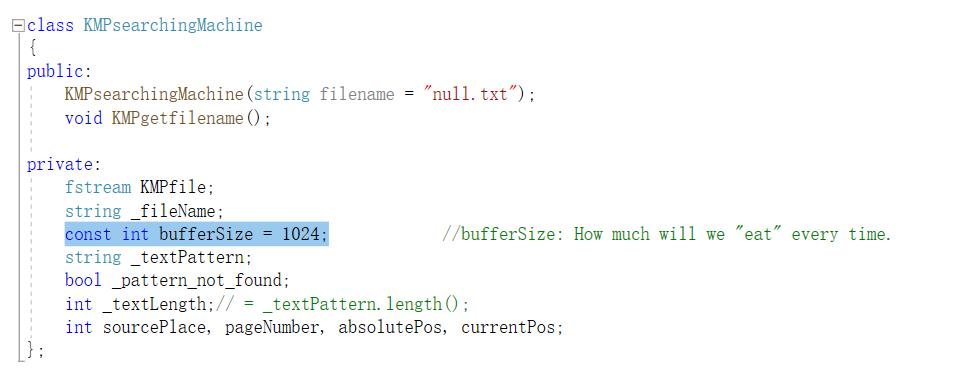
**Cin>> file>> always fails to record everything when you type in a SPACE.**

**Use getline(cin, bufferString);’**

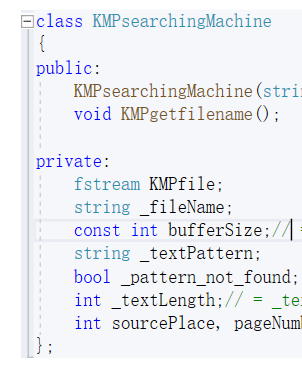
**Or getline(file, bufferString);**

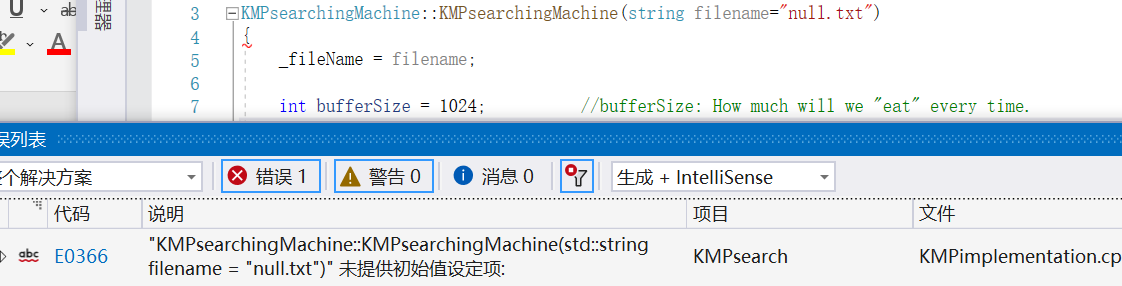
**有常对象时，不能在构造函数中赋值，必须在类体声明时直接赋值。**

**正确：**

****

**错误：**

****

****

**类模板的声明和定义必须写在同一个文件中。因为不同的文件时独立编译的，而且类模板/函数模板的代码的实现文件.cpp在编译时不会直接真正生成二进制文件，而是在编译到某个模板实例化语句时再编译它。**

**所以如果找到函数模板实例化语句时，会编译.h为二进制，然后找.cpp但是此时.cpp尚未被编译为二进制，机器不认识，就会报告链接错误LNK2019无法解析的外部命令。**

**而如果都写在.h中就能一口气都编译完，然后机器就能理解了。**

**类的成员函数不能将非静态的成员变量作为默认参数。**

**因为如果一个类的部分数据成员没有实例化时就调用这种成员函数，就必然会出错，所以编译器会禁止这种写法。**

**类的成员函数的默认参数需要用静态数据成员来写。**

**C++read函数不是很好，尤其是无法检测真正读入了多少数据，因此可能不能知道到底有多长。可以自己实现为:**

|  |
| --- |
| void KMPsearching\_engine::read\_safe(char\* \_text\_origin,unsigned long \_text\_origin\_length)  {  {  int i = 0;  for (; i < \_text\_origin\_length && KMPfile.peek() != EOF; i++)  {  KMPfile.get(\_text\_origin[i]);  }  \_text\_origin[i] = '\0';  }  \_text\_origin[\_text\_origin\_length] = '\0';  } |

//if you want a safe input, use std::string. If you want to input an integer, use atoi

//atoi(): convert a string"-1919810"~"1919810" to integer(with signum)

//If parameter doesn't represent any int, return 0.

//If parameter represent a float(double), return floor(para).

