C 和 C++有什么不同？

从机制上：c 是面向过程的（但 c 也可以编写面向对象的程序）；c++是面向对象的，提供了类。但是，c++编写面向对象的程序比 c 容易
从适用的方向：c 适合要求代码体积小的，效率高的场合，如嵌入式底层开发；c++适合更上层的，复杂的； linux 核心大部分是 c 写的，因为它是系统软件，效率要求极高。
从名称上也可以看出，c++比 c 多了+，说明 c++是 c 的超集；那为什么不叫 c+而叫 c++呢，是因为 c++比 c 来说扩充的东西太多了，所以就在 c 后面放上两个+；于是就成了 c++
C 语言是结构化编程语言，C++是面向对象编程语言。LUPA 开源社区 } n*r2C/M8f
C++侧重于对象而不是过程，侧重于类的设计而不是逻辑的设计。

c++概述

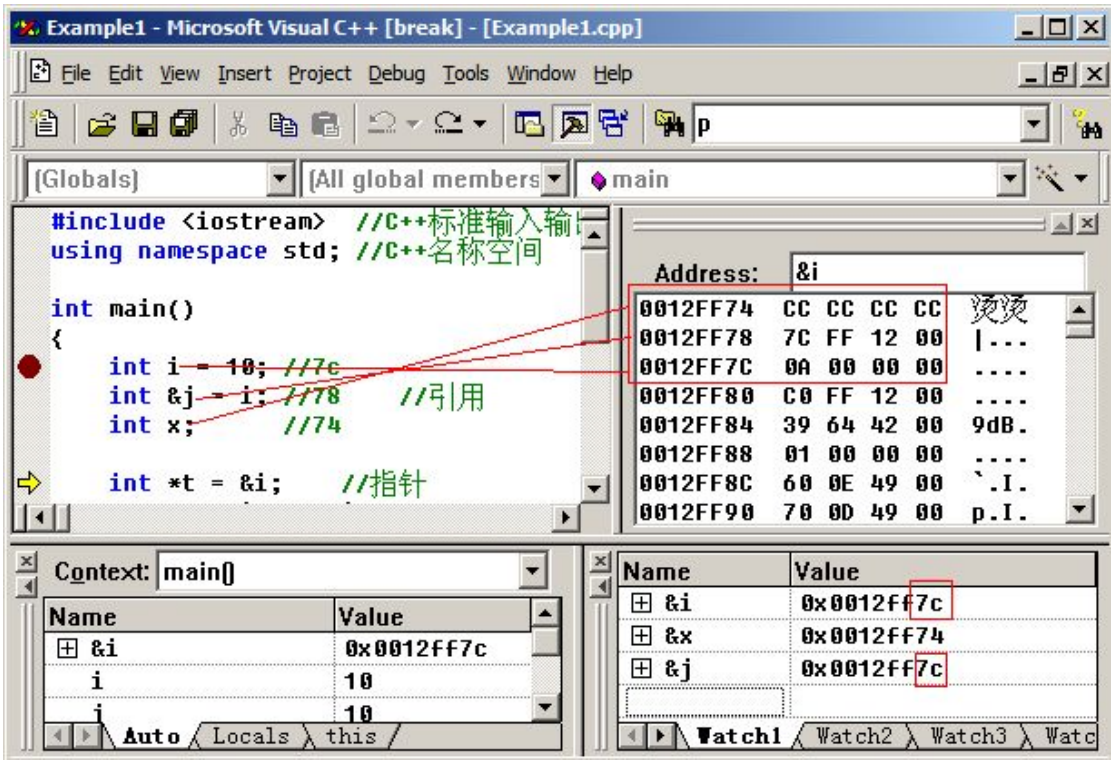
由 AT&T 贝尔实验室的 Bjarne Stroustrup 开发
从 C 语言派生的，与 C 语言是兼容的
新增保留字：class friend virtual inline private public protected const this string
新增运算符：new delete operator 作用域限定符：:

c++新增特性

- 引用
- const
- 函数的默认参数
- 内联函数
- 函数重载
- 输入/输出流

引用：

引用就是变量的别名，对引用的操作与对变量直接操作是完全一样的。就好比《射雕英雄传》里面，黄蓉叫郭靖为“靖哥哥”一样，靖哥哥和郭靖都是指同一个人，靖哥哥被梅超风的九阴白骨爪爪伤了，也就是郭靖被梅超风的九阴白骨爪抓伤了。
引用的声明： 数据类型 &引用名 = 初始值; （初始值是变量名）
引用在定义的时候必须要初始化。注意此处的&并不是取地址符，而是“引用声明符”。声明一个引用并不是定义了一个新的变量，只表示给变量取了一个别名，**不能再把该引用名作为其他变量名的别名。**编译器会给它分配内存空间，因此引用本身占据存储单元，但是引用表现出来给用户看到的，不是引用自身的地址，而是目标变量的地址。也就是说对引用取地址就是目标变量的内存地址。（参见代码 Example1）



