优秀程序员之路——C++开发经验及技巧大汇总

1. 主函数是 “void main()” 还是 “int main()”？

通常 ANSI/ISO 的标准，正确的形式只有两种：

```
int main(){/*……*/}
int main(int argc, char **argv){/*……*/}
```

其他的形式在移植的时候可能出现错误。

2. 如何暂停一个控制台程序？

如果是在 windows 平台，可以用：

```
system(“PAUSE”);
```

其他平台，可以用：

```
cin.get();
```

3. 如何在 C++ 使用汇编？

```
int add(int a, int b)//
{
  return (a + b);
}
  int main()
{
  int nRet;
  _asm{
    push 1;
    push 2;
    call add;
    mov nRet, eax;
  }
  cout << “nRet:”<<nRet<<endl;
  return 0;
}
```

4. 如何读取一个文件?

```
#include <fstream.h>
int main() {
```

```
ifstream OpenFile("test.txt");
char ch;
 while(!OpenFile.eof()) {
OpenFile >> ch;
cout << ch;
}
return 0;
}
```

5. 如何保存一个文件?

```
#include <fstream.h>
int main() {
ofstream SaveFile("test.txt");
SaveFile << "Hello World!";
return 0;
}
```

6. 如何把一个整型变量转变为字符串?

```
#include <iostream>
#include <string>
#include <sstream>
using namespace std;
int main()
{
  stringstream str;
  int i = 10;
  str << i;
  cout << str.str()<<endl;
  system("PAUSE");
  return 0;
}
```

7. typedef 和预处理宏有何区别?

Typedef 和#define 都可以用来对已有的数据类型赋予一个新的别名。而问题的出现是当我们对指针这种类型的操作时。Typedefs 在这个时候能做出正确的处理。看下面的例子:

```
typedef char *string_t;
#define char *string_d
string_t s1,s2;
```

```
string_d s3,s4;
```

在上面的声明中，s1,s2 和 s3 都被申明为字符型指针，而 s4 却被申明为字符，这并不是预期所想得到的结果。那既然是这样，为什么还要使用预处理宏呢？因为预处理宏的好处是在使用诸如#ifdef 时体现出来的

8. 如何定义一个安全字符串?

```
char ch[4] = “test”;
```

这样虽然在 C 中式合法的，但在 C++中最好这样

```
char ch[5] = “test”;
```

或者

```
char ch[] = “test”;
```

或者

```
string ch = “test”;
```

9. 如何获取本程序的程序名？

```
#include <iostream>
using namespace std;
int main(int argc, char **argv)
{
……
cout <<”Myprogram name:”<< argv[0]<<endl;
……
}
```

10. 如何使用正确的文件路径?

```
#include <iostream>
#include <fstream>
int main()
{
ifstream fin(“e:\\test.txt”);//注意这里是 两个\ 如果一个就会出现错误
char ch;
fin >> ch;
return 0;
}
```

11. 如何让‘=’返回一个*this 类型的引用？

```
class CTest{
public:
```

```
CTest &operator=(const CTest &val)
{
return *this;
}
……
}
```

12. const 修饰符如何使用？

下面就针对指针说明一下 const 的使用方法：

```
char const *pChar;
const char *pChar;
```

这两种意思一样，区别在于程序员的习惯，有的人喜欢第一种，有的喜欢第二种，不过这两种在实际编程中都有出现。这个意思是指针所指的值的类型不可变，也就是说 pChar 指向一个字符，它不能只只其他字符，如：

```
int tmp = 20;
*pChar = tmp;
```

这就是错误的。

```
char ch = 'a';
char *const pChar = &ch;
```

这个意思是指针的值不能变，也就是这个指针变量所存的一个内存地址不能变,这个类型变量必须初始化。如：

```
int tmp = 20;
*pChar =tmp;
```

这是对的，但是下面是错的

```
pchar = &tmp;
char ch = 'a';
const char *const pChar = &ch;
```

这个意思是指针所指的变量的类型不能变，而且这个指针的值也不能变。

综上，const 总是修饰在它前面的变量或者修饰符，如果是变量类型也可以放在 const 后面。

13. C++的四种强制类型转换.

⑴ const_cast(expression)

用于强制消除对象的常量性。

```
const int nVal;
int nVal2 = const_cast(nVal);
```

⑵ dynamic_cast(expression)

用于安全的向下转型。

```
double dVal;
int nVal = dynamic_cast(dVal);
```

⑶ reinterpret_cast(expression)

用于底层的强制转型。例如讲一个指针变量转型为一个整型变量。

```
char *pChar = 'a';
int nPoint = reinterpret_cast(pChar);
```

⑷ static_cast(expression)

用于强制隐形转换。例如可以将整型转换为浮点型。

```
int nVal;
float fVal = static_cast(nVal);
```

14. 如何在 C++中声明函数为 inline?

