# Предисловие

Данное методическое руководство предназначено для студентов

специальности «Информатика и вычислительная техника», выполняющих курсовую работу по программированию на языке С++.

Тематика курсовой работы — реализовать универсальную шаблонную коллекции. Для проверки работоспособности коллекции написать тестирующую программу, в которой функция вызывается для данных целого, вещественного и длинного целого типа. При выполнении курсовой работы используются знания, полученные студентами в курсе программирования на языке С++. В руководстве приведена вся необходимая информация о проектировании и реализации коллекции.

Курсовая работа является наиболее сложной компонентой обучения,

требующей от студентов целеустремленности, высокой трудоспособности и

дисциплины.

Успешное выполнение курсовой работы свидетельствует о полученных теоретических и практических знаниях по курсу программированию на языке С++.

# 1. Постановка цели и задач курсовой работы

# 1.1. Введение в обобщенное программирование

Обобщённое программирование — парадигма программирования, заключающаяся в таком описании данных и алгоритмов, которое можно применять к различным типам данных, не меняя само это описание.

В языке C++ имеются данные различных типов — целые числа, числа с дробной частью, символы, строки символов, определяемые пользователем сложные структуры, состоящие из данных нескольких типов. Если, например, требуется сортировать данные различных типов, то обычно для каждого типа создается отдельная функция сортировки. Обобщенное программирование расширяет язык таким образом, что позволяет один раз написать функцию для обобщенного (т.е. неопределенного) типа данных и затем использовать ее для разнообразных реальных типов данных. Это обеспечивается с помощью шаблонов языка C++.

Средства обобщённого программирования реализуются в языках программирования в виде тех или иных синтаксических средств, дающих возможность описывать данные (типы данных) и алгоритмы (процедуры, функции, методы), параметризуемые типами данных. У функции или типа данных явно описываются формальные параметры-типы. Это описание является обобщённым и в исходном виде непосредственно использовано быть не может.

В тех местах программы, где обобщённый тип или функция используется, программист должен явно указать фактический параметр-тип, конкретизирующий описание. Например, обобщённая процедура перестановки местами двух значений может иметь параметр-тип, определяющий тип значений, которые она меняет местами. Когда программисту нужно поменять местами два целых значения, он вызывает процедуру с параметром-типом «целое число» и двумя параметрами — целыми числами, когда две строки — с параметром-типом «строка» и двумя параметрами — строками. В случае с данными программист может, например, описать обобщённый тип «список» с параметром-типом, определяющим тип хранимых в списке значений. Тогда при описании реальных списков программист должен указать обобщённый тип и параметр-тип, получая, таким образом, любой желаемый список с помощью одного и того же описания.

Компилятор, встречая обращение к обобщённому типу или функции, выполняет необходимые процедуры статического контроля типов, оценивает возможность заданной конкретизации и при положительной оценке генерирует код, подставляя фактический параметр-тип на место формального параметра-типа в обобщённом описании. Для успешного использования обобщённых описаний фактические типы-параметры должны удовлетворять определённым условиям. Если обобщённая функция сравнивает значения типа-параметра, любой конкретный тип, использованный в ней, должен поддерживать операции сравнения, если присваивает значения типа-параметра переменным — конкретный тип должен обеспечивать корректное присваивание.

# 1.2. Понятие шаблона

Шабло́ны (англ. template) — средство языка C++, предназначенное для кодирования обобщённых алгоритмов, без привязки к некоторым параметрам (например, типам данных, размерам буферов, значениям по умолчанию).

Шаблоны позволяют использовать одни и те же функции или классы для обработки данных разных типов.

Концепция шаблонов может быть использована в двух видах: по отношению к функциям и по отношению к классам.

Описание шаблона начинается с ключевого слова template за которым в угловых скобках («<» и «>») следует список параметров шаблона. Далее идет объявление шаблонной сущности (например, функция или класс).

Ключевое слово template сообщает компилятору о том, что мы определяем шаблон функции. Ключевое слово, заключенное в угловые скобки, называется аргументом шаблона. Внутри всего определения шаблона любой конкретный тип данных, такой, как int, заменяет аргумент Т.

