

Chapter 11.

Structures

Structure

- Structure: combine different data into a new data types

結構可將型態不同的資料合併成為新的型態

- Format is as below: 定義結構與宣告結構變數的格式如下：

定義結構與宣告結構變數的語法

```
struct structure_name
{
    data_type1 member1;
    data_type2 member2;
    ...
    data_type3 membern;
};
```

結構的成員

定義結構

```
structure_name variable1, variable2;
```

宣告結構變數

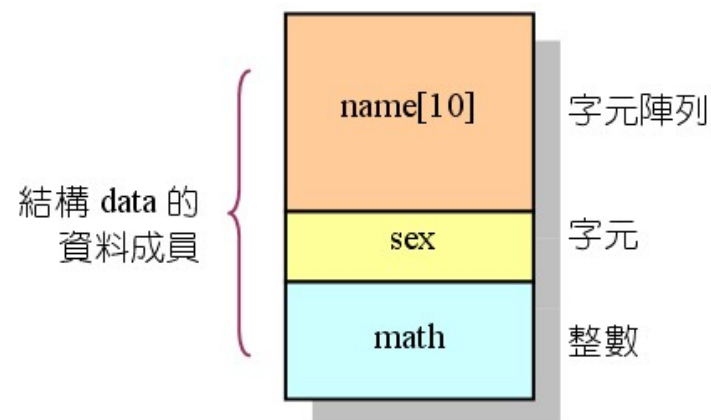
Structure - An Example



Structure Define:

```
struct STUDENT /* 定義 data 結構 */  
{  
    char name[10];  
    char sex;  
    int math;  
};
```

} 定義結構的成員



Declare Variables:

```
STUDENT    mary, tom; /* 宣告 data 型態的結構變數 */  
STUDENT    john;
```

Access Member data

```
struct data    /* 定義 data 結構*/  
{  
    char name[10];  
    char sex;  
    int math;  
};
```

} 定義結構的成員

存取結構變數的成員

結構變數名稱.成員名稱;

結構成員存取運算子

```
strcpy(mary.name, "Mary");    /* 設定 mary 的 name 成員為 "Mary" */  
mary.sex='F';                 /* 設定 sex 成員為 'F' */  
mary.math=95;                 /* 設定 math 成員為 95 */
```

An Example

/ prog11_1 OUTPUT---*

請輸入姓名: *Tom Lee*

請輸入成績: *89*

姓名: Tom Lee

成績: 89

-----/*

```

01  /* prog11_1, 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  }

```

結構變數所佔的記憶空間

/* prog11_2 OUTPUT--

sizeof(student)=16

-----*/

```
01  /* prog11_2, 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  }
```

結構變數初值的設定

/* prog11_3 OUTPUT--

學生姓名: Mary Wang

數學成績: 74

-----*/

```
01  /* prog11_3, 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  }
```

巢狀結構

- 結構內如有另一結構，則此結構稱為**巢狀結構**

巢狀結構的格式

```
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
};
```

結構 2 內包含有結構 1

結構陣列

- 下面為結構陣列的宣告格式：

結構陣列的宣告格式

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" */
```

結構陣列的範例

/* prog11_6 OUTPUT----

sizeof(student[3])=16

sizeof(student)=160

-----*/

```

01  /* prog11_6, 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  }

```

結構陣列的範例

```

01  /* prog11_7, 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  }

```

/* prog11_7 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

-----*/

Point to a Structure

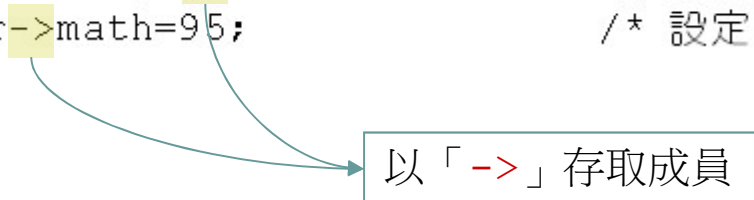
- Declare a **struct pointer** variable, the pointer point to a real struct data

```
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 */
```

- A struct pointer variable, using 「->」 to access its members :

```
strcpy(ptr->name, "Mary"); /* 設定 ptr 所指向之結構的 name 成員為 "Mary" */
ptr->math=95;              /* 設定 ptr 所指向之結構的 math 成員等於 95 */
```



以「->」存取成員

An Example

```

01  /* prog11_8, 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  }

```

/* prog11_8 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

-----*/

Structure as a function parameter

```

01  /* prog11_10, 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  }

```

學生姓名: Jenny
數學成績: 74
-----*/

將結構 **data** 定義在 **main()** 的外部, 這個結構就成了全域的結構

傳遞結構到函數的範例

/* prog11_10 OUTPUT---

學生姓名: Jenny

數學成績: 74

-----*/

```

01  /* prog11_10, 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

Structure Definitions

- Structure (結構): 把相關的變數資料放在一起，容易管理處理，結構裡面的變數稱為成員(member)

- 員工資料的結構

```
struct Employee {  
    char firstName[ 20 ];  
    char lastName[ 20 ];  
    int age;  
    char gender;  
};
```

struct 是關鍵字用來定義結構
Employee 為此結構的名稱
{ } 裡面是此結構的成員

- 結構為一種自訂的資料型態 (如int, float)，本身不配置記憶體，需再定義出變數(或稱物件)才有記憶體空間可以運作

```
struct Employee John;  
Employee Mary;
```

John, Mary才是變數，可以拿來用

Accessing Structure Members

■ 另一種直接定義結構變數的做法

```
struct Employee {  
    char firstName[ 20 ];  
    char lastName[ 20 ];  
    int age;  
    char gender;  
} John, Mary;
```

