第十週

王子銜, 陳毅軒, 吳尚龍

電機通訊程式設計

April 22, 2024

Outline

typedef

```
1 // 語法
2 typedef <資料型態> <名稱>;
```

typedef

- 將常用的資料型態組合給予一個比較直觀而易懂的別名。
- 擴充 C 原有的資料型態。
- 將較爲冗長複雜的資料型態轉換成較爲簡單的名稱。
- 由 C 語言的編譯器處理。

define

```
1 // 語法
2 #deifne <名稱> <內容>
```

define

- 創建固定的常數、條件編譯。
- 不限於類型別名。
- 是預處理器指令，用來替換文本。
- 在編譯器實際處理源代碼之前就指定的文本替換。

typedef vs define

```
1  // 舉例
2  typedef int* intp;
3  intp a,b;
4  // => int *a; int *b;
```

```
1  // 舉例
2  #define intp int*
3  intp a,b;
4  // int* a, b; => int *a; int b;
```

structure

structure

- 一種複合型別，用來表達由多個屬性組成的型別。
- 屬性可以是基本型別或是另一個複合型別所組成。
- 增加程式碼的可讀性。
- 可以讓資料更好分類與處理。

```
1  // 範例
2  struct student {
3    unsigned int id;
4    char name[10];
5    float weight;
6    int height;
7  };
```

structure 宣告

宣告 structure 變數，與一般變數宣告相似。
structure 的初始化有多種的方式。

```
1  // 宣告
2  struct student A;      //未初始化
3  struct student B={};   //初始化每個資料皆爲0
4  struct student C={1, "student C", 50, 160};//給予初始化
5  struct student D,E;
6  D = (struct student){2, "student D"};
7 //未指定的變數將初始化爲0，若使用{}宣告，需要指定型別，並且需要依照資料順序
     初始化
8  E = (struct student){
9    .id = 1,
10   .weight = 30.5
11 };//使用這種結構宣告可以不用依照順序宣告，並且未宣告的變數，初始化爲0
```

structure 存取

要存取裡面的變數使用" . " 存取。
structure 資料型態爲指標，需使用" -> " 存取。

```c
// 存取
  struct student A={1, "student A", 50, 160};
  A.id = 10;  //更改id
  printf("%s weight is %f", A.name, A.wright);
```

```c
// 存取
  struct student *A;
  A->id = 10;  //更改id
  printf("student A id is %d", A->id);
```

structure and typedef

```
1  // 宣告
2  typedef struct
3  {
4    unsigned int id;
5    char name[10];
6    float weight;
7    int height;
8  } Students;
9  Students A;//== struct student A;
```

structure 中包含 structure

```
1  // 宣告
2  typedef struct student
3  {
4    unsigned int id;
5    char name[10];
6    float weight;
7    int height;
8    struct student B;
9  } Students;
10 Students A;//== struct student A;
```

enum

enum

- 翻譯成列舉、枚舉。
- 將各種數值列表，並賦予該值有意義的名稱。
- 增加程式碼的可讀性。
- 沒有給予初始值預設爲 0。

enum

```
1 // 語法
2 enum <name> {
3 // enumerator-list
4 } <宣告變數>;
```

其中 < 宣告變數 > 不一定要填寫，可以之後再宣告變數。

```
1 // 語法
2 enum <name> <宣告變數>;
```

enum 範例

當宣告 enum 在全域時，在程式碼內，只要使用 enum 內定義的名稱，就會轉成 enum 所指示的常數。
workday 是 enum DAY 型別的資料，增加程式碼的可讀性和安全性。

```
1  // 範例
2  enum DAY
3  {
4      sunday,      // 0
5      monday,      // 1
6      tuesday,     // 2
7      wednesday,   // 3
8      thursday,    // 4
9      friday,      // 5
10     saturday    // 6
11 } workday;
```

```
1  // 範例
2  workday = saturday;
3  printf("workday %d", workday);
4  //workday 6
```

enum 範例

enum 裡面的常數預設從 0 開始，可以更改任意的起始值，或是其中常數的數值。
只要沒有定義的常數，會根據上一個常數遞增數值。

```
1  // 範例
2  enum DAY
3  {
4      sunday,      // 0
5      monday,      // 1
6      tuesday,     // 2
7      wednesday = 5,  // 5
8      thursday,    // 6
9      friday,      // 7
10     saturday    // 8
11 };
```

enum and typedef

- 可以使用 typedef 更改 enum 結構的名稱。
- weekday 爲重新定義 enum DAY 的名稱。
- 使用 typedef 可以不用寫 enum 的名稱。

```
1  // 範例
2    typedef enum DAY    // 可以不用定義名稱DAY
3    {
4        sunday,      // 0
5        monday,      // 1
6        ...
7    } weekday;
8
9    weekday workday = monday;
10   printf("workday %d",workday);
11   //workday 1
```

enum and typedef

使用 typedef 也可以先定義在宣告。

```
1  // 範例
2  typedef enum DAY weekday;
3  enum DAY   // 可以不用定義名稱DAY
4  {
5      sunday,      // 0
6      monday,      // 1
7      ...
8  };
9
10 weekday workday = monday;
11 printf("workday %d", workday);
12 // workday 1
```

union

union

- 與 struct 結構相似，用法也相同。
- union 內的屬性共用同一塊記憶體，同一時間內僅能用聯合內其中一種屬性。
- 主要用來表示同概念但不同資料類型的實體。
- 也可以使用 typedef，使用方式與 struct 相同。
- 記憶體空間爲宣告中最大的資料型態大小。

```
1  // 語法
2  union sample_t
3  {
4      float f;
5      int i;
6      char ch;
7  };
```

程式競賽


113年度
中正大學程式競賽
競賽日期：5/23 18:00-21:30
競賽地點：中正大學資訊處
報名時間：即日起至5/13
馬上掃碼報名！

報名網址

程式競賽

加分

- 金 +5
- 銀 +4
- 銅 +3
- 佳作 +2