//e.g. atoi("#$%^&\*")==0;

//e.g. atoi("3.6")==3;

|  |
| --- |
| string \_buff = "";  std::cin >> \_buff;  directive = atoi(\_buff.c\_str());  while (directive > 5 || directive < 1)  {  std::cout << "\nInvalid input, please type in again!\n";  std::cin >> \_buff;  directive = atoi(\_buff.c\_str());  } |

**How to initialize an Boolean array: (e.g. length==3)**

myBooleanArray[3] = {0};

**\*don’t use {false}; must use {0};**

**How to assign a file pointer .**

|  |
| --- |
| **fstream filesource;’**  **void method(fstream& fileref)**  **{**  **fstream\* fp= & fileref;**  **fp->write(…);**  **}** |

**How to write a std::string to file.**

|  |
| --- |
| ptr->write((char\*)this->dataSonSTR.c\_str(), sizeof(this->dataSonSTR.c\_str())); |

**If the compiler tells you “友元声明无效”**

**很有可能是没有include 友元类头文件。**

**“不允许使用不完整的类” ：仅仅生命但没有链接头文件。**

**凡是使用string 和 file 都必须要用namespace std;**

#include <fstream>

using namespace std;

**How to open files elegantly**

|  |
| --- |
| void FileManager::openFile()  {  \_file.close();  if (\_filename.length() == 0) //If haven't created any file...  { //Create and open.  getFilename();  \_file.open(\_filename.c\_str(), ios::out | ios::app | ios::binary );  }  else  \_file.open(\_filename.c\_str(), ios::out | ios::app | ios::binary|ios::\_Nocreate);  //If a file has been opened...  if (\_access\_s(\_filename.c\_str(), 06)) //Don't create, just open.  {  cout << "file open error!" << endl;  throw 1; //1==fileOpenErr;  }  }  //\*\*When you open a file more than once, sometimes error can happen if you didn’t use “ios::Nocreate” |

/\*ios::cur in a constant, it's a mode number and it doesn't tell you where the pointer is pointing to.

use (int)(tellg()) to get where the pointer is.

to move your file pointer forward, use " myfile.seekg((int)(myfile.tellg()) + 1) "

move back is myfile.seekg((int)(myfile.tellg()) -1 )

move back and check what's on the road : myfile.seekg((int)(myfile.tellg()) -2) + myfile.get(\_singleChBuffer); and if you find one,

move back one char--myfile.seekg((int)(myfile.tellg()) -1 )\*/

/\*find the nearrest '\n' upstream and place the pointer to it.\*/

int starPos = 21;

pKMPfile->seekg(starPos);

char \_bufferSingleCh='-';

pKMPfile->get(\_bufferSingleCh);

cout << \_bufferSingleCh;

while (\_bufferSingleCh!='\n')

{

pKMPfile->seekg((int)(pKMPfile->tellg())-2);

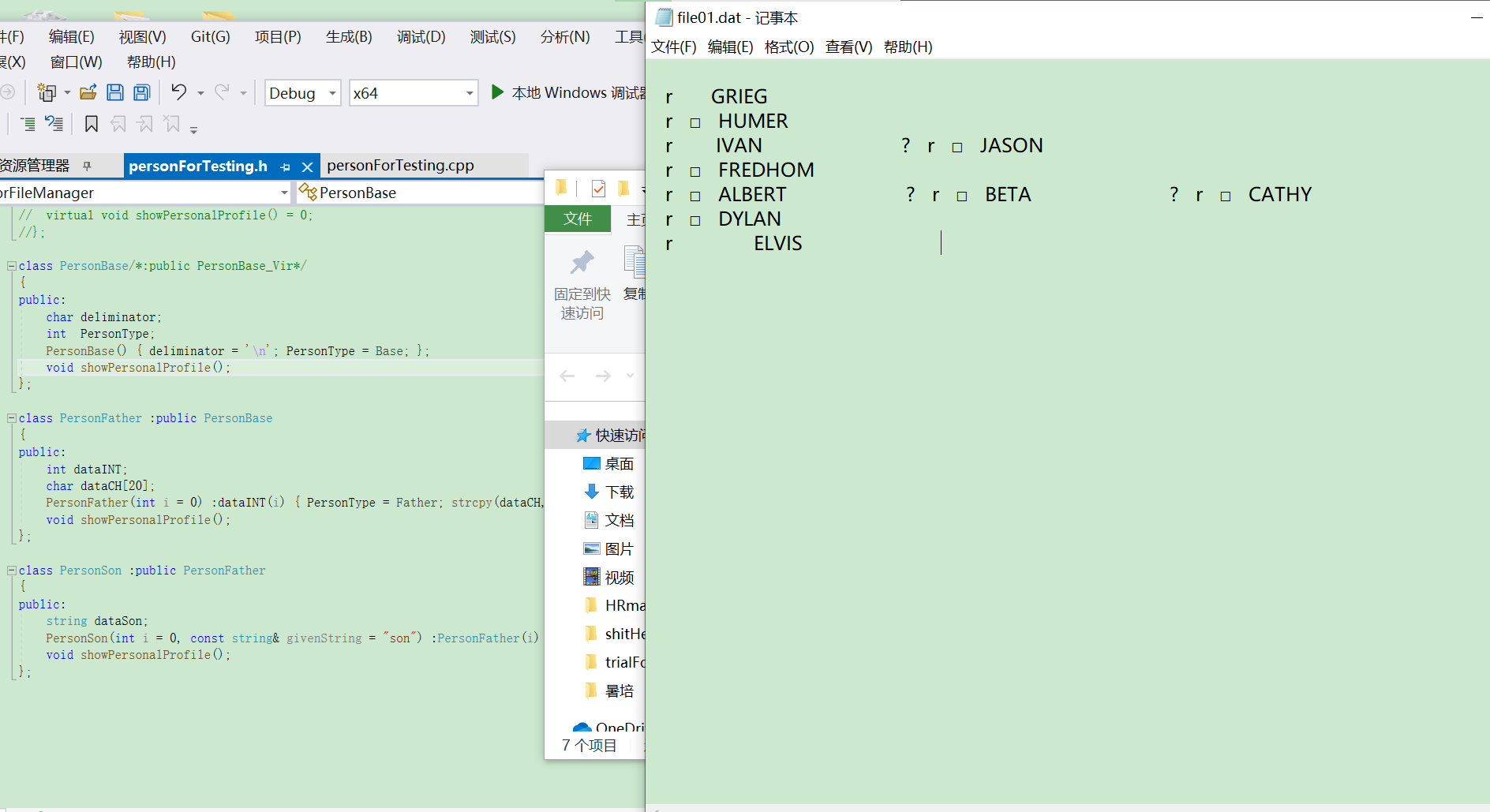
pKMPfile->get(\_bufferSingleCh);