Пример:

1. Шаблон функции

template <class Т>

T abs(Т n)

{

return (n < 0) ? -n : n;

}

1. Реализация стека в виде шаблона

template <class Type>

class Stack

{

private: Type st[MAX]; // стек: массив любого типа

int top; // индекс вершины стека

public: Stack() // конструктор

{

top = -1;

}

void push(Type var) // занести число в стек

{

st[++top] = var;

}

Type pop() // вынуть число из стека

{

return st[top--];

}

}

# 1.3. Универсальная шаблонная коллекция

Мы уже поняли, что шаблон функции — это на самом деле не функция, так как она не приводит к созданию программного кода в памяти. Это просто модель, трафарет для создания множества функций из одной. Это вполне гармонирует с философией ООП. Как и шаблон, класс не представляет собой ничего конкретного, это только база для создания множества похожих объектов.

Шаблоны классов отличаются от шаблонов функций способом реализации. Для создания шаблонной функции мы вызываем ее с аргументами нужного типа. Классы же реализуются с помощью определения объекта, использующего шаблонный аргумент.

Виды коллекций:

1. Коллекция ключ-значение (с уникальными или неуникальными ключами).
2. Стек на основе списка или расширяющегося массива.
3. Очередь с или без приоритета.

# 1.4. Цель курсового проекта

Целью курсовой работы является формирование у студентов правильных представлений о реализации универсальной шаблонной коллекции, разрабатываемой с использованием методологий объектно-ориентированного и обобщенного программирования.

# 1.5. Задачи курсового проекта

Для достижения поставленной цели студенту необходимо решить

следующие задачи:

1) изучить понятие шаблона в языке С++

2) реализовать шаблонный класс, который будет выступать универсальной коллекцией для хранения данных;

3) проверить правильность реализации универсальной коллекции: создать несколько объектов коллекции для хранения простых типов данных, а затем несколько объектов коллекции для хранения других объектов;

4) написать программу, использующую универсальную коллекцию, в соответствии с вариантом.

# 1.6. Требования к составу и оформлению пояснительной записки

1. Пояснительная записка (ПЗ) оформляется на стандартных листах белой бумаги формата А4.
2. Нумерация страниц – сквозная.
   1. Первой страницей является титульный лист (ПРИЛОЖЕНИЕ А), но на нем номер не ставится, на всех остальных листах, включая рисунки и таблицы, номер проставляется арабскими цифрами в правом нижнем углу.
   2. 2-й лист – лист задания на курсовую работу, заполненный и подписанный студентом (ПРИЛОЖЕНИЕ Б).
   3. 3-й лист – аннотация. Аннотация включает: характеристику основной темы, проблемы объекта, цели (и задачи) работы, результаты работы.
   4. 4-й лист – оглавление (в оглавлении включают: введение, название пунктов и подпунктов в соответствии с текстом ПЗ, заключение, библиографический список, приложения).
   5. 5-й лист – введение.
   6. Далее располагается основной материал по специальной части (ОПИСАНИЕ РЕАЛИЗАЦИИ ШАБЛОНА, СХЕМА ФРАГМЕНТА АЛГОРИТМА ПРОГРАММЫ, ЛИСТИНГ ПРОГРАММЫ, ТЕСТОВЫЕ ПРИМЕРЫ), заключение, библиографический список и Приложения. Листы приложений также подлежат сквозной нумерации.
3. При оформлении текста пояснительной записки используется русифицированный шрифт Times New Roman размером14 пт с одинарным (по решению кафедры с полуторным) межстрочным интервалом.

Красная строка абзаца набирается с отступом 0,7 см (по решению кафедры— 1,5 см).

