# 第三章 函数

3.1　引用传递参数

3.2　内联函数

3.3　带默认参数值的函数

3.4　函数重载

# 3.1引用传递参数

◆定义：引用(&)是标识符的别名,例如:

```
int i, j=10;
int &ri = i+j;
//建立一个int型的引用ri,并将其
//初始化为变量i的一个别名
ri = j;//相当于 i = j;
```

◆在用引用变量时需要注意什么呢？

# 引用变量注意3点：

◆声明一个引用时，必须同时对它进行初始化，使它指向一个已存在的对象。

◆一旦一个引用被初始化后，就不能改为指向其它对象。

◆引用可以作为形参

```
void swap(int &a, int &b) {…}
```

# 例 输入两个整数交换后输出

```
1.  #include<iostream>
2.  using namespace std;
3.  void swap(int a, int b) {
4.      int t = a;
5.      a = b;
6.      b = t;
7.  }
8.  int main() {
9.      int x = 5, y = 10;
10.     cout << "x = " << x << "  y = " << y << endl;
        swap(x, y);
11.     cout << "x = " << x << "  y = " << y << endl;
        return 0;
12. }
```

```
x = 5          y = 10
x = 5          y = 10
```

# 例 输入两个整数交换后输出

```
#include<iostream>
using namespace std;
void swap(int &a, int &b) {
        int t = a;
        a = b;
        b = t;
}
int main() {
        int x = 5, y = 10;
        cout << "x = " << x << "  y = " << y << endl;
        swap(x, y);
        cout << "x = " << x << "  y = " << y << endl;
        return 0;
}
```

```
x = 5        y = 10
x = 10       y = 5
```

# 3.2 内联函数

- 声明时使用关键字 inline。
- 编译时在调用处用函数体进行替换,节省了参数传递、控制转移等开销。

# 例3-14 内联函数应用举例

```
#include <iostream>
using namespace std;
const double PI = 3.1415926;
inline double calArea(double radius) {
  return PI * radius * radius;
}
int main() {
  double r = 3.0;
  double area  = calArea(r);
  cout << area << endl;
  return 0;
}
```

# 3.3 带默认参数值的函数

◆函数在定义形参时可预先给出默认值，调用时如给出实参，则采用实参值，否则采用预先给出的默认参数值。

```
int add(int x = 5,int y = 6) {
  return x + y;
}
int main() {
 int a,b;
 add(a,b);//10+20
 add(10);  //10+6
 add();  //5+6
```

# 默认参数值的说明次序

◆有默认参数的形参必须在形参列表的最后，也就是说默认参数值的右面不能有无默认值的参数。因为调用时实参与形参的结合是从左向右的顺序。

◆例：

```
int add(int x, int y = 5, int z = 6);//正确

int add(int x = 1, int y = 5, int z);//错误

int add(int x = 1, int y, int z = 6);//错误
```

# 默认参数值与函数的调用位置

◆如果一个函数有原型声明，且原型声明在定义之前，则默认参数值必须在函数原型声明中给出；

```
int add(int x = 5,int y = 6);
//原型声明在前
int main() {
  add();
}
int add(int x,int y) {
//此处不能再指定默认值
  return x + y;
}
```

# 默认参数值与函数的调用位置

◆而如果只有函数的定义，或函数定义在前，则默认参数值需在函数定义中给出。

```
int add(int x=5,int y=6){
//只有定义,没有原型声明
  return  x + y;
}
int main() {
  add();
}
```

# 例3-15计算长方体的体积

- 子函数getVolume是计算体积的函数，有三个形参：length（长）、width（宽）、height（高），其中width和height带有默认值。
- 主函数中以不同形式调用getVolume函数，分析程序的运行结果。

```cpp
//3_15.cpp
#include <iostream>
#include <iomanip>
using namespace std;

int getVolume(int length, int width = 2, int height = 3);

int main() {
  const int X = 10, Y = 12, Z = 15;
  cout << "Some box data is " ;
  cout << getVolume(X, Y, Z) << endl;
  cout << "Some box data is " ;
  cout << getVolume(X, Y) << endl;
  cout << "Some box data is " ;
  cout << getVolume(X) << endl;
  return 0;
}

int getVolume(int length, int width, int height){
  cout<<setw(5)<<length<<setw(5)<<width<<setw(5)<<height;
  return length * width * height;
}
```

# 3.4 函数重载

◆C++允许功能相近的函数在相同的作用域内以相同函数名声明，从而形成重载。方便使用，便于记忆。

```
int add(int x, int y);
float add(float x, float y);
```

形参类型不同

```
int add(int x, int y);
int add(int x, int y, int z);
```

形参个数不同

# 注意事项

- 重载函数的形参必须不同:个数不同或类型不同。
- 编译程序将根据实参和形参的类型及个数的最佳匹配来选择调用哪一个函数。

```
int add(int x,int y);
int add(int a,int b);
```

编译器不以形参名来区分

```
int add(int x,int y);
void add(int x,int y);
```

编译器不以返回值来区分

# 例3-16重载函数应用举例

- 编写两个名为sumOfSquare的重载函数，分别求两整数的平方和及两实数的平方和。

```cpp
#include <iostream>
using namespace std;

int sumOfSquare(int a, int b) {
    return a * a + b * b;
}
double sumOfSquare(double a, double b) {
    return a * a + b * b;
}
int main() {
    int m, n;
    cout << "Enter two integer: ";
    cin >> m >> n;
    cout << sumOfSquare(m, n) << endl;
    double x, y;
    cout << "Enter two real number: ";
    cin >> x >> y;
    cout << sumOfSquare(x, y) << endl;
    return 0;
}
```

# 小结

## ◆主要内容

- 函数的参数传递、内联函数、带默认参数值的函数、函数重载

## ◆达到的目标

- 学会将一段功能相对独立的程序写成一个函数，为下一章学习类和对象打好必要的基础。