隐式的声明方法是在一个类的定义的内部定义方法，如：

```
class CTest{
public:
int add(int a, int b){return (a + b);}
}
```

显示的声明方法是在类外部定义一个函数的时候，在函数名前面加 inline 关键字。例如：

```
template<typename T>
inline const T& std::max(const T& a, const T& b)
{ return a < b ? b : a; }
```

15. 如何使用 new 和 delete 动态分配和释放内存?

```
#include <iostream>
using namespace std;
int main()
{
int *pInt = new int[10];//使用 new 动态分配一个容量为 10  的 int 指针。
int count = 0;
for (;count <10; count++)
pInt[count] = count;
for (count = 0; count < 10; count++)
cout << pInt[count] <<endl;
delete []pInt;//动态是否内存，这里需要注意。。。
return 0;
```

```
}
```

16. 如何使用函数重载？

```
#include <iostream>
using namespace std;
class CTest{
public:
void Test();
void Test(int a);
void Test(int a, int b);
}
void CTest::Test(){cout<<”Empty!”<<endl;}//函数重载
void CTest::Test(int a){cout <<”One paramter:”<<a<<endl;}
void CTest::Test(int a, int b){cout <<”Two paramters:”<<a<<b<<endl;}
int main()
{
CTest tmp;
tmp.Test();
tmp.Test(1);
tmp.test(2, 3);
return 0;
}
```

17. 如何使用引用代替指针进行参数传递？

```
#include <iostream>
using namespace std;
void convert(int &a, int &b)//使用引用定义参数实现两个数的转换
{
int tmp = a;
a = b;
b = tmp;
}
int main()
{
int a, b;
a = 10;
b =20;
```

```
cout<<"Original!"<<endl;//输出转换前 a 和 b 的值
cout <<"a:"<<a<<endl<<"b:"<<b<<endl;
convert(a, b);//对 a 和 b 进行转换
cout<<"After convert!"<<endl;
cout <<"a:"<<a<<endl<<"b:"<<b<<endl;//输出转换后 a 和 b 的值
return 0;
}
```

18. 如何在函数中使用缺省参数?

```
#include <iostream>
#include <bitset>//包含 bitset 头
using namespace std;
void Test(int val, int num = 2)//函数定义，按 2 或者 10 进制输出一个整数
{
if(2 == num)//判定缺省值
{
bitset<32> bt(val);
cout << bt<<endl;
}
if( 10 == num)//使用输入值
cout<< val<<endl;
}
int main()
{
Test(128);//使用缺省参数
Test(128, 10);
return 0;
}
```

19. 如何使用 STL 中的 vector?

```
#include <iostream>
#include <vector>//使用 vector 头
using namespace std;
int main()
{
vector<int> vec;//定义一个 vector
for (int count = 0; count < 10; count++)//给 vector 赋值
```

```
vec.push_back(count);
for(int count = 0; count < vec.size(); count++)//输出 vector 中的值
cout<< vec[count]<<"\t";//类似数组的访问方式
cout <<endl;
return 0;
}
```

20. 如何定义一个类的继承类？

```
#include <iostream>
using namespace std;
class CBase{//基类
public:
void Test(){cout <<"Base class"<<endl;}
}
class CDerived: public Cbase{//从基类继承的子类
public:
void Test(){cout<<"Inherited class"<<endl;}//函数覆盖
}
int main()
{
  CDerived herit;
  herit.Test();//调用子类的 Test 函数
  ((CBase)herit).Test();//调用基类的 Test 函数
  return 0;
}
```

21. 如何使用一个联合体？

```
#include <iostream>
using namespace std;
int main()
{
  typedef union _TMP{
    int age;
    char sex;
} TMP;// 定义一个联合体，联合体内的变量是从同一个地址开始的。
TMP    uTmp;
uTmp.age = 23;//给联合体的 age 成员赋值
```

```
cout <<"Age:"<<uTmp.age<<endl;
uTmp.sex = 'm';//给联合体的 sex 成员赋值
cout <<"Sex:"<<uTmp.sex<<endl;
return 0;
}
```

