第二章总结

我把第二章分为两个部分，第一个部分为C与C++的区别

第二个部分为C++的新增应用。

第一部分：

|  |  |  |
| --- | --- | --- |
|  | C | C++ |
| 输入 输出 | scanf() printf( ) | cin>> cout<< |
| 主要头文件 | #include<stdio.h> | #include<iostream>  using namespace std |
| 换行符 | printf(“/n”) | cout<<”/n” ;  cout<<endl; |
| 注释方法 | /\*........\*/ | /\*.........\*/  // |
| 定义常量 | #define | const  (比起define更安全) |
| 强制转换 | float y;  int x=(float)f; | 在C基础上，新加  x=int(f) |
| 形式参数 | 实参的个数必须和形参的个数形同 | 在定义函数的首部，允许为一个或多个形参指定数值，从右到左赋值 |
| 动态内存空间管理 | 在#include<stdlib.h>  使用malloc和calloc函数。释放则用free() | 新增关键字new，简单的申请动态空间。形式：new 数据类型名  或 new 数据类型名[整型常量]  int \*ptr;  ptr=new int;//申请一个int大小的内存空间  ptr=new int[10];//连续申请十个  注：ptr=new int(10)；实际上  ptr=new int;\*ptr=10;申请一个空间，赋值为10；  释放：delete 指针变量名（用于申请一个空间ptr=new int）  delete []指针变量名（用于申请一维数组的空间ptr=new int[10]） |

第二部分：

|  |  |
| --- | --- |
| 新增bool类型 | bool表达式的值为true时输出1，  为false时为0；也可用boolalpha控制输出true，false，noboolalpha控制输出1，0； |
| 名字空间（namespace） | namespace 名字空间名称{......;}   1. using namespace 名字空间名称 2. :: 域解析符或作用域预算符 3. using namespace 名字空间名称::局部内容名 |
| 域解析符:: | 扩大全局变量的可见范围  在有局部变量的时候，对隐藏的全局变量做访问 |
| 内联函数 | 代替宏定义，增强安全性，函数首部加inline |
| 函数重载 | 在形参的个数，类型，顺序方面有区别的不同函数以相同的函数名来命名，该同名函数成为重载。  判断函数重载是否正确，要保证任何一次调用都不会引起歧义。 |
| 新增引用的应用 | 数据类型 & 引用名=一个已定义的变量名；  （1）“&”不是取地址符，而是声明引用的一个标志。  （2）与引用参数对应的是实参只能是变量，不能是常量或者表达式。  （3）引用与指针的区别  int a=2; int a=2;  int \*p=&a; int &p=a;  a,p是两个变量 a,p是一个变量  指向两个地址 指向一个地址   1. 引用作为形参时，可修改对用实参的值。swap(&a,&b) 2. 引用作为返回值   int& fun(const int &x,int &y,int z)  x,z均不能作为返回值，x受保护，z为局部自动型变量。 |
| void类型的指针 | void类型指针为通用性指针，即任何类型的指针均可直接赋值给void类型的指针。  需要进行显示类型转换； |
| 异常处理 | 1. 检查异常（使用try语块）   try{  <进行异常检查的的语句>}   1. 抛出异常（使用throw语块） 2. throw <表达式>;或throw; 3. 捕捉异常（使用catch语块）   catch (<异常信息类型>或<变量>)  {  <异常处理语句>  } |