}

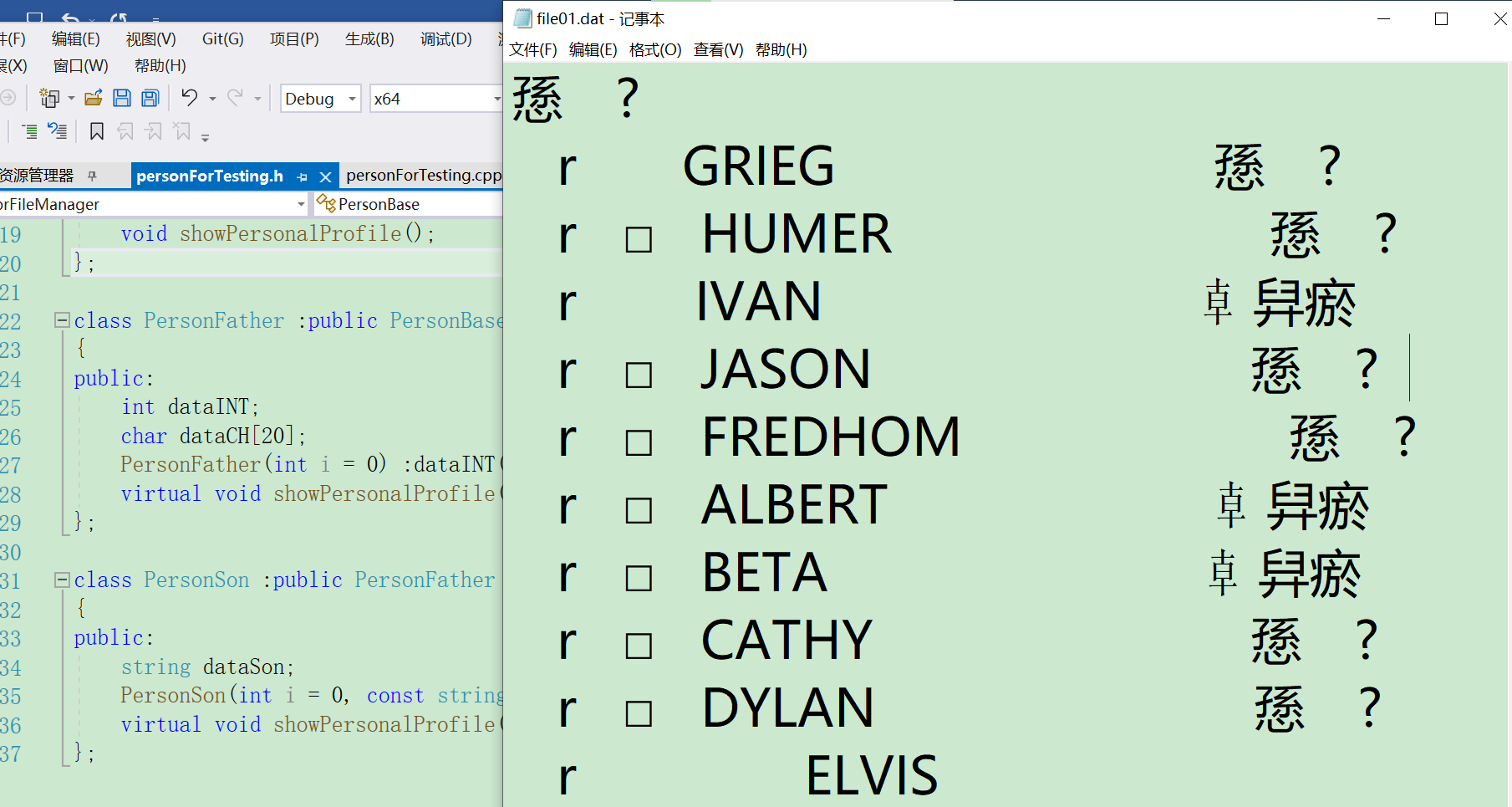
pKMPfile->seekg((int)(pKMPfile->tellg()) - 1);