（注意点：普通 int 变量 i 的地址是 0x0012ff7c，引用变量 j 本身的地址是 0x12ff78，可是当我们对 j 取地址的时候，不是 0x12ff78，而是 0x12ff7c，也许你会感觉很矛盾，但这是引用的特殊之处。也就是引用的特性和特点。）

c++的函数允许利用引用进行参数传递，具有高效性和安全性

引用作为函数参数的特点如下：

1. 在进行实参和形参的结合时，不会为形参分配内存空间，而是将形参作为实参的一个别名。使用引用传递函数的参数，在内存中并没有产生实参的副本，它是直接对实参操作；而使用一般变量传递函数的参数，当发生函数调用时，需要给形参分配存储单元，形参变量是实参变量的副本；因此，当参数传递的数据较大时，用引用比用一般变量传递参数的效率和所占空间都好。

2. 用引用，能达到同指针传递同样的效果，则函数内对形参的操作相当于直接对实参的操作，即形参的变化会影响实参。但是引用相对指针的优点如下：利用指针传递时，在被调函数中同样要给形参分配存储单元，且需要重复使用"*指针变量名"的形式进行运算，这很容易产生错误且程序的阅读性较差；另一方面，在主调函数的调用点处，必须用变量的地址作为实参。而引用更容易使用，更清晰。

```
#include <stdio.h>
void fun(int &m, int &n)
{
   int temp = m;
   m = n;
   n = temp;
}
int main()
{
   int m = 50;
```

操作的是传入的变量的地址，所以函数内改变了，传入的变量的值也就改变了

```
    int n = 100;
    printf("m = %d, n = %d\n", m, n);
    fun(m, n);
    printf("m = %d, n = %d\n", m, n);
    return 0;
}
```

返回引用:

```
int &fn(int &num)          返回一个引用变量
{
    return(num);
}
void main()
{
    int n1, n2;
    n1 = fn(n2);
}
```

const 常量定义：

c++中利用 const 来定义常量，与 C 语言中的#define 类似,用 const 修饰的不能修改,常用来定义一个符号常量。（参见代码 Example2）

函数的默认参数：

```
void func(int num1,int num2 = 3,char ch = '*'){}
void func(int num1=2,int num2,char ch='+');//错误
```

注意点：

1. 一旦给一个参数赋了默认值，则它后面的所有参数也都必须有默认值。
2. 默认值的类型必须正确。
3. 默认值可以在原型或者函数定义中给出，但不能在两个位置同时给出，建议在原型声明中指定默认值。

```
func(2,13,'+'); //都不采用默认值
func(1);        //第二个和第三个参数采用默认值
func(2,25);     //第三个参数采用默认值
func();         //所有这三个参数都采用默认值
func(2,,'+'); //错误！
```

优点:

如果要使用的参数在函数中几乎总是采用相同的值，则默认参数非常方便
通过添加参数来增加函数的功能时，默认参数也非常有用

(参见代码 Example3)

内联函数节省短函数的执行时间

```
inline float converter(float dollars);   //只需在函数的开头加上关键字 inline
```

注意点：

非常短的函数适合于内联
在链接时函数体会插入到发生函数调用的地方

预编译 → 编译 → 链接 → 运行

函数重载 overload:相同名称,不同参数(参数个数,类型,排列顺序)

```
void display();
void display(const char*);
void display(int one, int two);
void display(float number);
```

编译器通过调用时参数的个数和类型以及顺序来确定调用重载函数的哪个定义，不能仅仅通过函数的返回值类型的不同实现函数重载。

只有对不同的数据集完成基本相同任务的函数才应重载

就像人的信息是一个函数，但是具体是什么信息呢？有学生信息，有老师信息...

优点:

- 不必使用不同的函数名
- 有助于理解和调试代码
- 易于维护代码

（参见代码 Example4）

强制类型转换：

C++的强制类型转换相对 C 语言略有不同。

C 语言中的强制类型换成常常采用：int b; float a = (float)b;

C++中的强制类型转换常常采用：int b; float a = float(b); //这种方式类似于函数

string 类型

在 C 语言中定义字符串要用字符数组或者字符指针。如：

char szStr[10] = “hello”;　或　 char *szStr = “hello”;

但是在做字符串的相关复制和加法和比较等等运算时，要用到字符串相关的函数。

在 C++中用到 strcpy 等函数，包含头文件:

#include <string>　或　#include <string.h>

用到数学函数，如 sqrt()等时，包含头文件：

#include <cmath>　或　#include <math.h>

C++中对此进行了改进，引入了 string 类型。如：

string szStr1 = “hello”;　 string szStr2 = “world”;

string szStr = szStr1 + szStr2;　可以直接相加，而不用借助于 strcat 或 strcpy 等函数。

后面我们会详细讲解 string。并自己实现 string 类。

需要包含头文件：#include <string> using namespace std;

输入输出流:

流是字符集合或数据流的源或目的地

有两种流

输出流,在类库里称为 ostream

输入流,在类库里称为 istream

同时处理输入和输出,由 istream 和 ostream 派生双向流 iostream.

