8 Constants

Hu Sikang

skhu@163.com

School of Computer

Beijing Institute of Technology

Contents

- Constant as variables, arguments and returning types
- Constant as objects, members in a calss

8.1 Value substitution

- The concept of ***constant*** (expressed by the const keyword) was created to allow the programmer to draw a line between what changes and what doesn't. This provides safety and control in a C++ programming project.
- It has since been put to use for pointers, function arguments, return types, class objects and member functions.

8.1 Value substitution

When defining a constant, we use macro definition: #define in C. But in C++ we use a new keyword: *const*. *const* is often used when the value cannot be changed.

#define PI = 3.14 → **const double PI = 3.14;**

So programmer knows which type the PI is. With macro definition, PI is only a symbol which means 3.14 but not a double.

8.1.1 Constant value

A constant must be initialized, and cannot be asssigned to.

Constant can be defined like these:

1. **const data_type variable_name = a constant expression**
   data_type const variable_name = a constant expression

const int x = 10; ⟷ **int const x = 10;**

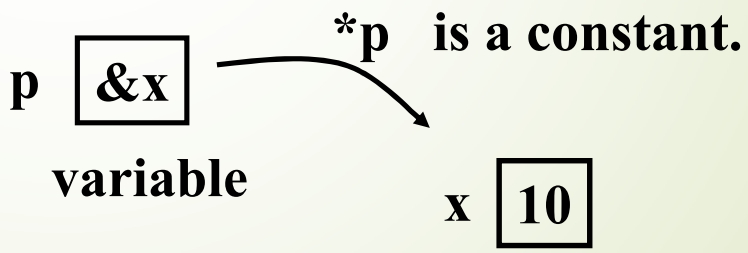
8.1.2 Pointer for constant value

2. pointer constant

   data_type const *variable_name

   In the definition, the value of variable_name is constant, but variable_name is variable.

```
int x = 10;
int const *p = &x;
```



8.1.2 Pointer for constant value

```
#include <iostream>
using namespace std;
void main()
{
        int x = 10, y = 100;
        int const *p = &x;
        *p = 50;     //error: *p is a constant.
        p = &y;     //ok.
}
```

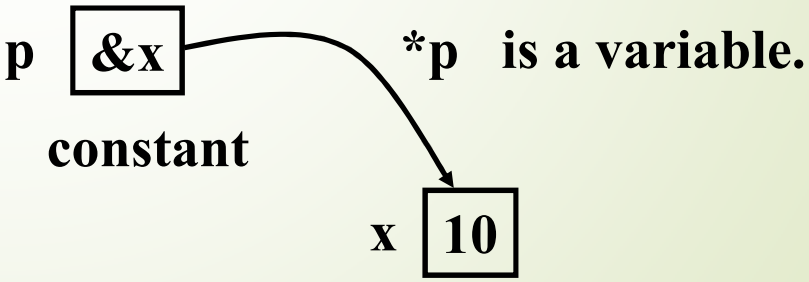
8.1.3 Constant pointer

3. pointer constant

data_type* const variable_name

In the definition, the value of *variable_name* is variable, but variable_name is constant.

```
int x = 10;
int * const p = &x;
```



***p is a variable.**

8.1.3 Constant pointer

```
#include <iostream>
using namespace std;
void main()
{
        int x = 10, y = 100;
        int * const p = &x;
        *p = 50;     //ok: *p is a variable.
        p = &y;     //error: p is a constant.
}
```

8.1.4 Constant array

4. array constant

data_type const array_name[constant expression]

const data_type array_name[constant expression]

In array definition, the every element of *array_name* is constant.

