引用的基本使用

2023年9月18日 16:30

作用： 给变量起别名

语法： `数据类型 &别名 = 原名`

示例：

```
int main() {

    int a = 10;
    int &b = a;

    cout << "a = " << a << endl;
    cout << "b = " << b << endl;

    b = 100;

    cout << "a = " << a << endl;
    cout << "b = " << b << endl;

    system("pause");

    return 0;
}
```

引用的注意事项

2023年9月18日 17:30

2.2 引用注意事项

- 引用必须初始化
- 引用在初始化后，不可以改变

示例：

```
int main() {

    int a = 10;
    int b = 20;
    //int &c; //错误，引用必须初始化
    int &c = a; //一旦初始化后，就不可以更改
    c = b; //这是赋值操作，不是更改引用

    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
    cout << "c = " << c << endl;

    system("pause");

    return 0;
}
```

不是说从变量c变成变量b，只是赋值

引用做函数参数

2023年9月18日 17:30

2.3 引用做函数参数

作用： 函数传参时，可以利用引用的技术让形参修饰实参

优点： 可以简化指针修改实参

```
//3. 引用传递
void mySwap03(int& a, int& b) {
    int temp = a;
    a = b;
    b = temp;
}

int main() {

    int a = 10;
    int b = 20;

    mySwap01(a, b);
    cout << "a:" << a << " b:" << b << endl;

    mySwap02(&a, &b);
    cout << "a:" << a << " b:" << b << endl;

    mySwap03(a, b);
    cout << "a:" << a << " b:" << b << endl;

    system("pause");

    return 0;
}
```

> 总结：通过引用参数产生的效果同按地址传递是一样的。引用的语法更清楚简单

引用做函数返回值

2023年9月18日 17:53

2.4 引用做函数返回值

作用：引用是可以作为函数的返回值存在的

注意：不要返回局部变量引用

用法：函数调用作为左值

```
//返回局部变量引用
int& test01() {
    int a = 10; //局部变量
    return a;
}

//返回静态变量引用
int& test02() {
    static int a = 20;
    return a;
}

int main() {

    //不能返回局部变量的引用
    int& ref = test01();
    cout << "ref = " << ref << endl;
    cout << "ref = " << ref << endl;

    //如果函数做左值，那么必须返回引用
    int& ref2 = test02();
    cout << "ref2 = " << ref2 << endl;
    cout << "ref2 = " << ref2 << endl;
```

```
22      cout << "ref2 = " << ref2 << endl;
23      cout << "ref2 = " << ref2 << endl;
24
```

```
test02() = 1000;

cout << "ref2 = " << ref2 << endl;
cout << "ref2 = " << ref2 << endl;

system("pause");

return 0;
```

引用的本质

2023年9月18日 17:30

2.5 引用的本质

本质：**引用的本质在c++内部实现是一个指针常量.**

讲解示例：

```
//发现是引用，转换为 int* const ref = &a;
void func(int& ref){
    ref = 100; // ref是引用，转换为*ref = 100
}
int main(){
    int a = 10;

    //自动转换为 int* const ref = &a; 指针常量是指针指向不可改，也说明为什么引用不可更改
    int& ref = a;
    ref = 20; //内部发现ref是引用，自动帮我们转换为: *ref = 20;

    cout << "a:" << a << endl;
    cout << "ref:" << ref << endl;

    func(a);
    return 0;
}
```

结论：C++推荐用引用技术，因为语法方便，引用本质是指针常量，但是所有的指针操作编译器都帮我们做了

常量引用

2023年9月18日 17:53

2.6 常量引用

作用： 常量引用主要用来修饰形参，防止误操作

在函数形参列表中，可以加const修饰形参，防止形参改变实参

```
//引用使用的场景，通常用来修饰形参
void showValue(const int& v) {
    //v += 10;
    cout << v << endl;
}

int main() {

    //int& ref = 10;  引用本身需要一个合法的内存空间，因此这行错误
    //加入const就可以了，编译器优化代码，int temp = 10; const int& ref = temp;
    const int& ref = 10;

    //ref = 100;  //加入const后不可以修改变量
    cout << ref << endl;

    //函数中利用常量引用防止误操作修改实参
    int a = 10;
    showValue(a);

    system("pause");

    return 0;
}
```