Оглавление

[Std::stream 1](#_Toc143858973)

[Конструкторы и операторы копирования и перемещения 1](#_Toc143858974)

[GUI Библиотеки 2](#_Toc143858975)

[ResourceAcquisitionIsInitialization 2](#_Toc143858976)

[Scoped guard 2](#_Toc143858977)

[Copy and Swap 2](#_Toc143858978)

[Copy on Write 2](#_Toc143858979)

[Интерфейсный класс 2](#_Toc143858980)

[Empty Base Optimization 2](#_Toc143858981)

[CRTP – Curiously Recurring Template Pattern 3](#_Toc143858982)

[Pointer on function 3](#_Toc143858983)

[Self-assignment check 3](#_Toc143858984)

[Pimpl 3](#_Toc143858985)

[Singleton 4](#_Toc143858986)

[Virtual constructor 4](#_Toc143858987)

[Mixin 4](#_Toc143858988)

[Enable\_shared\_from\_this 4](#_Toc143858989)

[SFINAE 4](#_Toc143858990)

# Std::stream

Operator << (std::stream &out, const myclass& class1) { // перегрузка потока

Std::cout << “поле1: “ << class1.field1 << “ поле2: “ << class1.field2;

return out; }

void myPrint (std::ostream &stream, int other\_var);

myPrint(std::cout, 42);

void myPrint(std::ostream &stream, int v){

stream << “answer is: ” << v;}

std::ifstream input{argv};

std::for\_each(std::istream\_iterator<std::string>(input),

std::istream\_iterator<std::string>(), [](){//lyambda});

# Конструкторы и операторы копирования и перемещения

Class s{

//нельзя менять arg (его значения, поля и т.п.)

но поля класса s с помощью my\_function изменить можно.

void my\_function(const MyType & arg); //применяется. чтобы избежать копирования

void my\_function(MyType & arg) const; //тут можно менять arg, но нельзя изменить поля класса s

}

Вместе с конструктором move использовать noexept, иначе он вместо мув будет делать копи на всякий случай, чтоб откатить если произойдет исключение

# GUI Библиотеки

>- какие библиотеки используют?  
Нативные под разные ОС (WPF или WInUI для windows, Qt(KDE) или GTK(Gnome) под Linux Cocoa под MacOS  
Кросс-платформенные: Qt, GTK  
Кросс-языковые (часть отвечающая за UI на одном языке, а основная логика приложения на С++): Flutter, Electron, etc

>- какие есть со свободной лицензией?  
Я так понимаю имеется в виду свободная для коммерческого использования?  
GTK выпускается под LGPL (там есть несколько нюансов, но если соблюдать условия LGPL, то можно выпускать коммерческое ПО без необходимости распространять его исходные коды)  
  
Flutter под BSD-3  
>- какие платные, но стоят не очень дорого с возможностью коммерческого использования?  
Qt. Но сейчас не продается для РФ  
Для WPF и WinUI нужно будет приобрести Visual Studio Prof, Для Cocoa кажется достаточно подписать на Apple Developer Account.

# ResourceAcquisitionIsInitialization

Суть: захват ресурса в конструкторе объекта, освобождение в деструкторе. Пример универсального RAII - std::unique\_ptr

# Scoped guard

RAII обертка над не RAII типами. Самописное.

# Copy and Swap

m\_class &operator= (const m\_class& other)// {delete [] m\_data; опасно. Если swap до конца не m\_size = other.m\_size; m\_data = new int[m\_size]; } //сработает, а m\_data уже потеряна

m\_class A{42}; m\_class B{56}; A=B;

Поэтому сначала копируем в темп, потом делаем замену, потом удаление.

# Copy on Write

Если надо передать копию объекта (вектор например или экземпляр класса) передастся все-равно ссылка. Но при первом изменении, сначала произойдет копирование, а потом изменение.

