

CS100 Introduction to Programming

Recitation 8

llk89

NO PLAGIARISM!!!

- The most likely cause for failing this course.
- You WILL be caught!
- We WILL punish!
- They WILL know!
 - Parents
 - University
 - School
 - Fellows

Disclaimer

- I do not use Visual Studio, nor MacOS
- Directives for Windows and MacOS can therefore be wrong and damage your computer
- Use at your own peril, or somehow get a Ubuntu instance

Clone the skeleton code

- `git clone https://github.com/11k89/cs100recitation8`
- This will be used through out the entire recitation

Overview

- Shared pointers
- friend
- References, const & namespace
- Solving linear systems with C++

Shared Pointers

Location: `shared_pointer/`

Shared Pointers

- Imagine the following simple Test class
 - The class does not do much except notifying us about construction and destruction

```
class TestClass {  
public:  
    TestClass() {  
        std::cout << "dummy object is created\n"; }  
    virtual ~TestClass() {  
        std::cout << "dummy object is destroyed\n"; }  
    void printSomething() { std::cout << "hello\n"; }  
private:  
    double m_dummy;  
};
```

Shared Pointers

- The main function

```
int main() {  
    int indexA = 1;  
    int indexB = 1;  
    if( indexA == 1) {  
        std::cout << "Entering scope A\n";  
        SharedPointer<TestClass> ptrA;  
        if( indexB == 1 ) {  
            std::cout << "Entering scope B\n";  
            SharedPointer<TestClass> ptrB( new TestClass() );  
            ptrA = ptrB;  
        }  
        std::cout << "Returning to scope A\n";  
    }  
    std::cout << "Returning to global scope\n";  
    return 0;  
}
```

Custom shared-pointer class

用 make_shared

Shared Pointers

- Typical shared pointer declaration

```
template<class T>
class SharedPointer {
public:
    SharedPointer();
    SharedPointer( T * ptr );
    SharedPointer( const SharedPointer & ptr );
    virtual ~SharedPointer();

    SharedPointer & operator=( const SharedPointer & ptr );

    T * get();
    T & operator*();
    T * operator->();
private:
    T * m_ptr;
    SharedPointerController * m_controller;
};
```

Remember: A shared pointer takes two members, a pointer to the underlying variable, and a pointer to a control structure with reference counter

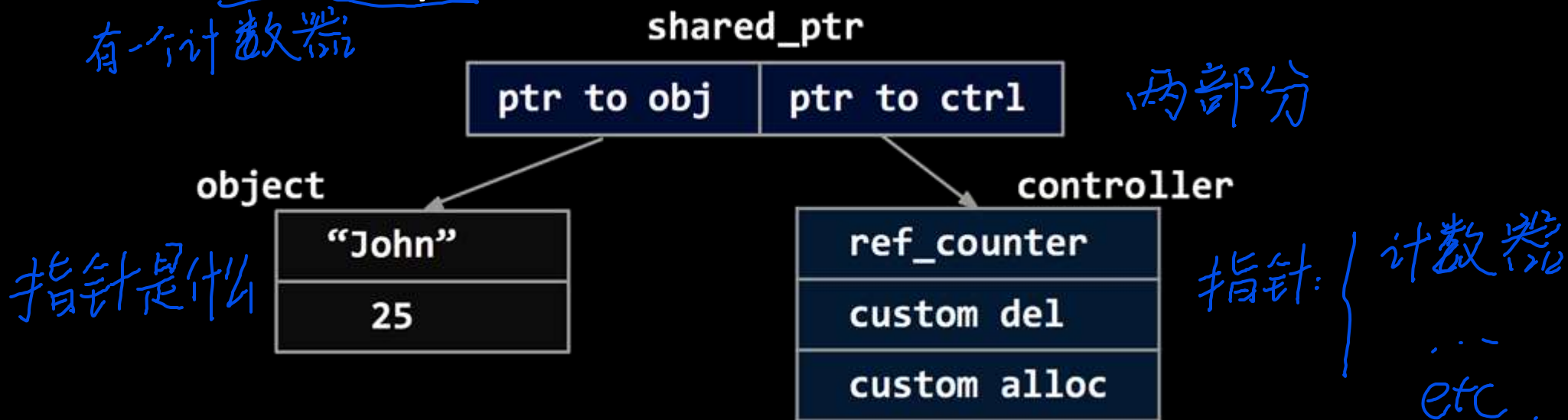
有两部分

std::shared_ptr

自动垃圾回收 (在return 0 前会
释放内存)

- Shared pointers have a garbage collection mechanism based on a reference counter contained in a control block
- Each new shared owner copies the pointer to the control block and increases the count by 1

有一个计数器



- 使用时: `shared_ptr<Test> ptest = make_shared<Test>(...);`

Shared pointers

一共四个智能指针

- Tasks:
 - Implement SharedPointerController
 - Implement destructor of SharedPointer

friend

Location: friend/

友元: { 函数
 类

一共四种

Using friend

- Look at this vector

```
class MyVector {  
public:  
    MyVector();  
    virtual ~MyVector();  
  
    double & at( int index );  
    const double & at( int index ) const;  
    void clear();  
    int size() const;  
    MyVector & push_back( double val );  
private:  
    std::vector<double> m_data;  
};
```

