結構與其他資料型態

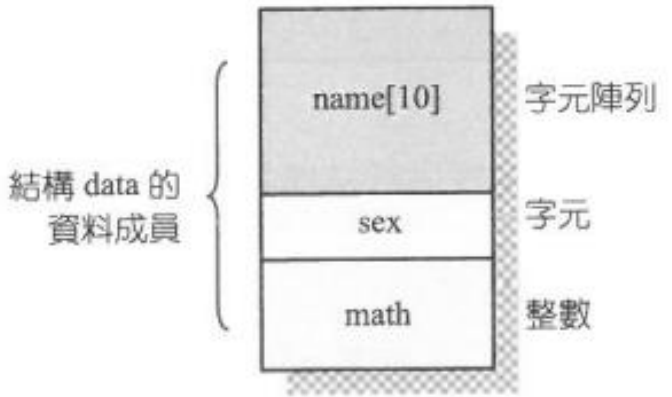
結構

- 結構 (structure)
  - 可把不同的資料型態組合再一起
  - 語法

```
struct 結構名稱          /* 定義結構 */
{
      資料型態 成員名稱 1;
      資料型態 成員名稱 2;
         ...
      資料型態 成員名稱 n;
};
```

宣告結構的成員

```
struct data
{
   char name[10];
   char sex;
   int math;
};
```



  - 宣告結構變數

```
struct 結構名稱 變數 1, 變數 2,..., 變數 n;
```

```
struct data student1, student2;     /* 宣告 data 型態的結構變數 */
```

結構宣告

```
struct 結構名稱                    /* 定義結構 */
{
    資料型態 成員名稱 1;
    資料型態 成員名稱 2;
            ...                    } 宣告結構的成員
    資料型態 成員名稱 n;
}結構變數 1,結構變數 2,..., 結構變數 n;    /* 宣告結構變數 */
```

```
struct data                        /* 定義 data 結構*/
{
   char name[10];
   char sex;
   int math;
}student1,student2;                /* 宣告結構變數 student1 與 student2 */
```

結構變數的使用及初值的設定

- 結構成員存取運算子「.」(dot operator)
  - 語法

```
結構變數名稱.成員名稱;
```

如 student1.name、student1.sex 及 student1.math 等。

範例-結構變數的輸入與輸出

```c
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    struct data       /* 定義結構data */
    {
        char name[10];
        int math;
    } student;        /* 宣告data型態的結構變數student */

    printf("請輸入姓名: ");
    gets(student.name);                  /* 輸入學生姓名 */
    printf("請輸入成績 :");
    scanf("%d",&student.math);    /* 輸入學生成績 */

    printf("姓名:%s\n", student.name);
    printf("成績:%d\n", student.math);

    system("pause");
    return 0;
}
```

範例-結構的大小

```c
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    struct data  /* 定義結構 */
    {
        char name[10];
        int math;
    } student;
    printf("sizeof(student)=%d\n",sizeof(student));

    system("pause");
    return 0;
}
```

範例詳解

- 字元陣列name佔10的位元組，整數變數nath佔有4個位元組，但sizeof出來的結果是16個位元組，而不是10+4=14位元組
- 編譯器會以結構成員裡，所佔的位元組最多的資料型別為單位來配置記憶體空間
  - 如範例而言,char只佔1位元組，而int佔了4位元組，所以基本單位是4位元組，因此結構變數所佔的位元組必須是4個倍數，而student結構變數裡的成員雖然只佔14位元組，但編譯器卻是配置16位元組

結構變數的初值設定

- 語法

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

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

範例-結構變數的初值設定

```c
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    struct data  /* 定義結構data */
    {
        char name[10];
        int math;
    };
    struct data student={"Mary Wang",74};  /* 設定結構變數初值 */

    printf("學生姓名: %s\n",student.name);
    printf("數學成績: %d\n",student.math);

    system("pause");
    return 0;
}
```