**Using virtual function table will change the structure of a class obj.**

**Before(not using virtual functions:\_**

****

**After☹using virtual functions)**

****

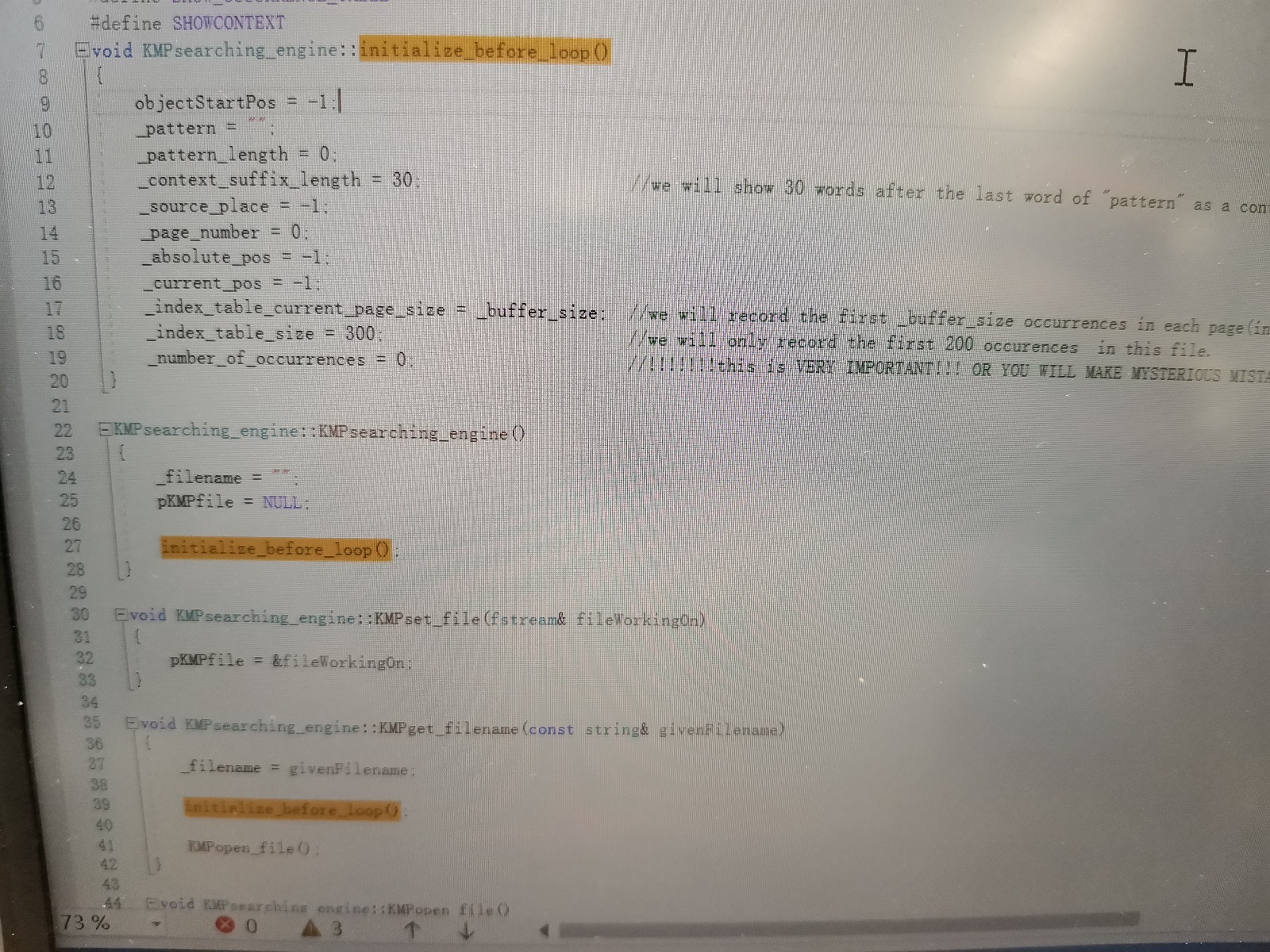
**When overriding a virtual function in a family, functions of derived classes should have the SAME RETURN VALUE TYPE as the base class. When RETURN VALUES are the sub-objects ’s pointer/reference of the return Val of base class virtual function, such kind of override is also acceptable(In this case, return Val of derived classes are “covariant” of the base class return values).**

