



С++ - Модуль 07

Шаблоны С++

Резюме:

*Этот документ содержит упражнения модуля 07 из модулей
С++.*

Версия: 6

Содержание

I	Введение	2
II	Общие правила	3
III	Упражнение 00: Начните с нескольких функций	5
IV	Упражнение 01: Итер	7
V	Упражнение 02: Массив	8

Глава I Введение

С++ - это язык программирования общего назначения, созданный Бьярном Струstrupом как продолжение языка программирования С, или "С с классами" (источник: [Википедия](#)).

Цель этих модулей - познакомить вас с **объектно-ориентированным программированием**. Это будет отправной точкой вашего путешествия по С++. Многие языки рекомендуются для изучения ООП. Мы решили выбрать С++, поскольку он является производным от вашего старого друга С. Поскольку это сложный язык, и для того, чтобы все было просто, ваш код будет соответствовать стандарту С++98.

Мы понимаем, что современный С++ во многих аспектах сильно отличается. Поэтому, если вы хотите стать квалифицированным разработчиком С++, вам предстоит пройти дальше 42 Common Core!

Глава II Общие правила

Компиляция

- Скомпилируйте ваш код с помощью `c++` и флагов `-Wall -Wextra -Werror`
- Ваш код будет компилироваться, если вы добавите флаг `-std=c++98`

Форматирование и соглашения об именовании

- Каталоги упражнений будут называться так: `ex00`, `ex01`, ... , `exp`
- Назовите свои файлы, классы, функции, функции-члены и атрибуты в соответствии с требованиями руководства.
- Записывайте имена классов в формате **UpperCamelCase**. Файлы, содержащие код класса, всегда будут именоваться в соответствии с именем класса. Например: `ClassName.hpp/ClassName.h`, `ClassName.cpp` или `ClassName.tpp`. Тогда, если у вас есть заголовочный файл, содержащий определение класса "BrickWall", обозначающего кирпичную стену, его имя будет `BrickWall.hpp`.
- Если не указано иное, каждое выходное сообщение должно завершаться символом новой строки и выводиться на стандартный вывод.
- *До свидания, Норминет!* В модулях C++ нет принудительного стиля кодирования. Вы можете следовать своему любимому стилю. Но имейте в виду, что код, который ваши коллеги-оценщики не могут понять, они не могут оценить. Делайте все возможное, чтобы писать чистый и читабельный код.

Разрешено/Запрещено

Вы больше не кодируете на C. Пора переходить на C++! Поэтому:

- Вам разрешено использовать почти все из стандартной библиотеки. Таким образом, вместо того чтобы придерживаться того, что вы уже знаете, было бы разумно использовать как можно больше C++-версий функций языка C, к которым вы привыкли.
- Однако вы не можете использовать никакие другие внешние библиотеки. Это означает, что библиотеки C++11 (и производные формы) и Boost запрещены. Также запрещены следующие функции: `*printf()`, `*alloc()` и `free()`. Если вы их используете, ваша оценка будет 0 и все.

- Обратите внимание, что если явно не указано иное, используемое пространство имен `<ns_name>` и ключевые слова-друзья запрещены. В противном случае ваша оценка будет равна -42.
- **Вам разрешено использовать STL только в модуле 08.** Это означает: никаких **контейнеров** (вектор/список/карта/и так далее) и никаких **алгоритмов** (все, что требует включения заголовка `<algorithm>`) до этого момента. В противном случае ваша оценка будет -42.

Несколько требований к дизайну

- Утечка памяти происходит и в C++. Когда вы выделяете память (с помощью функции `new` ключевое слово), вы должны избегать **утечек памяти**.
- С модуля 02 по модуль 08 ваши занятия должны быть построены в **православной канонической форме, за исключением случаев, когда прямо указано иное**.
- Любая реализация функции, помещенная в заголовочный файл (за исключением шаблонов функций), означает 0 для упражнения.
- Вы должны иметь возможность использовать каждый из ваших заголовков независимо от других. Таким образом, они должны включать все необходимые зависимости. Однако вы должны избегать проблемы двойного включения, добавляя **защитные элементы include**. В противном случае ваша оценка будет равна 0.

Читать

- Вы можете добавить несколько дополнительных файлов, если это необходимо (например, для разделения вашего кода). Поскольку эти задания не проверяются программой, не стесняйтесь делать это, если вы сдаете обязательные файлы.
- Иногда указания к упражнению выглядят кратко, но на примерах можно увидеть требования, которые не прописаны в инструкциях в явном виде.
- Перед началом работы полностью прочитайте каждый модуль! Действительно, сделайте это.
- Одином, Тором! Используйте свой мозг!!!