範例-結構變數的值複製給另一個結構變數

```c
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    struct data
    {
        char name[10];
        int math;
    } s1={"Lily Chen",83};          /* 宣告結構變數s1，並設定初值 */
    struct data s2;                 /* 宣告結構變數s2 */
    s2=s1;                          /* 把結構變數s1的值設定給結構變數s2 */

    printf("s1.name=%s, s1.math=%d\n",s1.name,s1.math);
    printf("s2.name=%s, s2.math=%d\n",s2.name,s2.math);

    system("pause");
    return 0;
}
```

巢狀結構

- 結構裡在包含結構

```
struct 結構 1
{
    /* 結構 1 的成員 */
};
struct 結構 2
{
    /* 結構 2 的成員 */
    struct 結構 1 變數名稱
};
```

定義巢狀結構

範例-巢狀結構

```c
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    struct date       /* 定義結構date */
    {
        int month;
        int day;
    };
    struct data       /* 定義巢狀結構data */
    {
        char name[10];
        int math;
        struct date birthday;
    } s1={"Mary Wang",74,{10,2}};    /* 設定結構變數s1的初值 */

    printf("學生姓名: %s\n",s1.name);
    printf("生日: %d月%d日\n",s1.birthday.month,s1.birthday.day);
    printf("數學成績: %d\n",s1.math);

    system("pause");
    return 0;
}
```

結構陣列

- 語法

```
struct 結構型態 結構陣列名稱[元素個數];
```

```
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" */
```

範例-結構陣列的大小

```c
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    struct data        /* 定義結構 */
    {
        char name[10];
        int math;
    }student[10];

    printf("sizeof(student[3])=%d\n",sizeof(student[3]));
    printf("sizeof(student)=%d\n",sizeof(student));
    system("pause");
    return 0;
}
```

範例-結構陣列的使用

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 2
int main(void)
{
    int i;
    struct data          /* 定義結構data */
    {
        char name[10];
        int math;
    } student[MAX];      /* 宣告結構陣列student */

    for(i=0;i<MAX;i++)
    {
        printf("學生姓名: ");
        gets(student[i].name);            /* 輸入學生姓名 */
        printf("數學成績: ");
        scanf("%d",&student[i].math);     /* 輸入學生數學成績 */
        fflush(stdin);                      /* 清空緩衝區內的資料 */
    }

    for(i=0;i<MAX;i++)                    /* 輸出結構陣列的內容 */
        printf("%s的數學成績=%d\n",student[i].name,student[i].math);

    system("pause");
    return 0;
}
```

指向結構的指標

- 語法

```
struct 結構型態 *結構指標名稱;
```

```
struct data              /* 定義結構 data */
{
   char name[10];
   int math;
}student;                /* 宣告結構 data 型態之變數 student */
```

```
struct data *ptr;              /* 宣告指向結構 data 型態之指標 ptr */
```

```
ptr=&student;                  /* 將指標 ptr 指向結構變數 student */
```

- 使用指標存取結構變數的成員時要使用「->」來連接欲存取的成員

```
結構指標名稱->結構變數成員;
```

範例-使用指向結構的指標

```c
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    struct data  /* 定義結構 */
    {
        char name[10];
        int math;
        int eng;
    } student,*ptr;  /* 宣告結構變數student及指向結構的指標ptr */

    ptr=&student;              /* 將ptr指向結構變數student的位址 */
    printf("學生姓名: ");
    gets(ptr->name);           /* 輸入字串給student的name成員存放 */
    printf("數學成績: ");
    scanf("%d",&ptr->math);    /* 輸入整數給student的math成員存放*/
    printf("英文成績: ");
    scanf("%d",&ptr->eng); /* 輸入整數給student的eng成員存放*/

    printf("數學成績=%d, ",ptr->math);
    printf("英文成績=%d, ",ptr->eng);
    printf("平均分數=%.2f\n",(ptr->math + ptr->eng)/2.0);
    system("pause");
    return 0;
}
```

範例-以指標來表示結構陣列

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 3
int main(void)
{
    int i,m,index=0;
    struct data           /* 定義結構data */
    {
        char name[10];
        int math;
    } student[MAX]={{"Mary",87},{"Flora",93},{"Jenny",74}};

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

將結構傳遞到函數

- 結構以傳值(call by value)的方式傳遞到函數
- 語法

