C++ Primer Plus 6th Edition 笔记



# 4. 复合类型

## 1.数组

- 定义

  ```
  typeName arrayName[arraySize];
  ```

- 初始化

## 2. 结构

- 结构体
  - 定义

```
struct structName{
  type typeName1;
  type typeName2;
  …… ……
};

structName struct_name;
structName* struct_ptr_name;
```

- 共用体
  - 定义

```
union unionName{
  type typeName1;
  type typeName2;
  …… ……
};

unionName union_name;
unionName* union_ptr_name;
```

- 结构体和共用体的区别：

  结构体占用的字节数是每个成员占据字节数的总和，可以同时通过`struct_name.typeName`或`struct_ptr_name->typeName`存储访问；

  共用体占用的字节数是其中成员字节长度最大的，共用体每次只能存储一个值；

- 枚举
  - 定义

    `enum enumName{string1,string2,string3,…… ……};`
  - 默认情况下，将整数值赋给枚举量，从0开始依次分配；
    - `eg. enum bits{one=1, two, four=4, eight=8 };`第一个默认为0，后面未被初始化的，其值将比前面大1，上例中`two=2`；

## 3. 指针

- 例子

```
int * p1;
int* p2;
int *p3;
int*p4;
int* p5,p6;          //p5为int指针，p6为int型变量
```

- 注意
  - 指针必须指向一个确定的地址，否则可能会出现内存泄漏
    -
    ```
    int* ptr;
    *ptr = 321;          //错误，int型常量321会随机分配在内存中，可能会覆盖掉其他程序的重要数据
    ```
  - 指针初始化
    -
    ```
    int a = 321;
    int b =654;
    int* ptr1;
    int* ptr2;
    int* ptr3 = &b;
    ptr1 = &a;
    *ptr2 = b;       //错误，ptr2还没有指向确定的地址，不能进行赋值操作
    ```

```
int* ptr1;
int* ptr2 = new int;
ptr1 = new int;
*ptr1 = 321;
*ptr2 = 654;
delete ptr1;
delete ptr2;         //new与delete必须一起使用，否则可能会出现内存泄漏

//指针数组
int* p = new int[20];
…… ……
delete [] p;
```

## 4. 字符串

- char[]数组

```
char name1[] = {"Leonardo DiCaprio"};
char name2[] {"Matt Damon"};
char name3[20] = "Scarlett Johansson";
```

- string类

```
string name1 = {"Leonardo DiCaprio"};
string name2 {"Matt Damon"};
string name3 = "Scarlett Johansson";
```

- cin读取输入
  - 遇见空格该次读取结束，后续的字符进入输入队列，赋给下个cin读取

```
char name[20];
cin>>name;
char sex[10];
cin>>sex;
cout<<"name: "<<name<<endl;
cout<<"sex: "<<sex<<endl;

Result:
-> Scarlett Johansson
name: Scarlett
sex: Johansson
```

- get()和getline()读取一行字符串输入
  - getline()读取一行时将丢弃换行符
  - get()将换行符保留在输入队列中