists of a set of enumerators (named integer constant)

```
enum enum-type-name
{
    enumerator-1,
    enumerator-2,
    ...
    enumerator-n
};

/* variable decleration*/
enum enum-type-name var-1, var-2,…, var-k;
```



Enumeration Definition

Definition 1

```
enum color
{
        red,
        green,
        purple
};
enum color apple, grape;
```

Default = 0
Default = 1
Default = 2

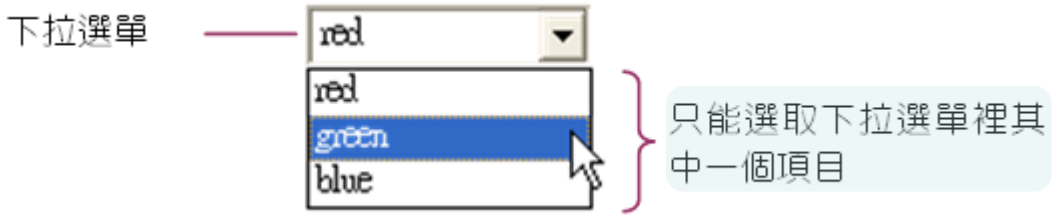
Definition 2

```
enum color
{
        red,
        green,
        purple
} apple, grape;
```

Declare variables right after enum definition

Enumeration Definition (Cont.)

- 下拉選單的設計非常類似於列舉型態：




```
enum color
{
    red,
    blue,
    green
} shirt,hat;
```

shirt與hat的值只可以是 red, green 與 blue 其中之一，不能為其它的值



Enumeration Example

```
01  /* 列舉型態的使用 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     enum color  /* 定義列舉型態 color */
07     {
08        red,
09        green,
10        blue
11     };
12     enum color shirt;  /* 宣告列舉型態的變數 shirt */
13     printf("sizeof(shirt)=%d\n",sizeof(shirt));
14     printf("red=%d\n",red);
15     printf("green=%d\n",green);
16     printf("blue=%d\n",blue);
17     shirt=green;        /* 將 shirt 的值設為 green */
18     if(shirt==green)
19        printf("您選擇了綠色的衣服\n");
20     else
21        printf("您選擇了非綠色的衣服\n");
22     system("pause");
23     return 0;
24  }
```

```
/* OUTPUT---

sizeof(shirt)=4
red=0
green=1
blue=2
您選擇了綠色的衣服
----------------------*/
```

Enumeration Value

- Enumerated value could start from other integers:

```
enum color
{
    red=5,
    green,
    blue
};
```

預設值設為 5

```
/* 設定 red 的值為 5 */
/* green 的預設值為 6 */
/* blue 的預設值為 7 */
```

預設值變成6

預設值變成7

```
enum color
{
    red=10,
    green=20,
    blue=30
}shirt=blue;
```

預設值設為 10

```
/* 設定 red 的值為 10 */
/* 設定 green 的值為 20 */
/* 設定 blue 的值為 30 */
```

預設值設為 20

預設值設為 30



Enumeration Example (Cont.)

```
01  /* 列舉型態的使用範例 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     char key;       /* 用來儲存按鍵的資訊 */
07     enum color      /* 定義列舉型態 color */
08     {
09        red=114,     /* 將列舉常數 red 設定為 114，即字母 r 的 ASCII 碼 */
10        green=103,   /* 將列舉常數 green 設定為 103 (g 的 ASCII 碼) */
11        blue=98      /* 將列舉常數 blue 設定為 98 (b 的 ASCII 碼) */
12     } shirt;        /* 宣告列舉型態的變數 shirt */
13
14     do
15     {
16        printf("請輸入 r,g 或 b: ");
17        scanf("%c",&key);     /* 讀入一個字元 */
18        fflush(stdin);        /* 清空緩衝區內的資料 */
19     } while((key!=red)&&(key!=green)&&(key!=blue));
20
```



Enumeration Example (Cont.)

```
21      shirt=key;                  /* 將 key 的值指定給 shirt 變數存放 */
22
23      switch(shirt)               /* 根據 shirt 的值印出字串 */
24      {
25         case red:
26            printf("您選擇了紅色\n");
27            break;
28         case green:
29             printf("您選擇了綠色\n");
30            break;
31         case blue:
32             printf("您選擇了藍色\n");
33            break;
34      }
35      system("pause");
36      return 0;
37  }
```

```
/* OUTPUT --

請輸入 r,g 或 b: h
請輸入 r,g 或 b: k
請輸入 r,g 或 b: b
您選擇了藍色
----------------------*/
```



typedef

- Create synonyms (aliases) for previously defined data types
- Use typedef to create shorter type names
- Use typedef to enhance the program's readability.

```
typedef data-type alias;
```

- Example

```
typedef int studentid;
studentid a, b, c;   // studentid = int

typedef struct student Student;
Student stu1, stu2;
```



Example: typedef

```
struct student {
    int id;
    char name[256];
    float avggrade;
    int rank;
};

typedef struct student Student;

void main() {
    Student a = {1, "tom", 88.9, 2};
    printf("id %d, name %s, grade %f, rank %d\n",
        a.id, a.name, a.avggrade, a.rank);
}
```



typedef for Structure

```
struct student {
    int id;
    char name[20];
};

typedef struct student STUDENT;
Student stu1, stu2;
```

```
typedef struct {
    int id;
    char name[20];
} STUDENT;

Student stu1, stu2;
```



Example

```
01  /* 利用 typedef 來定義資料型態 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  struct data
05  {
06     char name[10];
07     int math;
08  };
09  typedef struct data SCORE;      /* 把 struct data 定義成新的型態 */
10  void display(SCORE);            /* 宣告函數 display() 的原型 */
11  int main(void)
12  {
13     SCORE s1={"Jenny",74};       /* 設定結構變數 s1 的初值 */
14     display(s1);                 /* 呼叫 display()，傳入結構變數 s1 */
15     system("pause");
16     return 0;
17  }
18  void display(SCORE st)          /* 定義函數 display()*/
19  {
20     printf("學生姓名: %s\n",st.name);
21     printf("數學成績: %d\n",st.math);
22  }
```

```
/* OUTPUT---
學生姓名: Jenny
數學成績: 74
--------------------*/
```

Lab 12-1

- Declare a structure array with size 3
  - The structure contains the student id (int), student name (string), math grade (int)
  - Let the user input the information of three students
  - Compute the average grade (float)
  - Rank the student by grades
    - Print out the average grade, and print the sorted student names in descending order
    - ***Output:***
      Average grade: xxx.xxx
      Rank: student_B, student_C, student_A



Lab 12-2

- 試撰寫一程式，使其能夠完成下列功能：
- 建立一時間結構time，其成員包括hour（小時）、minutes（分）及second（秒），其中hour與minutes的型態皆為int，而second的型態則為double。
- 宣告一個結構time型態的變數start，並設定初值為 {12, 32, 23.45}。
- 宣告一個結構time型態的變數end，並設定初值為 {22, 43, 23.44}。
- 以hh:mm:ss.ss的格式印出結構start與end的值。hh代表小時，佔有2格；mm代表分，佔有2格；ss.ss代表秒，其中秒數部分，整數與小數部分均取兩位。例如05:19:20.43代表了5小時19分20.43秒。
- 試計算從start開始，到end結束為止，總共經歷了多少時間，請把經歷的時間用另一個結構變數elapse來儲存，並以hh:mm:ss.ss的格式列印出來。