```
傳回值型態 函數名稱(struct 結構名稱 變數名稱)
{
    /* 函數的定義 */
}
```

範例-傳遞結構到函數

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

struct data
{
    char name[10];
    int math;
};

void display(struct data);      /* 宣告函數display()的原型 */

int main(void)
{
    struct data s1={"Jenny",74};    /* 設定結構變數s1的初值 */
    display(s1);      /* 呼叫函數display()，傳入結構變數s1 */

    system("pause");
    return 0;
}

void display(struct data st)      /* 定義display()函數 */
{
    printf("學生姓名: %s\n",st.name);
    printf("數學成績: %d\n",st.math);
}
```

範例-傳遞結構的位址

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

struct data     /* 定義全域的結構data */
{
   char name[10];
   int math;
};
void swap(struct data *,struct data *);    /* swap()的原型 */

int main(void)
{
   struct data s1={"Jenny",74}; /* 宣告結構變數s1，並設定初值 */
   struct data s2={"Teresa",88};    /* 宣告結構變數s2，並設定初值 */

   swap(&s1,&s2);          /* 呼叫swap()函數 */
   printf("呼叫swap()函數後:\n");
   printf("s1.name=%s, s1.math=%d\n",s1.name,s1.math);
   printf("s2.name=%s, s2.math=%d\n",s2.name,s2.math);

   system("pause");
   return 0;
}
```

```c
void swap(struct data *p1,struct data *p2)
{
   struct data tmp;
   tmp=*p1;
   *p1=*p2;
   *p2=tmp;
}
```

範例-傳遞結構陣列

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 3

struct data          /* 定義全域的結構data */
{
   char name[10];
   int math;
};
int maximum(struct data arr[]); /* 宣告函數maximum()的原型 */
int main(void)
{
   int idx;
   struct data s1[MAX]={{"Mary",87},{"Flora",93},{"Jenny",74}};

   idx=maximum(s1);     /* 呼叫maximum()函數 */
   printf("%s的成績最高, ",(s1+idx)->name); /* 印出最高分的姓名 */
   printf("分數為%d分\n",(s1+idx)->math);   /* 印出最高分的成績 */

   system("pause");
   return 0;
}
```

```c
int maximum(struct data arr[])  /* maximum()函數的定義 */
{
   int m,i,index;
   m=arr->math;              /* 將m設值為arr[0].math */
   for(i=0;i<MAX;i++)
     if((arr+i)->math>m)
     {
        m=(arr+i)->math;
        index=i;
     }
   return index;             /* 傳回陣列的索引值 */
}
```

列舉型態

- 列舉型態(enumeration)
  - 特殊的常數定義方式
  - 可利用一組有意義的名稱來取代一組整數常數
  - 語法

```
enum 列舉型態名稱
{
    列舉常數 1,
    列舉常數 2,
      ...
    列舉常數 n
};
enum 列舉型態名稱 變數 1, 變數 2, ..., 變數 m;   /* 宣告變數 */
```

定義列舉型態

列舉型態宣告範例

```
enum color
{
    red,
    blue,
    green
};
enum color shirt, hat;    /* 宣告列舉型態 color 之變數 shirt 與 hat */
```

定義列舉型態 color

```
enum color          /* 宣告列舉型態 color */
{
    red,
    blue,
    green
} shirt,hat;        /* 定義完列舉型態後，便立即宣告變數 shirt 與 hat */
```