Вам придется реализовать множество классов. Это может показаться утомительным, если только вы не умеете писать сценарии в своем любимом текстовом редакторе.




Вам предоставляется определенная свобода в выполнении упражнений. Однако соблюдайте обязательные правила и не ленитесь.
полезной информации!

Иначе вы пропустите много
Не стесняйтесь читать о

Глава III

Упражнение 00: Начните с нескольких функций

	Упражнение : 00
Начните с нескольких функций	
Входящий каталог : <code>ex00/</code>	
Файлы для сдачи : <code>Makefile, main.cpp, whatever.{h, hpp}</code>	
Запрещенные функции : Нет	

Внедрите следующие шаблоны функций:

- `swap`: Меняет местами значения двух заданных аргументов. Ничего не возвращает.
- `min`: Сравнивает два значения, переданные в аргументах, и возвращает наименьшее из них. Если два значения равны, то возвращается второе.
- `max`: Сравнивает два значения, переданные в аргументах, и возвращает наибольшее из них. Если два значения равны, то возвращается второе.

Эти функции можно вызывать с аргументами любого типа. Единственное требование - два аргумента должны иметь одинаковый тип и поддерживать все операторы сравнения.



Шаблоны должны быть определены в заголовочных файлах.

Выполняем следующий код:

```
int main( void ) {  
  
    int a = 2;  
    int b = 3;  
  
    ::swap( a, b );  
    std::cout << "a = " << a << ", b = " << b << std::endl; std::cout <<  
    "min( a, b ) = " << ::min( a, b ) << std::endl;  
    std::cout << "max( a, b ) = " << ::max( a, b ) << std::endl;  
  
    std::string c = "chaine1"; std::string d =  
    "chaine2";  
  
    ::swap( c, d );  
    std::cout << "c = " << c << ", d = " << d << std::endl; std::cout <<  
    "min( c, d ) = " << ::min( c, d ) << std::endl;  
    std::cout << "max( c, d ) = " << ::max( c, d ) << std::endl;  
  
    вернуть 0;  
}
```


Должен вывести:

```
a = 3, b = 2  
min(a, b) = 2  
max(a, b) = 3  
c = chaine2, d = chaine1  
min(c, d) = chaine1 max(c,  
d) = chaine2
```

Глава IV

Упражнение 01:

Итер

	Упражнение : 01
	Iter
	Входящий каталог : <i>ex01/</i>
	Файлы для сдачи : Makefile, main.cpp, iter.{h, hpp}

Реализуйте шаблон функции `iter`, которая принимает 3 параметра и ничего не возвращает.

- Первый параметр - это адрес массива.
- Второй - длина массива.
- Третья - это функция, которая будет вызываться на каждом элементе массива.


Создайте файл `main.cpp`, содержащий ваши тесты. Предоставьте достаточно кода для создания исполняемого файла теста.

Ваш шаблон функции `iter` должен работать с любым типом массива. Третьим параметром может быть инстанцированный шаблон функции.

Глава V

Упражнение 02:

Массив

	Упражнение : 02
	Массив
	Входящий каталог : <i>ex02/</i>
	Файлы для сдачи : <i>Makefile, main.cpp, Array.{h, hpp}</i> и дополнительный файл: <i>Array.tpp</i>
	Запрещенные функции : Нет

Разработайте шаблон класса **Array**, который содержит элементы типа **T** и реализует следующее поведение и функции:

- Конструкция без параметра: Создает пустой массив.
- Конструкция с `unsigned int n` в качестве параметра: Создает массив из `n` элементов, инициализированный по умолчанию.
Совет: Попробуйте скомпилировать `int * a = new int();` затем выведите `*a`.
- Построение с помощью копирования и оператора присваивания. В обоих случаях изменение исходного массива или его копии после копирования не может повлиять на другой массив.
- Вы ДОЛЖНЫ использовать оператор `new[]` для выделения памяти. Превентивное выделение (заранее определяющее местоположение памяти) запрещено. Ваша программа никогда не должна обращаться к нераспределенной памяти.
- Доступ к элементам можно получить с помощью оператора subscript: `[]`.
- При обращении к элементу с помощью оператора `[]`, если его индекс выходит за пределы, возникает ошибка возникает исключение `std::exception`.
- Функция-член `size()`, которая возвращает количество элементов в массиве. Эта функция-член не принимает никаких параметров и не может изменять текущий экземпляр.

Как обычно, убедитесь, что все работает, как ожидалось, и сдайте файл `main.cpp`, содержащий ваши тесты.