# Интерфейсный класс

Это класс, у которого все методы пустые и виртуальные. Например struct adapter {virtual void open()=0; virtual void close() = 0;};

# Empty Base Optimization

Кучу мелких классов (например struct1 и 2 без данных, просто что-то печатает. Даже без данных он должен что-то занимать в памяти – поэтому будет 1 байт) прячем в одном пустом классе с помощью **приватного** (чтобы снаружи никто не смог вызвать) наследования. Экономим память и уменьшаем строчки кода.

|  |  |
| --- | --- |
| Было    sizeof(other) = 8 (m\_value 4 байта; str1 1 байт; str2 1байт и выравнивание добьет до слова – итого будет 8 байт) | Стало    sizeof(other) = 4 (value 4 байта и все. Struct не занимают места) |

# CRTP – Curiously Recurring Template Pattern

Странно рекурсивный шаблон в классе Eye. Это мета программирование (т.е. пишет код С++ на С++ для С++ во время компиляции). Берем этот класс (this) и с помощью статик каста приводим к нужному типу и вытаскиваем нужный мембер (eye\_open). Потом делаем mixin, т.е. подмешиваем это в класс циклопа. По сути делаем наследника от циклопа, который обратится к нужным методам eye

Eye – интерфейсный класс, look\_for\_banana – функции объектов имеющие интерфейс, Cyclop – объект. Это все статический (компайл тайм) полиморфизм. Можно все это на виртуальные методы заменить, будет динамический (ран тайм) полиморфизм, но в ран тайме будет работать медленнее.

CRTP позволяет подмешать в чужой класс (например в класс stl) свои собственные методы не трогая и не переопределяя, не меняя иерархию, не перегружая ни свой код ни чужие классы.

# Pointer on function

int (\*iccid)(int) = iccid\_code(){};

int (\*fuel\_type)(int) = fuel\_type\_code(){};

return iccid(data);

# Self-assignment check

|  |  |
| --- | --- |
| Если по коду случайно произойдет что-то типа a>x ? a=a : a=b то будет беда |  |

# Pimpl

Pointer to implementation. Скрываем класс. Обращаемся к нему с помощью указателя. Интерфейс скрытого класса не меняем, а в реализации возможны изменения. Плюсы: время компиляции, стабильный интерфейс. Минусы: оптимизация компилятора между пимплом и остальным кодом не работает.

|  |  |
| --- | --- |
| struct my\_class {  public:  explicit my\_class(const char \* path);  ~my\_class();  size\_t do\_work() const;  void do\_other\_work(int param);  private:  struct impl;  impl\* m\_impl; // pimpl  }; | Cpp file  struct my\_class::impl {  //some code  }  my\_class::my\_class(const char \*path) {  //some code  } |

# Singleton

// Singleton.h

**class** Singleton

{

**private**:

**static** Singleton \* p\_instance;

    // Конструкторы и оператор присваивания недоступны клиентам

    Singleton() {}

    Singleton( **const** Singleton& );

    Singleton& operator=( Singleton& );

**public**:

**static** Singleton \* getInstance() {

**if**(!p\_instance)

            p\_instance = **new** Singleton();

**return** p\_instance;

    }

};

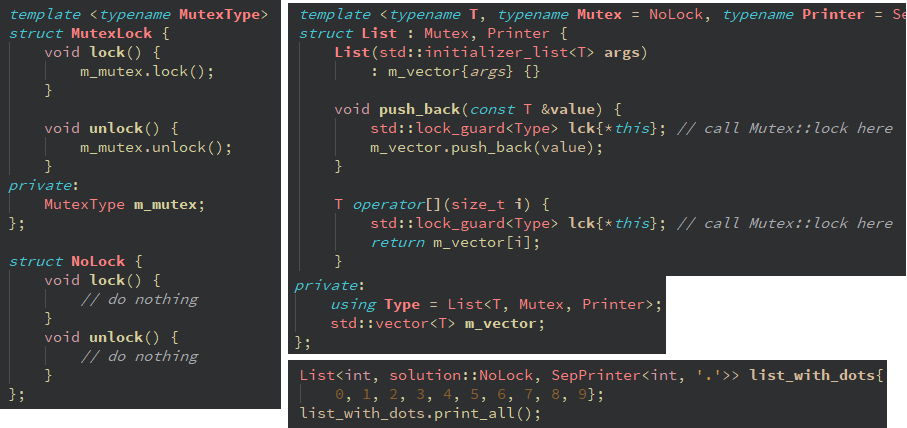
// Singleton.cpp

#include "Singleton.h"

Singleton\* Singleton::p\_instance = 0;

# Virtual constructor

# Mixin

Virtual (foo) owerride – динамический полиморфизм, занимает ран-тайм время. Можно заменить на статический полиморфизм с помощью mixin.

# Enable\_shared\_from\_this

# SFINAE

Запретить инстанциравать шаблон, если тип ТТ равен My\_list – тогда компилятор вызовет ошибку инстанцирования и решит что explicit My\_list(T element){} не подходит, и пойдет дальше. Возьмет не так точно подходящее: My\_list(const My\_list &rhs){}