列舉常數

列舉型態的使用與初值設定

- 預設下red的值0、blue為1、green為2，且這些值不能再被更改

```
enum color        /* 定義列舉型態 color */
{
    red,
    blue,
    green
}shirt,hat;       /* 宣告列舉型態 color 的變數 shirt 與 hat */
```

範例-列舉型態的使用

```c
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    enum color        /* 定義列舉型態color */
    {
        red,
        green,
        blue
    };
    enum color shirt;     /* 宣告列舉型態的變數shirt */

    printf("sizeof(shirt)=%d\n",sizeof(shirt));
    printf("red=%d\n",red);
    printf("green=%d\n",green);
    printf("blue=%d\n",blue);

    shirt=green;          /* 將shirt的值設為green */
    if(shirt==green)
        printf("您選擇了綠色的衣服\n");
    else
        printf("您選擇了非綠色的衣服\n");

    system("pause");
    return 0;
}
```

列舉型態初值的設定

```
enum color          /* 定義列舉型態 color */
{
    red=5,          /* 設定 red 的值為 5 */
    green,          /* green 的預設值為 6 */
    blue            /* blue 的預設值為 7 */
};
```

```
enum color          /* 定義列舉型態 color */
{
    red=10,         /* 設定 red 的值為 10 */
    green=20,       /* 設定 green 的值為 20 */
    blue=30         /* 設定 blue 的值為 30 */
}shirt=blue;
```

範例-列舉型態的使用

```c
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    char key;          /* 用來儲存放鍵的資訊 */
    enum color    /* 定義列舉型態color */
    {
        red=114,   /* 將列舉常數red設定為114，即字母r的ASCII碼 */
        green=103,     /* 將列舉常數green設定為103 (g的ASCII碼) */
        blue=98    /* 將列舉常數blue設定為98 (b的ASCII碼) */
    } shirt;       /* 宣告列舉型態的變數shirt */

    do
    {
        printf("請輸入r,g或b: ");
        scanf("%c",&key); /* 鍵入一個字元 */
        fflush(stdin);          /* 清空緩衝區內的資料 */
    } while((key!=red)&&(key!=green)&&(key!=blue));
    shirt=key;             /* 將key的值指定給shirt變數存放 */

    switch(shirt)              /* 根據shirt的值印出字串 */
    {
        case red:
            printf("您選擇了紅色\n");
            break;
        case green:
            printf("您選擇了綠色\n");
            break;
        case blue:
            printf("您選擇了藍色\n");
            break;
    }
    system("pause");
    return 0;
}
```

使用自訂的型態 typedef

- typedef (type definition)
  - 將已經有的資料型態重新定義其識別名稱
  - 可以定義屬於自己的資料型態
  - 語法

```
typedef 資料型態 識別字;
```

```
typedef int clock;     /* 定義 clock 為整數型態 */
clock hour,second;     /* 宣告 hour,second 為 clock 型態 */
```

  - 作用範圍和一般變數的生命週期相同

範例-利用typedef定義資料型態

```c
#include <stdio.h>
#include <stdlib.h>
struct data
{
    char name[10];
    int math;
};
typedef struct data SCORE;  /* 把 struct data 定義成新的型態 */
void display(SCORE);            /* 宣告函數display()的原型 */
int main(void)
{
    SCORE s1={"Jenny",74};        /* 設定結構變數s1的初值 */
    display(s1);                /* 呼叫display()，傳入結構變數s1 */

    system("pause");
    return 0;
}
void display(SCORE st)        /* 定義函數display()*/
{
    printf("學生姓名: %s\n",st.name);
    printf("數學成績: %d\n",st.math);
}
```

範例詳解

```
typedef struct
{
    char name[10];
    int math
} SCORE;
```

新的資料型態名稱

習題

- 請依下列條件輸出以下畫面
  - 請使用結構date，其成員year、month、day型態皆為整數
  - 可由鍵盤輸入年月日並輸出「YYYY-MM-DD」日期格式，月和日要有補0效果