22. 如何用指针操作数组？

```
#include <iostream>
using namespace std;
int main()
{
 char *pArr = new char[10];// 使用 new 操作符初始化一个 char 类型的指针
 for(int k = 0; k < 10; k++)
 pArr[k] = (65 + k);// 以数组的形式对其赋值
 for(int k = 0; k < 10; k++)
{
 cout << *pArr;
 pArr++;//通过指针的增加 来实现数据的访问
}
cout <<endl;
return 0;
}
```

23. 如何通过类保护数据？

```
#include <iostream>
#include <bitset>
using namespace std;
class Tmp{
private:
bitset<32> bt;// 通过把要数设置成 private 来保护。
public:
Tmp(){bt = 0;}
void SetBitset(int value){bt = value;}//通过此函数来设置被保护的数据 bt
bitset<32> &GetBitset(){return bt;}//通过此函数来获得被保护数据 bt 的一个引用
};
int main()
{
```

```
    Tmp tp;
    cout <<tp.GetBitset()<<endl;
    tp.SetBitset(32);
    cout<<tp.GetBitset()<<endl;
    return 0;
}
```

24. 如何在 main 函数执行之后再另外执行一个函数？

这个功能可以在 main 函数内部通过_onexit()函数注册一个函数，被注册的函数的返回值必须是 int 类型的，而且不能有参数。

```
#include <iostream>
using namespace std;
int func();//main 函数执行之后要执行的函数
int main()
{
  cout <<"This is on main"<<endl;
  _onexit(func);//注册 func 函数，以供在 main 执行之后执行
  system("PAUSE");
  return 0;
}
int func()
{
cout <<"This is after main"<<endl;
system("PAUSE");
return 0;
}
```

25. 如何定义一个函数模板？

有时候一些函数需要根据不同的变量类型来进行输出，这样函数模板就很有用，一下是如何定义一个简单的函数模板：

```
#include <iostream>
using namespace std;
template <class T>
T add(T a, T b)
{
return (a + b);
}
```

```
int main()
{
 cout<<"Add two int: 2 + 3 = "<<add(2, 3)<<endl;// 两个 int 型数据相加
 cout <<"Add two float: 1.1 + 2.2 = "<<add(1.1, 2.2)<<endl;//两个 float 型数据相加
 system("PAUSE");
 return 0;
}
```

26. 如何通过迭代器来实现对一个数据的访问？

```
#include <iostream>
#include <deque>
using namespace std;
int main()
{
 deque<int> que;// 定义一个整型的 deque 变量
 deque<int>::iterator que_i;// 定义一个 迭代器
 for(int k = 0; k < 10; k++)
 que.push_back(k + 65);// 通过一个循环向 deque 中写入十个字符
 que_i = que.begin();// begin 函数返回一个指向 deque 头部的迭代器
 for(;que_i != que.end(); que_i++)
 cout<<" "<<static_cast<char>(*que_i);//通过迭代器来访问 deque 的每一个成员
 cout <<endl;
 system("PAUSE");
 return 0;
}
```

27. 如何通过 map 实现 26 个字符的 ASCII 输出？

```
#include <map>
#include <iostream>
using namespace std;
int main()
{
 map<char, int> mp;//定义一个 map 键对应字符，值对应 ASCII
 map<char, int>::iterator mp_i;
 typedef pair<char, int> Tag_pair;
 for(int count = 0; count < 26; count++)
 mp.insert(Tag_pair(count + 65, count + 65));//向 map 中写入 ASCII 和 26 个字符
```

```
 for(mp_i = mp.begin(); mp_i != mp.end(); mp_i++)
 cout<<mp_i->first<<"\t"<<mp_i->second<<endl;//通过一个循环实现输出
 return 0;
}
```

28. 如何通过 bitset 来实现大小写字母的转换？

```
#include <bitset>
#include <string>
#include <iostream>
using namespace std;
void convert(string &str);
int main()
{

string test("hello world");
convert(test);
cout <<test<<endl;
system("PAUSE");
return 0;
}
void convert(string &str)
{
 bitset<8> bt;
 for(int count = 0; count < str.length(); count++)
 {
  if(('A' <= str[count] <= 'Z')||('a' <= str[count] <= 'z'))
  //测试是不是字母
  {
   bt= static_cast<int>(str[count]);
   //将字符转换为正说
   if(bt.test(5) == 1)
   //如是是小写就转换
   {
    bt.set(5, 0);//设置第五位为 1，使其成为大写
    str[count] = bt.to_ulong();//把修改后的字符赋值给原字符
   }
  }
```

```
 }
}
```