C++名字	对应设备	C 对应名字	默认设备
cin	标准输入流	stdin	键盘

cout	标准输出流	stdout	屏幕
cerr	标准错误流（非缓冲）	stderr	屏幕
clog	标准错误流（缓冲)	stdprn	打印机

1.输出流

cout 输出到屏幕 对应 ostream 类

```
char *str = "Hello";
int a = 100;
int *pa = &a;
cout << "a=" << a << endl;
cout << "*pa=" << *pa << endl;
cout << "The string is " << str << endl;
```

cout 提供: put 输出字符 write(char *buf,int n)输出字符串

```
cout << 'a' << ', ' << 'b' << '\n';
➔cout.put('a').put(', ').put('b').put('\n');
cout<<"\n\"Least said\n\t\tsoonest mended.\"\n\a"<<endl;
char pa[] = "hello world!";
cout << pa << endl;
➔cout.write(pa, sizeof(pa));   cout.put('\n');
```

cout 格式化控制输入输出

必须包含在头文件**#include <iomanip>**中

width()函数或 setw(int w)用于设置输出的字段宽度，默认是 0

fill()函数用于设置填充字符，默认是空格。如果指定的宽度大于实际的输出，C++用空格填充多余的位置

precision()函数或 setprecision(int d)将精度位数设置为 d，默认是 6

如：

```
cout << setw(6) << “cjy” << setw(10) << “123” <<”456”<< endl;
cout.width(10); cout.fill('*'); cout <<”cjy”<<endl;
```

对于 setw()和 width()都只能影响当前的一次输出。（参见代码 Example5）

控制浮点数显示: 浮点 定点 科学计数

```
#include <iostream>
#include <iomanip> //要用到格式控制符
using namespace std;

int main()
{
  double amount = 22.0/7;
  cout << amount << endl;
  cout << setprecision(0) << amount << endl       //精度为 0
       << setprecision(1) << amount << endl       //精度为 1
```

```
        << setprecision(2) << amount << endl      //精度为 2
        << setprecision(3) << amount << endl      //精度为 3
        << setprecision(4) << amount << endl;     //精度为 4
    cout << setiosflags(ios::fixed); //定点表示
    cout << setprecision(8) << amount << endl;      //精度为 8
    cout << setiosflags(ios::scientific) << amount << endl; //科学计数法表示
    cout << setprecision(6); //重新设置成原默认设置
    return 0;
}
```

使用头文件 iomanip.h 中的 setiosflags(ios::left)和(ios::right)控制输出对齐

输出 8 进制和 16 进制数　对应 C 语言中的 printf 的%x,%o,%d

```
int number = 1001;
十进制      cout << dec << number <<endl;
十六进制    cout << hex << number <<endl;
八进制      cout << oct << number <<endl;
```

2．输入流

int a,b; cin>>a>>b;空白符（空格、tab 键、换行符）只用于字符的分隔符，而本身不作为从输入流中提取的字符；

```
    char a[6];cin>>a;//容易内存越界（strcpy strcat）
```

使用 get()获取一个字符 char ch = cin.get(); putchar(ch); 或 char ch; cin.get(ch);

利用 cin.sync();清空缓冲区。类似于 C 语言的 fflush(stdin);

cin.get(char *buffer, int size,char delim='\n')读 size –1 个字符到 buffer，遇到'\n'在 buffer 最后加'\0'

```
    char a[6];cin.get(a,sizeof(a)); puts(a);
```

getline(char *buf,int Limit,Deline='\n');读入 1 行，不保存\n

```
    cin.getline(array, ARRAY_SIZE); //输入一行
    cin.getline(a,6);
```

使用 cin.read 作无格式读取一串字符 cin.read(buffer, 20);

```
   cout.write(buffer, cin.gcount());//表示 cin 读入的字符个数
```

（参见代码 Example6）

iostream.h 与 iostream 的区别

iostream.h 是为兼容旧有系统而保留的版本,是从 C 继承下来.而 iostream 是新的 C++标准支持的，最好是用 iostream 然后用 using namespace std;