1. Все рисунки должны иметь подрисуночные надписи.
2. Заголовки и подзаголовки ПЗ не подчеркиваются и не выделяются другим цветом. Описки, опечатки и графические неточности, обнаруженные в процессе выполнения документа, допускается исправлять подчисткой или закрашиванием белой краской. На краску наносится на том же месте исправленный текст рукописным (или другим) способом черной пастой.
3. Разделы, подразделы должны иметь заголовки, которые точно и кратко отражают их содержание. Допускается не нумеровать заголовки пунктов и подпунктов. Заголовки разделов печатают прописными буквами, а заголовки подразделов–строчными. Разделам «ВВЕДЕНИЕ, ЗАКЛЮЧЕНИЕ и БИБЛИОГРАФИЧЕСКИЙ СПИСОК» номера не присваиваются.
4. Иллюстрации, таблицы, программы и другие документы для расчетов на ЭВМ или тексты вспомогательного характера допускается давать в виде приложений к ПЗ с указанием наверху посередине страницы слова «ПРИЛОЖЕНИЕ» прописными буквами и его обозначения. Строкой ниже записывается тематический заголовок приложения с прописной буквы. Приложения обозначают заглавными буквами русского алфавита, начиная с А, кроме букв Ё, 3, И, О, Ч, Ь, Ы, Ъ.
5. Все иллюстрации в ПЗ (эскизы, схемы, графики, фотографии) называются рисунками и их нумеруют в пределах раздела. В работе допускаются цветные рисунки. Название рисунка состоит из его номера и наименования. Наименование может включать расшифровку обозначений, использованных в рисунке. Все рисунки нумеруются последовательно арабскими цифрами в пределах одного раздела. Номер рисунка состоит из порядкового номера раздела и порядкового номера рисунка в разделе, разделенных точкой. При небольшом числе рисунков допускается сквозная нумерация рисунков в пределах всей ПЗ. В номер рисунка включается также слово «Рисунок», отделенное знаком «пробел» и тире от цифрового обозначения.

# 2. Основные этапы реализации универсальной шаблонной коллекции

1. Создание шаблонного класса.

Для создания универсальной шаблонной коллекции вам необходимо использовать механизм template<>.

Пример реализации шаблонного массива:

template <typename DataT>

class DynamicArray

{

private:

DataT\* \_data;

int \_size;

int \_index;

void \_stretch()

{

\_size = \_size \* 3 / 2;

DataT\* temp = new DataT[\_size];

for (int i = 0; i < \_index; i++)

temp[i] = \_data[i];

delete[] \_data;

\_data = temp;

}

public:

DynamicArray()

{

\_data = nullptr;

\_size = 2;

\_index = -1;

}

~DynamicArray()

{

delete[] \_data;

}

DataT& operator[] (int i)

{

if (i >= 0 && i <= \_index)

return \_data[i];

return \_data[i];

}

//другие методы

};

На защите курсовой будет необходимо пояснить разницу между

template<class SomeT> и template<type SomeT>.

1. Реализовать методы получения и добавления данных в коллекцию.

Реализовать методы void push(int i) и SomeT& pop (int i), которые будут по заданному алгоритму добавлять и возвращать элементы коллекции.

Пример:

DataT& pop(int i)

{

if (i >= 0 && i <= \_index)

return \_data[i];

return \_data[i];

}

void push(DataT& data)

{

if (!\_data) \_data = new DataT[2];

if (++\_index == \_size) \_stretch();

\_data[\_index] = data;

}

Бизнес-логика класса (т.е. его поведение, совокупность методов) никак не должна взаимодейитвовать с интерфейсом приложения. Класс не должен зависеть от типа приложения. Он должен работать как в консольном приложении, так и в приложении с графическим интерфейсом.

Пример:

void push(DataT& data)

{

if (!\_data) \_data = new DataT[2];

if (++\_index == \_size) \_stretch();

\_data[\_index] = data;

}

DataT& operator[] (int i)

{

if (i >= 0 && i <= \_index)

return \_data[i];

return \_data[i];

}

1. Методы получения и сохранения данных.

Правильным подходом будет перегрузка операторов побитового сдвига (>> / <<) в таком случае объекты класса смогут работать, как и с файловыми, так и с консольными потоками ввода-вывода.

Так же можно реализовать методы: void readFromFile(char\* path) / void writeToFile(char\* path).

Пример перегрузки операторов:

friend istream& operator >> (istream& in, Str& str)

{

char\* temp = new char[200];

in.getline(temp, 200);

str = temp;

delete[] temp;

return in;

}

friend ostream& operator << (ostream& out, Str& str)

{

out << str.\_str;

return out;

}

1. Проверка.

Нужно помнить, что класс коллекции не знает c каким типом данных он работает. Ваш класс не должен использовать никакие методы объектов которые он хранит, кроме стандартных операторов: =, ==, != и т.д.. Но в таком случае, ваш класс, объекты которого вы будете хранить в коллекции, должен иметь перегрузки используемых операторов или же специальные дружественные функции.

Перегрузка оператора: SomeT& operator[](int i).