29. 如何通过 set 来实现正序和倒序的输出？

```
#include <set>
#include <iostream>
using namespace std;
int main()
{
  set<int> st;
  set<int>::iterator i;
  set<int>::reverse_iterator j;
  st.insert(10);
  st.insert(200);
  st.insert(2);
  st.insert(1);
  for(i = st.begin(); i != st.end(); i++)//正序  输出
  cout << *i<< " ";
  cout <<endl;
    for(j = st.rbegin(); j != st.rend(); j++)//反序  输出
  cout << *j<< " ";
  cout <<endl;
    system("PAUSE");
  return 0;
  }
```

30. 如何定义一个类模板？

```
#include <iostream>
using namespace std;
template <typename T> class Test{
private:
        T val1;
        T val2;
public:
        ~Test();
        Test(T a, T b);
        T Max();
        T Min();
```

```
};
template <typename T> Test<T>::~Test(){}//定义析构函数
template <typename T> Test<T>::Test(T a, T b)//定义构造函数
{
val1 = a;
val2 = b;
}
template <typename T> T Test<T>::Max()//返回一个最大值
{
 return (val1 < val2) ? val2 : val1;
}
template <typename T> T Test<T>::Min()返回一个最小值
{
 return (val1 < val2) ? val1 : val2;
}
int main()
{
      Test<int> t(2, 4);
      cout <<"Max:"<<t.Max()<<endl;
      cout <<"Min:"<<t.Min()<<endl;

      system("PAUSE");
      return 0;
}
```

31. 如何定义一个整数使其只能取固定的几个值？

```
#include <iostream>
using namespace std;
int main()
{
 enum week{Mon = 1, Tue = 2, Wed = 3, Thu = 4, Fri = 5, Sat = 6, Sun = 7};
//week 定义的变量只能取 里面固定的几个值
week test;
test = Mon;
cout << test<< endl;
return 0;
}
```

32. 如何定义和使用宏？

```
#include <iostream>
using namespace std;
#define DISPLAY(paramter)    cout<< paramter<<endl
#define MIN(a, b) ((a < b) ? a : b)
#define RET return 0
int main()
{
  int val = MIN(4, 10);
  DISPLAY(val);
  RET;
}
```

33. 如何在一个字符串中删除一个指定的字符？

```
#include <iostream>
#include <string>
using namespace std;
void del(string &str, char ch);//定义删除指定字符的函数
int main()
{
  string test("abcdefghijk");
  del(test, 'f');//删除字符'f'
  cout <<test<<endl;
  cin.get();
  return 0;
}
void del (string &str, char ch)
{
  int a = static_cast<int>(str.find(ch, 0));
  basic_string<char>::iterator itr = str.begin() + a;
  str.erase(itr);
}
```

34. 如何通过 cout 输出 16 和 8 进制数？

```
#include <iostream>
using namespace std;
int main()
```

```
{
  int test = 64;
  cout <<"DEC:"<<test<<endl;
  cout <<"HEX:"<<hex<<test<<endl;
  cout <<"OCT:"<<oct<<test<<endl;
  return 0;
}
```

35. 如何在程序执行的时候响铃？

```
#include <iostream>
using namespace std;
int main()
{
  cout<<"hello"<<"\a"<<"world"<<"\a"<<endl;
  return 0;
}
```

36. 如何用 rand 生成随机数？

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
  srand(static_cast<int>(time(NULL)));// 以时间为种子
  int num = rand();//生成随机数
  cout <<num<<endl;
system("pause");
return 0;
}
```

37. 如何使用函数指针？

函数指针定义的时候要根据函数原型来定义，例如函数原型：int add(int a, int b)，这个函数的函数指针这样定义：int (*pFunc)(int a, int b);下面是示例：

```
#include <iostream>
using namespace std;
typedef   void (*pFunc)();// 定义一个函数指针 pFunc，
void Func()
{
```