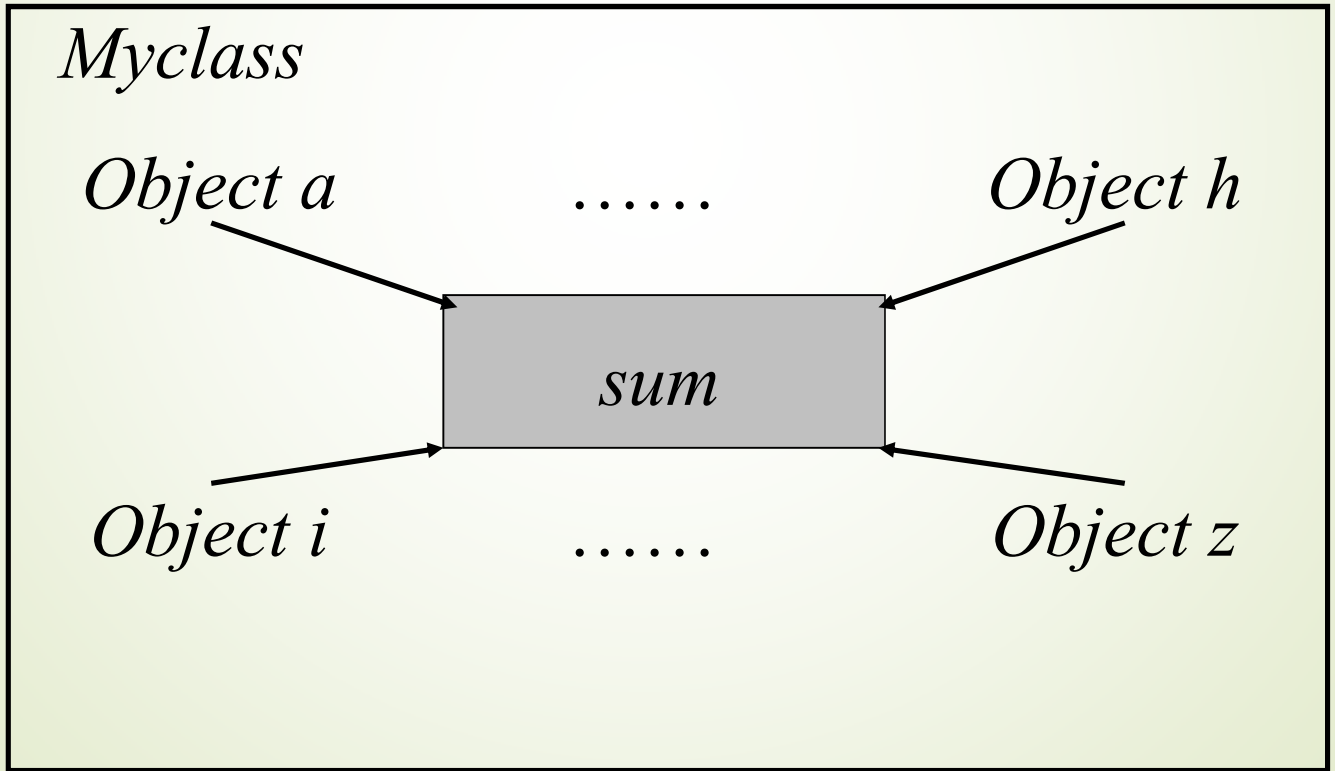
```
#include <iostream>
using namespace std;
void main()
{
        int const a[7] = {1, 2, 3, 4, 5, 6, 7};
        a[2] = 100;   //error: element is constant.
}
```

8.2 Classes

- Static Data Members
- Const Data Members
- Const Member Functions
- Const objects

8.2.1 static data members

private:
static int sum; ***//static data member***



8.2.1 Static data members

- *static* data members must be *initialized*.
  - *outside the class body*
  - *no static keyword .*
  - *qualified by the class name*

```
class Myclass
{
    static int sum;        //static data member
};

int Myclass::sum=8;  //initialization
```

8.2.1 *Static* data members

```
class Sample {
public:
    static void f(Sample& s)
    {
        x = 100;
        s.y = 100;
    }
    Obj.x = 10; //It's also OK!
private:
    int y;
};

int Sample::x = 0;  //initialization
```

```
void main()
{
    Sample obj, sub;
    obj.f(sub);          //ok
    Sample::f(sub);      //ok
    Sample::x = 10;      //ok
}
```

Only one copy exists to all objects of the class Sample.

8.2.2 Const Data Members

- The Const data members can not be modified.

- Const data members must be initialized in the ***member initializer list*** of constructor.

- Because ***consts and references*** must be initialized, a class containing *const or reference* members cannot be default-constructed.

8.2.2 Const Data Members

```
class A
{
public:
        A(int i);
        void Print();
private:
        const int a;            // const data member
        const int& r;           // const reference
};

A::A(int i):a(i),r(a)     // it must be initialized at list table
```

8.2.3 Const Member Functions

type function-name (arguments) const;

```
class Date
 {
  public:
        Date(int i,int j, int k){y=i; j=m; d=k;}
        int year()  const;
        int month() const     {  return m;  }
        int day()  { return d; }
 private:
        int y, m, d ;
  };
```

8.2.3 Const Member Functions

- The const member functions *do not modify the data members*.

```
int Date::year() const
{
        return ++y; // error: attempt to change member value
}
```

- When a *const* member function is defined outside its class, *the const suffix is required*.

```
int Date::year() const          // correct
{
    return y;
}
```

8.2.4 Const objects

- **A const object means that no data members of the object are changed during the object's lifetime.**

```
void main()
{
    Date A(2003,10,1);          // non-const object
    A.year();                   // ok
    const Date B(2000,9,8);     // const object
    B.year();                   // error
}
```

Summary

- The **const** keyword gives you the ability to define objects, function arguments, return values and member functions as constants.
- It can eliminate the preprocessor for value substitution without losing any preprocessor benefits.
- This provides a significant additional form of type checking and safety in your programming. The