



“引用”的概念和应用

引用的概念 （教材第62页）

➤ 下面的写法定义了一个引用，并将其初始化为引用某个变量。

类型名 & 引用名 = 某变量名;

```
int n = 4;
```

```
int & r = n; // r引用了 n, r的类型是
```

引用的概念 （教材第62页）

➤ 下面的写法定义了一个引用，并将其初始化为引用某个变量。

类型名 & 引用名 = 某变量名;

```
int n = 4;
```

```
int & r = n; // r引用了 n, r的类型是 int &
```

引用的概念 （教材第62页）

➤ 下面的写法定义了一个引用，并将其初始化为引用某个变量。

类型名 & 引用名 = 某变量名；

```
int n = 4;
```

```
int & r = n; // r引用了 n, r的类型是 int &
```

➤ 某个变量的引用，等价于这个变量，相当于该变量的一个别名。

引用的概念

```
int n = 4;  
int & r = n;  
r = 4;  
cout << r; //输出 4  
cout << n;  
n = 5;  
cout << r;
```

引用的概念

```
int n = 4;  
int & r = n;  
r = 4;  
cout << r; //输出 4  
cout << n; //输出 4  
n = 5;  
cout << r;
```

引用的概念

```
int n = 4;
```

```
int & r = n;
```

```
r = 4;
```

```
cout << r; //输出 4
```

```
cout << n; //输出 4
```

```
n = 5;
```

```
cout << r; //输出 5
```

引用的概念

➤ 定义引用时一定要将其初始化成引用某个变量。

引用的概念

- 定义引用时一定要将其初始化成引用某个变量。
- 初始化后，它就一直引用该变量，不会再引用别的变量了。

引用的概念

- 定义引用时一定要将其初始化成引用某个变量。
- 初始化后，它就一直引用该变量，不会再引用别的变量了。
- 引用只能引用变量，不能引用常量和表达式。

引用的概念

```
double a = 4, b = 5;  
double & r1 = a;  
double & r2 = r1;  // r2也引用 a  
r2 = 10;  
cout << a << endl;  
r1 = b;  
cout << a << endl;
```

引用的概念

```
double a = 4, b = 5;  
double & r1 = a;  
double & r2 = r1; // r2也引用 a  
r2 = 10;  
cout << a << endl; // 输出 10  
r1 = b;  
cout << a << endl;
```

引用的概念

```
double a = 4, b = 5;  
double & r1 = a;  
double & r2 = r1;    // r2也引用 a  
r2 = 10;  
cout << a << endl;  // 输出 10  
r1 = b;              // r1并没有引用b  
cout << a << endl;
```

引用的概念

```
double a = 4, b = 5;  
double & r1 = a;  
double & r2 = r1;    // r2也引用 a  
r2 = 10;  
cout << a << endl;  // 输出 10  
r1 = b;              // r1并没有引用b  
cout << a << endl;  //输出 5
```

引用应用的简单示例

C语言中，如何编写交换两个整型变量值的函数？

引用应用的简单示例

C语言中，如何编写交换两个整型变量值的函数？

```
void swap( int * a, int * b)
```

```
{
```

```
    int tmp;
```

```
    tmp = * a; * a = * b; * b = tmp;
```

```
}
```

```
int n1, n2;
```

```
swap(& n1, & n2); // n1, n2的值被交换
```


引用应用的简单示例

➤有了C++的引用：

```
void swap( int & a, int & b)
```

```
{
```

```
    int tmp;
```

```
    tmp = a; a = b; b = tmp;
```

```
}
```

```
int n1, n2;
```

```
swap(n1,n2) ; // n1,n2的值被交换
```

引用作为函数的返回值(教材第63页)

```
int n = 4;  
int & SetValue() { return n; }  
int main()  
{  
    SetValue() = 40;  
    cout << n;  
    return 0;  
}
```

引用作为函数的返回值(教材第63页)

```
int n = 4;  
int & SetValue() { return n; }  
int main()  
{  
    SetValue() = 40;  
    cout << n;  
    return 0;  
} //输出: 40
```

常引用 (教材第65页)

定义引用时，前面加`const`关键字，即为“常引用”

```
int n;
```

```
const int & r = n;
```

r 的类型是

常引用

(教材第65页)

定义引用时，前面加const关键字，即为“常引用”

```
int n;
```

```
const int & r = n;
```

r 的类型是 `const int &`

常引用

不能通过常引用去修改其引用的内容:

```
int n = 100;
```

```
const int & r = n;
```

```
r = 200; //编译错
```

```
n = 300; //没问题
```

常引用和非常引用的转换

`const T &` 和 `T &` 是不同的类型!!!

`T &` 类型的引用或 `T` 类型的变量可以用来初始化

`const T &` 类型的引用。

`const T` 类型的常变量和 `const T &` 类型的引用则不能用来初始化 `T &` 类型的引用，除非进行强制类型转换。

QUIZ 1

下面程序片段哪个没错？

A) `int n = 4;`
 `int & r = n * 5;`

B) `int n = 6;`
 `const int & r = n;`
 `r = 7;`

C) `int n = 8;`
 `const int & r1 = n;`
 `int & r2 = r1;`

D) `int n = 8;`
 `int & r1 = n;`
 `const int r2 = r1;`

QUIZ 2

下面程序片段输出结果是什么？

```
int a = 1,b = 2;  
int & r = a;  
r = b;  
r = 7;  
cout << a << endl;
```

A) 1 B) 2 C) 7



下一小节：“const” 的用法