**IF YOU MEET A PROBLEM/BUG THAT SHOWS A DISTINCE FEATURE OF “HISTORY RELIANT”----THAT IS, RESULTS OF LATTER CALLING OF A FUNCION CARRIES THE ‘MEMORY’ OF FORMER TIMES OF CALLING---**

**THE FIRST THING YOU SHOULD DO IS NOT DIVING INTO THE SEA OF CODES TO DEBUG.**

**THE FIRST THING U SHOULD DO IS TO GET BACK TO THE START OF EACH LOOP(ROUND OF CALL)**

**AND CHECK WHETHER IMORTANT DATA HAS BEEN RESET OR NOT!!!!!!!!!!!!!**

****

**If you find that you can’t read anything back from a file,**

1. **(Highly possible) your file pointer is not adjusted to a proper place(e.g. it’s at end of file)**
2. **(Highly possible) your file doesn’t have read permission. OR(more commonly) you used ios::app and this mode will glue your file pointer to end of file and you will never read anything.**

**If your compiler tells you that your abstract doesn’t exist or, more frequently if the compiler declares your concrete class as an abstract class and thus is can’t instantiate, DECLARE ABSTRACT CLASS IN A SINGLE .h HEADER FILE!!!!!!!!!!!!!!!!!!!!!!!!!!!!**

**If a class needs to define data member or member functions related to a struct , place structure definitions in the SAME FILE of these classes. Or the compiler will tell you “ Syntax error（语法错误：标识符blablabla）”**

**NEVER MAKE THE PARAMETERS LOOK EXACTLY THE SAME AS YOUR DATA MEMBER.**

**YOU WILL DEFINITELY FOOL YOUR SELF INTO BELIEVING THAT YOU’VE ALREADY UPDATED THIS MEMBER’S VALUE BY CALLING THIS FUNCITON.**

****

**A template class can have virtual functions.**

**A function template cannot be a virtual function.**

**When using PURE VIRTUAL FUNCTIONS IN ABSTRACT CLASSES, virtual functions in base class must have the same PARAMETER, FUNCTION\_NAME AND RETURN VALUE(Exception: covariant return values…)**

**That’s what you call an override (completely the same)[重写]**

**While Overload happens when function names are the same but parameters are not[重载]**

**HOW TO READ A SENTENCE TYPED IN BY USER SAFELY**

1. **DON’T NEED TO WORRY ABOUT INPUT LENGTH;**
2. **ALLOW ‘SPACES’ TO EXIST AND SENTENCE ENDS AT 'ENTER’**

|  |
| --- |
| string whatToFind = "";  getline(cin,whatToFind); |

**HOW TO OPEN A FILE ELEGANTLY:**

|  |
| --- |
| try  {  fstream s;  string filename = "C:\\Users\\tongh\\Desktop\\shitHere\\file01.dat";  s.open(filename, ios::in | ios::ate | ios::\_Nocreate | ios::binary);  if (\_access(filename.c\_str(), 02) != 0)  {  throw - 1;  cerr << "file open erro!" << endl;  }  **s.seekg(ios::beg);//This is very important**  }    catch (int erro)  {  ;  } |

**想把一个类A声明为另一个类B的友元类，而A又含有B类的指针: 如何做**

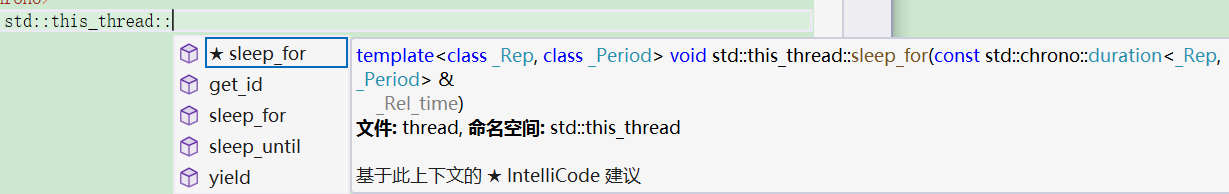
|  |  |
| --- | --- |
| **#include “classB”**  **class B; //前向声明**  **class A**  **{**  **…**  **}** | **#include “classA”**  **class A； //前向声明**  **class B**  **{**  **private:**  **A\* ptr;**  **}** |