```
  cout <<"This is a test for func point"<<endl;
}
int main()
{
  pFunc test = Func;
  test();
  return 0;
}
```

38. 如何判断有没有键按下？

通过使用 conio 库的一个函数来判断，这个函数是：int kbhit(void);

```
#include <conio.h>
#include <iostream>
using namespace std;
int main()
{
while(!kbhit()) //当没有键按下
{
cout<<"No key pressed"<<endl;
}
   cout<<"A key pressed"<<endl;
   system("pause");
   return 0;
}
```

39. 如何从一个字符串中提取整数？

```
#include <iostream>
#include <string>
#include <sstream>
using namespace std;
int main()
{
string str = "2 3 bigd 100 +";
istringstream is(str);
int i;
char ch;
cout <<"输出字符串中的数字："<<endl;
while(is >> ch)
```

```
    {
        if(ch >= '0' && ch <= '9')
        {
          is.putback(ch);
          is >> i;
          cout << i << endl;
        }
    }
    system("pause");
    return 0;
    }
```

40. 如何倒置一个字符串？

这需要用到字符串的逆向函数。示例：

```
    #include <iostream>
    #include <string>
    using namespace std;
    int main()
    {
    string testStr("hello, world");
    cout << testStr<<endl;
    testStr.assign(testStr.rbegin(), testStr.rend());
    cout <<testStr<<endl;
    system("pause");
    return 0;
    }
```

41. 如何实现直接插入排序？

```
    #include <iostream>
    #include <cmath>
    #include <ctime>
    using namespace std;
    class Sort
    {
    private:
    int num[10];
    int key;
    public:
```

```
Sort()//初始化测试数据
{
    srand((unsigned)time(NULL));
    for(int i = 0;i < 10;i++)
      num[i] = static_cast<int>((((double)rand() / (double)RAND_MAX) * 100.0 ) +
0.0);

}
void SortNum()//排序的具体实现
{
    int j;
    for(int i = 1;i < 10;i++)
    {
      key = num[i];
      j = i - 1;
      while(j >= 0 && key < num[j])
      {
        num[j + 1] = num[j];
        j--;
      }
      num[j + 1] = key;
    }
}
void Display()//显示数据
{
    for(int i = 0;i < 10;i++)
      cout << num[i] << endl;
}
};
int main()
{
Sort tmp;
tmp.SortNum();
tmp.Display();
system("pause");
return 0;
}
```

42. 如何理解指针也是一种迭代器？

在 C++里面，指针也是一种迭代器，可以用指针操作迭代器。

```
#include <iostream>
#include <algorithm>
#define SIZE 20
using namespace std;
int main()
{
  int arr[SIZE], *p;
  arr[10] = 20;
  p = find(arr, arr + SIZE, 20);
  if( p == (arr+ SIZE))
  cout <<"Not find '20'<<endl;
  else
  cout <<"already find 20<<endl;
  return 0;
}
```

43. 如何使用 STL 中的绑定器函数对象？

一个绑定器使用另一个函数对象 f()和参数值 V 创建一个函数对象。被绑定函数对象必须为双目函数，也就是说有两个参数,A 和 B。STL 中的帮定器有：
bind1st() 创建一个函数对象，该函数对象将值 V 作为第一个参数 A。
bind2nd()创建一个函数对象，该函数对象将值 V 作为第二个参数 B
示例如下：

```
#include <iostream>
#include <list>
#include <algorithm>
#include <functional>
using namespace std;
int arr[10] = {1,2,3,4,5,6,7,8,10};
list<int> lst(arr, arr + 10);
int main()
{
  int num = 0;
  count_if(lst.begin(), lst.end(), bind1st(greater<int>(), 8), num);
cout<< " Number less than 8:"<< num <<endl;
return 0;
```

```
}
```

44. 如何删除字符串中的所有空格，或者其他字符？

string 没提供这样的函数，下面试是其实现方法：

```
#include <iostream>
#include <string>
#include <functional>
using namespace std;
int main()
{
  string str(“ this    is   a    t e s t   ! B    y e”);
  str.erase(remove_if(str.begin(),   str.end(),   bind2nd(equal_to<char>(),   ‘   ‘)),
str.end());
  // 这里就完成了对所有空格的删除操作。字符可以随便指定。
  cout << str<<endl;
  return 0;
}
```

45. 如何重定向输出流？

就是把应该输出到屏幕的东西输出到文件或者其他流对象。只要改变 ostream 的 rdbuf，就可以实现重定向，下面试例子：

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
      ofstream outf("e:\\out.txt");
      streambuf *strm_buf=cout.rdbuf();
      cout.rdbuf(outf.rdbuf());
      cout<<"write something to file"<<endl;
      cout.rdbuf(strm_buf);    //recover
      cout<<"display something on screen"<<endl;
      system("PAUSE");
      return 0;
}
```