Дружественная функция: friend bool operator== (SomeT a, SomeT b).

1. Чтобы убедиться, что ваша шаблонная коллекция верно реализована, реализовать проверку на типе int, структуре POINT (Windows.h), и для любого класса.

# 3. Задания на курсовой проект

**Вариант 1**

Написать программу, отыскивающую проход по лабиринту, с использованием класса «стек» на основе двусвязного списка. Лабиринт представляется в виде матрицы, состоящей из квадратов. Каждый квадрат либо открыт, либо закрыт. Вход в закрытый квадрат запрещен. Если квадрат открыт, то вход в него возможен со стороны, но не с угла. Программа находит проход через лабиринт, двигаясь от заданного входа. После отыскания прохода программа выводит найденный путь в виде координат квадратов. Карта считывается из файла вместе с координатой входа в лабиринт. Программа должна в пошаговом режиме продемонстрировать поиск выхода.

**Вариант 2**

Написать программу, моделирующую управление каталогом в файловой системе. Для каждого файла в каталоге содержатся следующие сведения: имя файла, дата создания, количество обращений к файлу. Программа должна обеспечивать:

- создание каталога или файла в выбранном каталоге;

- вывод содержимого каталога (аналог команды dir в cmd);

- удаление файлов, дата создания которых раньше заданной;

- просмотр и редактирование файла;

- выборку файла с наибольшим количеством просмотров;

Выбор моделируемой функции должен осуществляться с помощью меню. Для представления каталога использовать класс «коллекция ключ-значение». Хранить всю информацию (всю систему каталогов и файлов) в базе данных в виде текстового файла.

**Вариант 3**

Написать программу учета книг в библиотеке. Сведения о книгах содержат: фамилию и инициалы автора, название, год издания, количество экземпляров данной книги в библиотеке. Программа должна обеспечивать выбор с помощью меню и выполнение одной из следующих функций: добавление данных о книгах, вновь поступающих в библиотеку; удаление данных о списываемых книгах; выдача сведений о всех книгах, упорядоченных по фамилиям авторов; выдача сведений о всех книгах, упорядоченных по годам издания. Хранение данных организовать с применением класса «коллекция ключ-значение» (с возможностью хранить данные с одинаковыми ключами), в качестве ключа использовать «фамилию и инициалы автора». Предусмотреть сохранение всех данных при выходе в файл и восстановление при повторном запуске программы.

**Вариант 4**

Написать программу «Моя записная книжка». Предусмотреть возможность работы с произвольным числом записей, поиска записи по какому-либо признаку (например, по фамилии, дате рождения или номеру телефона), добавления и удаления записей, сортировки по разным полям. Хранение данных организовать с применением класса «коллекция ключ-значение», с ее же помощью организовать хранение признаков (ключ) и ссылок на сообщения (значение). Предусмотреть сохранение всех данных при выходе в файл и восстановление при повторном запуске программы.

**Вариант 5**

Написать программу учета заявок на обмен квартир и поиска вариантов обмена. Каждая заявка содержит сведения о двух квартирах: требуемой (искомой) и имеющейся. Сведения о каждой квартире содержат: количество комнат, площадь, этаж, район. Программа должна обеспечивать выбор с помощью меню и выполнение одной из следующих функций: ввод заявки на обмен; поиск в картотеке подходящего ### Варианта: при совпадении требований и предложений по количеству комнат и этажности и различии по показателю «площадь» в пределах 10% выводится соответствующая карточка и удаляется из списка, в противном случае поступившая заявка включается в картотеку; вывод всей картотеки. Для хранения данных картотеки использовать класс «коллекция ключ-значение» с неуникальными ключами. Предусмотреть сохранение всех данных при выходе в файл и восстановление при повторном запуске программы.

**Вариант 6**

Написать программу «Автоматизированная информационная система на железнодорожном вокзале». Информационная система содержит сведения об отправлении поездов дальнего следования. Для каждого поезда указывается: номер поезда, станция назначения, время отправления.

Программа должна обеспечивать выбор с помощью меню и выполнение одной из следующих функций:

- ввод данных в информационную систему;

- вывод сведений по всем поездам;

- вывод сведений по поезду с запрошенным номером;

- вывод сведений по тем поездам, которые следуют до запрошенной станции назначения.