Using friend

- Task
 - Make it possible to stream a MyVector to an `std::ostream` (base class of `std::ofstream`, `std::cout`, `std::stringstream`, etc.)
 - Make it possible for this function to have direct access to MyVector's data (by using friend)

References, const & namespace

Location: map/

Fix up old code

- Finish skeleton code in `LinkedMap.hpp`
 - This skeleton is wrong in almost every sense, and incomplete
 - The only thing correct is existing names and main function

- Compile with

```
D:\>cl /c /Wall main.cpp // May not work!
```

```
llk89@athena: ~$ g++ --std=c++11 -c main.cpp -o main.o
```

- This is not executable...
 - All you need to do is to make it pass the type checker

Special requirements

- You may add new functions
- You should not rename existing classes and functions
- Add as many const as you can
 - Do not add const that has no effect
- Use references wherever possible
- Adjust main.cpp as needed
 - Do not remove any const from main.cpp
 - Do not remove any line from main.cpp

Check

- You should have added
 - 2 new functions
 - 9+ consts
 - many “&”
 - 1 friend
 - 1 new nested class with its functions
- You should be able to compile main.cpp
- You should have somehow fixed the name collision

How big are these?

No skeleton

Instruments

- `sizeof()`: a compile time function-like syntax construct to figure out the size of passed in element
- `"a" "b"`: In C/C++, two adjacent string literal will be automatically concatenated together
 - e.g. `printf("This is " "the same string!")`
- `#`: Convert the item to a string literal
 - e.g. `#define msg(x) printf("This is " #x "!")`

Exersice

- Print the size of
 - char
 - short
 - short int
 - int
 - long int
 - unsigned int
- Remember to include cstdint and ctime
- Less than 3 sizeof.
 - It could be done with only one sizeof

Solving large* linear systems

Location: [eigen/](#)

*: dimension > 1000

Project Specification

- Read dimension n from `stdin`
- Cap n at 4000 if a certain macro is defined at compile time
- Generate a random $n \times n$ matrix
- Generate a random vector from R^n
- Solve the linear system fast
- Print the time used to solve the linear system

Design considerations

- Matrix operations can be expensive
- A poor implementation could lead to severe performance degradation
 - e.g. from $<0.1s$ to $>100s$
- Solution: Learn ~~linear algebra, algorithms, data structures, computer architecture, parallel computing, CUDA, and become an expert~~
- Matrix operations is very common
- There *must* be some very brilliant implementation out there already
- Solution: Use [Eigen](#), a high level C++ library for linear algebra

Install Eigen3 - Ubuntu

- `sudo apt-get update`
- `sudo apt-get install -y libeigen3-dev`

Install Eigen3 - MacOS

- Install homebrew
- `brew install eigen`

Install Eigen3 - Windows

- Download the desired release from <https://eigen.tuxfamily.org>.
- Extract the inner folder and rename it to Eigen3 or Eigen
- Put that folder at C:\
 - You can put it elsewhere as long as you can make CMake to find it

CMakeLists.txt

- Open eigen/CMakeLists.txt in cloned repo
- Explain the file
- Check other CMakeLists.txt in the cloned repo
- Make sure you understand them as well

Build and Execute

- mkdir build
 - cd build
 - cmake ..
 - make
 - ./eigen
-
- Input 4000 and hit enter, so Eigen starts working
 - The program seems to hang...
 - A bug?

Debug vs Release

- By default, CMake compiles in debug mode
 - optimizer off
 - add debug symbols
 - prefer compile speed over anything else
- Use `-DCMAKE_BUILD_TYPE=Release` to compile with release mode
 - optimizer on
 - no debug symbols
 - prefer execution speed
- Try it now. It should take less than 10 seconds to finish now.
 - You might need to consider renting a VPS for study if it goes over 10 seconds

add_definition()

- add_definition() is the cmake way to provide preprocessors with macro info
 - e.g. add_definition(-DCAP_N) is equivalent to gcc -DCAP_N
- Try implement the cap on n functionality now
- It is taboo to use set(CMAKE_CXX_FLAGS "-DCAP_N")
 - This works, why it is not allowed?

option()

- You don't always want to have the cap
- Changing CMakeLists.txt to get another executable is tedious
- How about let cmake accept an option to configure it?

option()

- Use cmake command
 `option(<option_name> "documentation" [initial_value])`
 - Default to OFF unless `initial_value` specified
 - Use `cmake -Doption_name=ON/OFF` to change at command line
 - Use `if(option_name) ... endif(option_name)` to check
- Try it now!

Shipping both version

- Some want binary with cap off, some want the other
- To distribute both version, both binaries need to be built.
- Tedious to build one with cap on and another with cap off
- Best to combine both in the same CMakeLists.txt
- Add another target called `eigen_test_capped`

target_compile_definition()

- Specify macro definition on a per target basis
 - `target_compile_definitions(<target> <INTERFACE|PUBLIC|PRIVATE> [item1...])`
- Use PRIVATE for now
- Use PUBLIC when you are writing a library
- Try it now!

Installer

- Tasks:
 - install()s
 - CPack

QA Time

- If you have any problems with...
 - last week's lecture
 - Git
 - CMake
 - Homework
 - Recitation 8
- Ask now