**How to input sentences with whitespaces beautifully:**

|  |
| --- |
| **#include <string>**  **using namespace std;**  **//declare a function:** |
| cin.ignore(); //enter after selecting person type.  getline(cin, this->name);    cout << "\tAge:";  cin>>this->age;  cin.ignore(1);  cout << "\tSex:";  getline(cin, this->sex);//Getline helps you eat the ENTER pressed after the last typing.  //but cin>> don't. So you always have to add cin.ignore(1) after cin>>  //buf can safely use getlin(cin,somestring) before codes. |

**How to use “sleep” on different platforms(Win, Mac,Xnus)**

|  |
| --- |
| #include <thread>  #include <chrono>  std::this\_thread::sleep\_for(std::chrono::milliseconds(3500)); **//this makes current thread sleep for 3500 ms** |

****

**Enumerates are not basic data types.(enums are user-defined data structures, not integer groups).**

**So if you defined an enum to tag error types, you will throw enums and catch(enums) too.**

**If you throw an enum but catch an integer, this program will collapse**

**Use then like this instead:**

|  |  |
| --- | --- |
| **//ExecptionList.h**  const enum erroType  {  \_operation\_erro\_file\_creation\_erro= - 6,  \_operation\_erro\_selection\_erro = 0,  \_operation\_erro\_file\_open\_error= 1,  \_operation\_erro\_no\_read\_permission\_= - 3,  \_operation\_erro\_invalid\_io\_person\_type= - 4,  \_operation\_erro\_invalid\_occurrence\_number= - 5,  \_security\_offence\_wrong\_password= - 2,  }; | **//main.cpp** |

**myFile.write((char\*) &…, sizeof(….));**

**CAN ONLY WRITE int, double, float, char and their simple arrays like “char(\*)[] int(\*)[]” TO FILE.**

**IT CANNOT WRITE POINTERSTO FILE SAFELY.**

**EVEN IF IT WRITES POINTERS TO FILE, THESE DATA WILL SOON BECOME MEANINGLESS ONCE YOU READ THEM OUT. BECAUSE POINTERS’ VALUES ARE RANDOMLY ALLOCATED**

**AND THEY VARY THROUGH TIME.**

**HERE’S HOW TO WRITE ARRAYS SAFELY:**

|  |
| --- |
| #include <fstream>  #include <iostream>  using namespace std;  int main()  {    int myArray[10] = { };  int myBuffer[10] = { 0,0,0,0,0,0,0,0,0,0 };  fstream myFile;  myFile.open("C:\\Users\\tongh\\Desktop\\shit.dat", ios::out | ios::binary | ios::\_Nocreate);  myFile.seekp(ios::beg);  myFile.write((char\*)&myArray,sizeof(myArray));  myFile.seekg(ios::beg);  myFile.read((char\*)&myBuffer, 40);    for(int i=0;i<10;i++)  cout<<myBuffer[i];  //cout<<myBuffer;  } |
| NOTE THAT if you write “cout<<myBuffer “,it will only print the address of myBuffer; |

|  |
| --- |
| **std::string & POD conversion functions**  float \_floatBuf =std::stof(strbuf);  double \_doubleBuf =std::stod(strbuf);  int \_intBuf =std::stoi(strbuf);        std::string strbuf =std::to\_string(\_intBuf);  or float, long,long double, unsigned long , long lonng, ...  **to \_string has 9 overloads.** |
| Read and write functions only write char s to file.  These chars don’t look the same as integers in binary files, even if they were ints before  When you get an integer \_inT;  If you write it to file as an integer:  cin<<\_inT;  and search it as an integer: pattern = (int) \_inT;  you can find it.  But if you search it as a string(when you receive user-input data from the console, which is always chars and strings…)  Then you can’t find it.  If you write it to file as a char  somefile.write((char\*)&\_inT,sizeof(int));  thins are bad.  If you use  cin >> \_intBuf;  string strbuf = std::to\_string(\_intBuf);  someFile.write(strbuf.c\_str(), sizeof(strbuf));  then you can find this integer in file by finding occurrences of a string \_pattern which you use to contain user-input data from console, not worrying its data type.  cout<<”what to find?\n”;  string \_pattern=””;  cin>>string;  the lines above can apply to other application scenarios like inputting a char, double, or long integer, … |

**\*\*\*\*operator new returns a pointer to its parameter.**

**Noder<T> \* p = new Node<T>; //Note that RHS has a ‘\*’star while RHS doesn’t have it.**

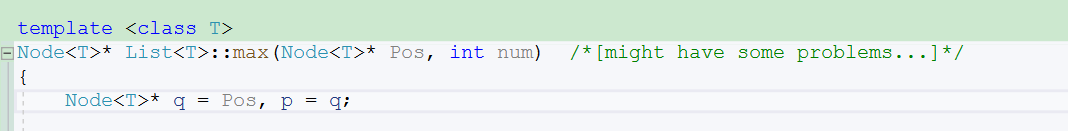
**How to defince function templates in a template class**