Хранение данных организовать с применением класса «коллекция ключ-значение» с уникальными ключами. Предусмотреть сохранение всех данных при выходе в файл и восстановление при повторном запуске программы.

**Вариант 7**

Написать программу «Англо-русский и русско-английский словарь». «База данных» словаря должна содержать синонимичные варианты перевода слов.

Программа должна обеспечивать выбор с помощью меню и выполнение одной из следующих функций:

- загрузка «базы данных» словаря (из файла);

- выбор режима работы (англо-русский / русско-английский);

- вывод вариантов перевода заданного английского слова;

- вывод вариантов перевода заданного русского слова;

- базу данных словаря реализовать в виде двух контейнеров типа «коллекция ключ-значение».

**Вариант 8**

Написать программу, реализующую игру «Крестики-нолики» между двумя игроками: пользователем и компьютером (роботом). В программе использовать стек.

**Вариант 9**

Написать программу, решающую игру-головоломку «Игра в 15». Начальное размещение номеров — случайное. Предусмотреть два режима демонстрации решения: непрерывный (с некоторой задержкой визуализации) и пошаговый (по нажатию любой клавиши). В программе использовать стек.

**Вариант 10**

Составить программу формирования списка кандидатов, участвующих в выборах губернатора. Каждая заявка от кандидата содержит: фамилию и инициалы, дату рождения, место рождения, индекс популярности. Программа должна обеспечивать выбор с помощью меню и выполнение одной из следующих функций:

- добавление заявки в список кандидатов;

- для ввода индекса популярности (значение указано в скобках) предусмотреть выбор с помощью подменю одного из следующих вариантов:

1) поддержан президентом (70)

2) поддержан оппозиционной партией (15)

3) оппозиционный кандидат, который снимет свою кандидатуру в пользу кандидата № 1 (10)

4) прочие (5)

- удаление заявки по заявлению кандидата;

- формирование и вывод списка для голосования;

- хранение данных организовать с применением класса «очередь с приоритетом».

Для надлежащего функционирования очереди с приоритетами побеспокоиться о надлежащем определении операции < (меньше) в классе, описывающем заявку кандидата. Формирование и вывод списка для голосования реализовать посредством выборки заявок из очереди.

**Вариант 11**

Написать программу моделирования работы автобусного парка. Сведения о каждом автобусе содержат: номер автобуса, фамилию и инициалы водителя, номер маршрута. Программа должна обеспечивать выбор с помощью меню и выполнение одной из следующих функций:

- формирование данных об автобусах в парке;

- имитация выезда автобуса из парка: вводится номер автобуса (программа удаляет данные об этом автобусе из списка автобусов, находящихся в парке, и записывает эти данные в список автобусов, находящихся на маршруте);

- имитация въезда автобуса в парк: вводится номер автобуса (программа удаляет данные об этом автобусе из списка автобусов, находящихся на маршруте, и записывает эти данные в список автобусов, находящихся в парке);

- вывод сведений об автобусах, находящихся в парке, и об автобусах, находящихся на маршруте.

Для представления необходимых списков использовать класс «коллекция ключ-значение», в качестве ключа использовать «номер автобуса». Предусмотреть сохранение всех данных при выходе в файл и восстановление при повторном запуске программы.

**Вариант 12**

Написать программу учета заявок на авиабилеты. Каждая заявка содержит: пункт назначения, номер рейса, фамилию и инициалы пассажира, желаемую дату вылета.

Программа должна обеспечивать выбор с помощью меню и выполнение одной из следующих функций:

- добавление заявок в список;

- удаление заявок;

- вывод заявок по заданному номеру рейса и дате вылета;

- вывод всех заявок.

Для хранения данных использовать класс «коллекция ключ-значение» (с неуникальными ключами), в качестве ключа использовать «пункт назначения». Предусмотреть сохранение всех данных при выходе в файл и восстановление при повторном запуске программы.

**Вариант 13**

Написать программу, отыскивающую проход по лабиринту, с использованием класса «стек» на основе расширяющегося массива. Лабиринт представляется в виде матрицы, состоящей из квадратов. Каждый квадрат либо открыт, либо закрыт. Вход в закрытый квадрат запрещен. Если квадрат открыт, то вход в него возможен со стороны, но не с угла. Программа находит проход через лабиринт, двигаясь от заданного входа. После отыскания прохода программа выводит найденный путь в виде координат квадратов. Карта считывается из файла вместе с координатой входа в лабиринт. Программа должна в пошаговом режиме продемонстрировать поиск выхода.