John . firstname

■ 存取結構成員

- ◆ structure member operator (.)—also called the dot operator

```
scanf("%s", John.firstname);  
printf("name: %s", John.firstname);
```

```
scanf("%d", &(Mary.age));  
printf("age %d", Mary.age);
```

Accessing Structure Members

pManager->firstname

■ 存取結構成員

- ◆ structure pointer operator (->)—also called the arrow operator.

```
Employee *pManager = &John;
```

```
scanf("%s", pManager->firstname);  
printf("name: %s", pManager->firstname);
```

```
scanf("%d", &(pManager->age));  
printf("age %d", pManager->age);  
printf("age %d", (*pManager).age);
```

■ Use sizeof() to get the size of a structure

結構記憶體大小不一定是所有成員的記憶體總和

- ◆ 可能是8 bytes的倍數，需用sizeof (Employee) 確認

Using Structures with Functions

■ Passing a pointer to a structure (call-by-address 傳指標)

```
struct Employee {
    char firstName[ 20 ];
    char lastName[ 20 ];
    int age;
    char gender;
};

void printdata(Employee *p);

int main()
{
    struct Employee John;
    struct Employee* Mary;

    Mary = &John;
    printf("input age: ");
    scanf("%d", &Mary->age);
    printf("age :%d \n", John.age);
    printf("%d\n", (*Mary).age);
```

```
printdata (&John) ;
printdata (Mary) ;

system("pause");
return 0;
}

void printdata(Employee *p)
{
    printf("age %d\n", p->age);
}
```

- 傳結構變數指標, 只有4 bytes
- p 指到主程式的 John 記憶體

Using Structures with Functions

- Structures may be passed to functions
 - ◆ passing an entire structure (call-by-value)
 - ◆ passing individual structure members (call-by-value)
 - ◆ passing a pointer to a structure
 - call-by-address, call-by-reference
- Call-by-value是整份的記憶體copy到function, 所以function有一份獨立的記憶體, 跟caller (主程式)的變數不會互相干擾, 但因為要copy整份記憶體, 執行速度稍慢
- Call-by-address, call-by-reference 只傳caller (主程式)的變數的指標, 只有4 bytes, 速度較快, 也跟caller (主程式)的變數共用記憶體

傳遞結構到函數的範例

call by value

/* prog11_10 OUTPUT---

學生姓名: Jenny

數學成績: 74

-----*/

```

01  /* prog11_10, 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

傳遞結構的位址 *call by address*

```


01  /* progl1_11, 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
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  }

```

Accessing Structure Members

```
struct Employee {
    char firstName[ 20 ];
    char lastName[ 20 ];
    int age;
    int salary;
};

int main()
{
    struct Employee John; //這兩種宣告都可以
    Employee Mary;
    Employee *p;           //pointer to some Employee

    //p 指到John
    //藉由 p修改John的age
}
```

Allocate/Delete Memory for structure pointer

```
struct Employee {
    char firstName[ 20 ];
    char lastName[ 20 ];
    int age;
    int salary;
};

int main()
{
    Employee *Mary;

    Mary = new Employee; //allocate memory to a new Employee
    Mary->age=30;
    Mary->salary=20000;
    delete Mary;
}
```


Allocate/Delete Memory for string

```
char *str;  
  
// Don't forget extra char for null character.  
str = new char[length + 1];  
  
.....  
  
// Done with str.  
delete [] str;
```

Enumeration Constants

- 自訂列舉型態，用來表示整數常數， keyword **enum**
- For example, the enumeration
 - ◆ **enum** months {**JAN**, **FEB**, **MAR**, **APR**, **MAY**, **JUN**, **JUL**, **AUG**, **SEP**, **OCT**, **NOV**, **DEC** };
 - ◆ creates a new type, **enum months**, in which the identifiers are set to the integers 0 to 11, respectively.
 - ◆ 除非特別指定，否則內定由0開始+1遞增
- To number the months 1 to 12
 - ◆ **enum** months {**JAN** = 1, **FEB**, **MAR**, **APR**, **MAY**, **JUN**, **JUL**, **AUG**, **SEP**, **OCT**, **NOV**, **DEC** };

```
enum months m;  
for (m=JAN; m<=DEC; m++) {  
    ...  
}
```

Preprocessor

- # 開頭的都是**preprocessor** 指令
- **preprocessor** 指令會先處理完，之後的程式碼再交由**compiler**進行編譯
- **preprocessor** 指令後面不接分號

Editor

Preprocessor

Compiler

Linker

#include Preprocessor Directive

- `#include <filename>`
 - ◆ Searches standard library for file
 - ◆ Use for standard library files
- `#include "filename"`
 - ◆ Searches current directory, then standard library
 - ◆ Use for user-defined files

#define Symbolic Constants

- 定義符號常數 Symbolic constants
 - ◆ When program compiled, all occurrences of symbolic constant replaced with replacement text
- Format
 - #define identifier replacement-text*
- Example:
 - #define PI 3.14159*
- Cannot redefine symbolic constants once they have been created

Conditional Compilation

- "Comment out" code
 - ◆ Cannot use `/* ... */`, or a lots of codes ...
 - ◆ Use

`#if 0`

code commented out

`#endif`

- ◆ To enable code, change 0 to 1

Conditional Compilation - Version Control

```
#define VERSION1
```

```
#ifdef VERSION1
```

```
    code for VERSION1
```

```
#else
```

```
    code for other versions
```

```
#endif
```

```
//#define VERSION1
```

```
#ifdef VERSION1
```

```
    code for VERSION1
```

```
#else
```

```
    code for other versions
```

```
#endif
```