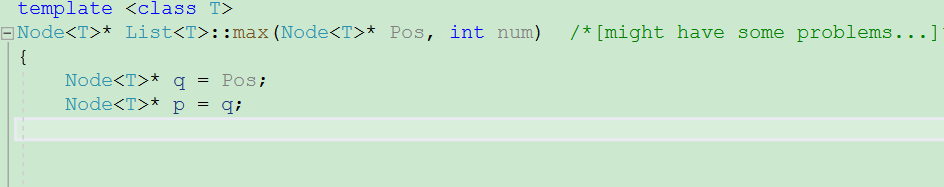
|  |
| --- |
| template <class T> template<class VisitingFunctor>  void List<T>::for\_each(VisitingFunctor& functor)  {  Node<T>\* p = ;  for (p = this->\_head; p != this->\_tail; p = p->\_successor)  functor(p->\_node\_data);  } |
| ***Note that the parameter indicating class type and para indicating function type should be declared separately. Or else template<class T, VisitingFunctor> will be seen as declaring these paras for another template class with 2 template paras.*** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Functors(functional objects)**  **Are the classes that are equipped with**  **RetV operator() (Parameter list…);**  **This reloaded operator() can receive multiple paras and return any data type u defined.**  **Simple (classical ) operators can also be seen as (and used as) functors.**  **How to define functors as a class:**   |  | | --- | | **//Classes:**  class Condition  {  public:  template<class T>  bool operator() (const T& item,const T& target) { return item>=target; };  };  class MultiplicationOperator  {  public:  MultiplicationOperator(double i = 0) :multiplier(i){};  bool operator()(double& toBmultiplied)  {  toBmultiplied \*= multiplier;  return true;  }  private:  double multiplier;  };  **//Objects:**  Condition biggerThan;  MultiplicationOperator becomesNegative(-1); |   **How to make functors a parameter:**  **!@declaration:**   |  | | --- | | template <class VisitingFunctor>  Node<T>\* searchBy(VisitingFunctor& condition,const T& targetData); |   **!@definition(define a member function of a template class with a functor parameter.**   |  | | --- | | template <class T> template <class VisitingFunctor>  Node<T>\* List<T>::searchBy(VisitingFunctor& condition,const T& targetData,Node<T>\* pos)  {  if(condition((pos = pos->\_predecessor)->\_node\_data, targetData))  return pos;  return nullptr;  } |   **!@ calling:**   |  | | --- | | myList.searchBy(*biggerThan*,114514);  //use user-defined operators(functors)at this time, parameter is object name. | |  | |

**WARNING:**

**Don’t write too concisely. Let the compiler know.**

****

****

**Normally, function pointers can only be used as pointers to GLOBAL FUNCTIONS(don’t use them)**

**or STATIC MEMBER FUNCTIONS**

**NON-STATIC MEMBER FUNCTIONS ARE NOT ORDINARY FUNCTION TYPE.**

**like**

**clas SomeClass**

**{**

**void fun() ;**

**}**

**is not a void(\*)() function.**

**it’s a void (SomeClass:\*)() function.**

**But when you try to use template to write a method to deal with this extraordinary function,**

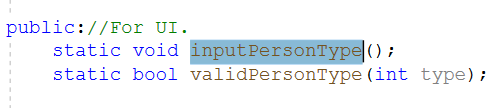
**compiler will fail to compile, because not all para T has that member.**

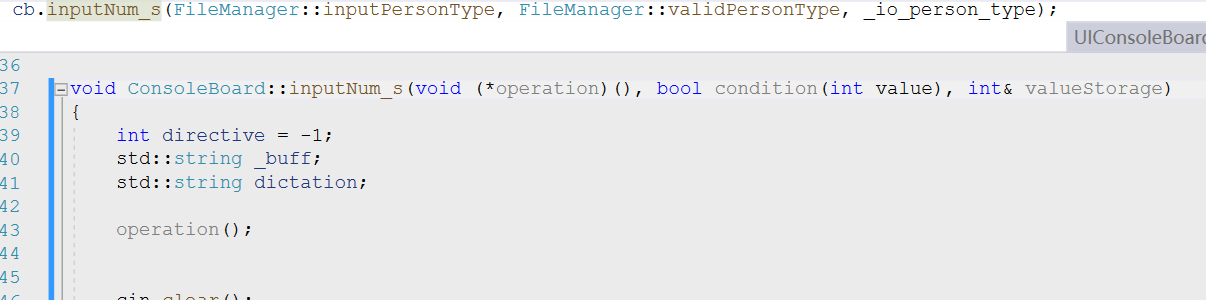
**Probably using functors can solve this?**

