撰寫 C 程式用到的核心語法

丁培毅

型態宣告與定義

`int i;`	整數變數
`int *j, k;`	j: 整數指標變數, k: 整數變數
`unsigned char *ch;`	ch: 無正負號字元變數的指標變數
`double f[10];`	10 個倍精準浮點數的陣列
`char nextChar(int, char*);`	2 個參數的函式
`int a[3][5][10];`	3 個元素的陣列, 每一個元素是 5 個子元素的陣列, 每一個子元素是 10 個整數的陣列
`int *func1(float);`	回傳整數指標的函式, 此函式接受單一浮點數參數
`int (*func2)(void);`	函數指標變數, 指到的函式不接受參數, 回傳整數值

輸入輸出

- #include <stdio.h>
- scanf(), getchar(), gets(), fscanf(), getc(), fgets(), sscanf()
  %d, %f, %lf, %c, %lld (%I64d), %s, %n, %[abc], space
- printf(), putchar(), puts(), fprintf(), fputc(), fputs(), sprintf()
  %d, %f, %c, %lld (%I64d), %s, other char
- FILE *fp = fopen("input.txt", "r");
  …fscanf(), fgets(), getc(), ftell(), fseek(), rewind() …
  fclose(fp);
- FILE *fp = fopen("output.txt", "w");
  …fprintf(), fputs(), putc(), fflush(), …
  fclose(fp);

函式

- Function definition
  ```
  return_type  func_name(func_parameters)
  {
     statements
  }
  ```
- Function call
  func_name(func_arguments) … used as a return_type expr
- Function prototype
  return_type func_name(func_parameters);
- Function pointer
  return_type (*func_ptr)(func_parameters);
- Function call with function pointer
  (*func_ptr)(func_arguments) … used as a return_type expr

條件判斷

- **if (**condition**)**
  compound_statement
- **if (**condition**)**
  compound_statement
  else
  compound_statement
- **if (**condition**)**
  compound_statement
  else if
  compound_statement
  else if
  compound_statement
  else
  compound_statement
- **switch (**int_value**)**
  {
  case value1**:**
  statements
  break;
  case value2**:**
  statements
  case value3:
  statements
  break;
  default:
  statements
  }
- condition **?** stmt1 **:** stmt2

迴圈

- **for (**initialization**;** looping_condition**;** update**)**
  compound_statement
- initialization
  while (looping_condition**)**
  compound_statement
- **do**
  compound_statement
  while (looping_condition**);**

consists of the update

陣列

- Definition
  type name[size];
- Usage
  name[index]   *(name+index)
- Multi-dimensional
  type name[size1][size2];

  typedef type newType[size2];
  newType name[size1];

結構

- Definition
  struct Name {
  type field1;
  type field2;
  …
  };
  struct Name sVar, sVar2, sArray[size], *sPtr = &sVar;
  (C++: Name sVar, sVar2, sArray[size], *sPtr = &sVar;)
  typedef struct Name NAME;
  NAME sVar, sVar2, sArray[size], *sPtr = &sVar;
- Usage
  sVar.field1   sPtr->field2   (*sPtr).field2
  sVar = sVar2;

指標

- Definition
```
type *ptr;
type *ptr_array[size];         // typedef type *PTR;
                               // PTR ptr_array[size];
type (*ptr_to_array)[size];    // typedef type ARY[size];
                               // ARY *ptr_to_array;
```
- Usage
```
type var, *ptr = &var;
*ptr is the same as var
struct {int x; double y} svar, *ptr = &svar;
*ptr is the same as svar
ptr->x is the same as (*ptr).x
```