46. 如何使用智能指针？

智能指针 auto_ptr 也是一个对象，不过用它操作指针更安全，不用考虑内存泄露问题。下面是一个实例：

```
#include <iostream>
using namespace std;
class A{
private:
        char ch;
public:
        A(char c){ch = c;}
        void Func(){cout <<"Test from "<<ch<<endl;}
};
int main()
{
 A *a = new A('a');// 构造一个对象 a
 auto_ptr<A> b (a);//通过智能指针创建一个指向 a 的指针
 b->Func();      //通过指针操作 a 对象。
 return 0;
}
```

47. 如何通过 new 申请一个三维数组？

```
#include <iostream>
using namespace std;
int main()
{
 int ***arr;
 int i, j, side, height, width;
 side = 5;
 height = 6;
 width = 7;
 arr = new int **[side];
 for (i = 0; i < side; i++)
 {
  arr[i] = new int *[height];
  for (j = 0; j < height; j++)
  arr[i][j] = new int[width];
 }
 arr[1][2][3] = 123;// 测试一下啊三维数组
```

```
  cout << arr[1][2][3]<<endl;
  return 0;
}
```

48. 如何使用 istream_iterator 和 ostream_iterator?

下面的例子展示了使用方法：

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <algorithm>
#include <vector>
#include <iterator>
using namespace std;
int main()
{
  vector<int> vec;
  for (int i = 0; i < 10; i++)
  vec.push_back(i);
  copy(vec.begin(), vec.end(), ostream_iterator<int>(cout, " "));
  cout <<endl;
  ostream_iterator<double> os_iter(cout, "~");
  *os_iter = 1.0;
  os_iter++;
  *os_iter = 2.0;
  *os_iter = 3.0;
   return 0;
  }
```

49. 如何计算一个文件的行数?

计算一个文件的行数，也就是计算一个文件有多少个’\n’字符，通过下面的列子详细说明：

```
#include <iostream>
#include <fstream>
#include <sstream>
using namespace std;
int main()
{
  ifstream inFile("e:\\out.txt");
```

```
  int line_count = count(istreambuf_iterator<char>(inFile.rdbuf()),
istreambuf_iterator<char>(), '\n');// 关键地方，通过计算'\n'计算文件的行数
  cout <<"line:"<<line_count<<endl;
  cin.get();
  return 0;
  }
```

50. 如何判断一个文件是不是 PE 文件？

主要是通过 PE 头的相关信息来确定，用到的数据结构有 IMAGE_DOS_HEADER 何 IMAGE_NT_HEADERS，示例程序如下：

```
#include <windows.h>
#include <iostream>
using namespace std;
int main()
{
  HMODULE imageBase = GetModuleHandle(NULL);
  PIMAGE_DOS_HEADER pDosHeader =
(PIMAGE_DOS_HEADER)imageBase;
  if(pDosHeader->e_magic != IMAGE_DOS_SIGNATURE)
  {
    cout<<"Unknown file type!"<<endl;
    return -1;
  }
  PIMAGE_NT_HEADERS pNtHeader =
(PIMAGE_NT_HEADERS)((DWORD)pDosHeader + pDosHeader->e_lfanew);
  if(pNtHeader->Signature != IMAGE_NT_SIGNATURE)
  {
     cout<<"Unknown file type!"<<endl;
     return -2;
  }
  cout <<"This is a PE file"<<endl;
  return 0;
}
```

51. 如何显示当前的时间？

```
#include <iostream>
#include <time.h>
using namespace std;
```

```
int main()
{
time_t t = time(NULL);
struct tm    *local = localtime(&t);
int year = local -> tm_year + 1900;
int month = local -> tm_mon + 1;
int day = local -> tm_mday;/
int hour = local -> tm_hour;
int minute = local -> tm_min;
int second = local -> tm_sec;
cout <<"当前时间:    "<<year<<"-"<<month<<"-"<<day;
cout <<" "<<hour<<":"<<minute<<":"<<second<<endl;
system("pause");
return 0;
}
```