**Didn’t try it.**

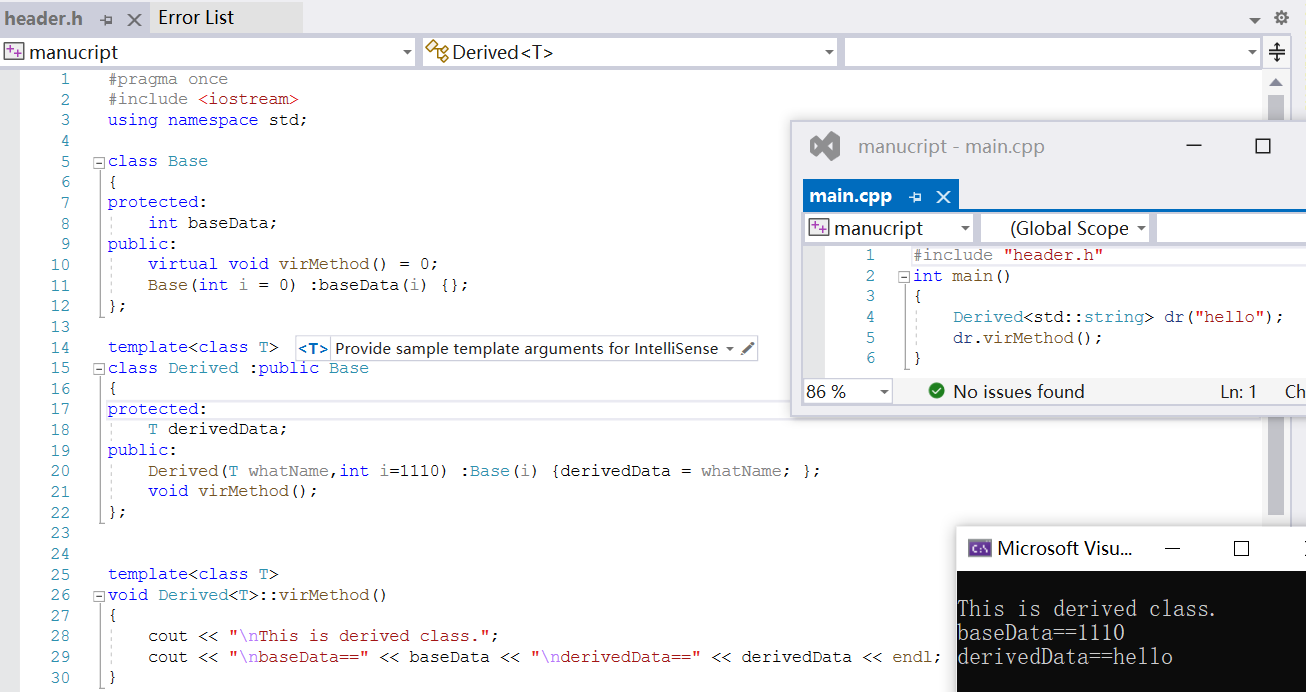
**But this one(declaring that extraordinary function as a STATIC FUNTION)**

**works:**

****

****

**how to derive a template class from a normal class:**

****

|  |
| --- |
| //header.h  #pragma once  #include <iostream>  using namespace std;  class Base  {  protected:  int baseData;  public:  virtual void virMethod() = 0;  Base(int I = 0) :baseData(i) {};  };  template<class T>  class Derived :public Base  {  protected:  T derivedData;  public:  Derived(T whatName,int i=1110) :Base(i) {derivedData = whatName; };  void virMethod();  };  template<class T>  void Derived<T>::virMethod()  {  cout << “\nThis is derived class.”;  cout << “\nbaseData==” << baseData << “\nderivedData==” << derivedData << endl;  } |
| //main.h  #include "header.h"  int main()  {  Derived<std::string> dr("hello");  dr.virMethod();  } |

**when designing a class template, you can write some specific member functions of certain type T s and compile correctly, As long as that type T are the only classes that will use this template.**

**because templates are nothing before compiling.**

**They become “real” only after class type T is specified.**

**so you can write a member function passed down in a class family and use this template for all the descendants.**

|  |
| --- |
| **how you should pass pointers between functions:**   1. **if you ONLY WANT READ PERMISSION TO THIS POINTER,**   ***return an address and receive it by a pointer variable.***  **Then you *read the data* this address points to (you need to verify that this pointer !=nullptr)**  **like if this address belong to some object or more complicated data structures, you can**  **use pt-> to call all of its owner’s public functions. (this is READ permission)**   1. **if you want READ AND WRITE PERMISSION TO THIS POINTER,**   ***return a REFERENCE TO POINTER and receive it by REFERENCE TO POINTER***  **Then you can CHANGE THE ORIGINAL POINTER(like MOVING THIS POINTER or MAKE IT POINT TO OTHER places) IN ANOTHER FUNCTION .**  **These operations include assignment of pointers so it NEEDS WRITE PERMISSION TO POINTER.**   1. **NEVER pass and receive a pointer and then try to move/change the original pointer in another function.**   **in that case, the pointer on your hand is only a read-only copy and a temporary variable.**  **Even though you can move it in current function and call functions at other places,**  **You will never make any tangible influence on the original pointer and its owner.** |
|  |
| |  |  | | --- | --- | |  |  | |