**Вариант 14**

Написать программу «Автоматизированная информационная система на железнодорожном вокзале». Информационная система содержит сведения об отправлении поездов дальнего следования. Для каждого поезда указывается: номер поезда, станция назначения, время отправления.

Программа должна обеспечивать выбор с помощью меню и выполнение одной из следующих функций:

- ввод данных в информационную систему;

- вывод сведений по всем поездам;

- вывод сведений по поезду с запрошенным номером;

- вывод сведений по тем поездам, которые следуют до запрошенной станции назначения.

Хранение данных организовать с применением класса «коллекция ключ-значение» (с неуникальными ключами), в качестве ключа использовать номер поезда. Предусмотреть сохранение всех данных при выходе в файл и восстановление при повторном запуске программы.

# Библиографический список

1) Роберт Лафоре Объектно-ориентированное программирование в С++/ изд.4-е, Спб.:Питер, 2003.- 923 с.

2) Глушаков С.В. Программирование на С++/ изд.2-е, доп. и переработ.- М.:АСТ, 2008. - 685 с.

3) Подбельский, В.В. Программирование на языке Си: Учеб. пособие /

С.С. Фомин, В.В. Подбельский. — 2-е доп. изд. — М: Финансы и статистика,

2002. — 600 с.

4) Шилдт, Герберт. С++:базовый курс, 3-е издание. : Пер. с англ. – М. : Издательский дом «Вильямс». 2010. – 624 с. : ил. – Парал. тит. англ.

# **ПРИЛОЖЕНИЯ**

# **Приложение А**

**ТИТУЛЬНЫЙ ЛИСТ (КУРСОВАЯ РАБОТА)**

Министерство образования и науки Российской Федерации

Государственное образовательное учреждение высшего профессионального образования Южно-Уральский государственный университет Приборостроительный факультет

Кафедра электронные вычислительные машины

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по дисциплине «Объектно-ориентированное программирование»

Тема: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(обозначение курсовой работы: аббревиатура университета, номер специальности (шесть цифр), год работы (четыре цифры), предпоследние три цифры номера студенческого удостоверения, ПЗ КР)

Нормоконтролер

\_\_\_\_\_\_\_\_\_ / Кирсанова А.А.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_201\_ г.

Руководитель

\_\_\_\_\_\_\_\_\_ / Кирсанова А.А.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_201\_ г.

Автор работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

студент группы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 201\_ г.

Работа защищена с оценкой

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 201\_ г.

Челябинск 201\_

# **Приложение Б**

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ**

«Реализация универсальной шаблонной коллекции» (2 семестр)

Южно-Уральский государственный университет

Кафедра электронные вычислительные машины

20\_\_/\_\_ учебный год

Т Е Х Н И Ч Е С К О Е З А Д А Н И Е

на курсовую работу " Реализация универсальной шаблонной коллекции "

Студента группы КЭ-\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дисциплина: Объектно-ориентированное программирование

Тема работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи работы: «\_\_»\_\_\_\_\_\_\_\_\_201\_ г.

Срок сдачи работы: «\_\_» \_\_\_\_\_\_\_\_\_ 201\_ г.

Задание:

Вариант № \_\_

Текст задания

Оглавление

[Предисловие 1](#_Toc504573237)

[1. Постановка цели и задач курсовой работы 1](#_Toc504573238)

[1.1. Введение в обобщенное программирование 1](#_Toc504573239)

[1.2. Понятие шаблона 2](#_Toc504573240)

[1.3. Универсальная шаблонная коллекция 3](#_Toc504573241)

[1.4. Цель курсового проекта 4](#_Toc504573242)

[1.5. Задачи курсового проекта 4](#_Toc504573243)

[1.6. Требования к составу и оформлению пояснительной записки 4](#_Toc504573244)

[2. Основные этапы реализации универсальной шаблонной коллекции 6](#_Toc504573245)

[3. Задания на курсовой проект 9](#_Toc504573246)

[Библиографический список 14](#_Toc504573247)

[ПРИЛОЖЕНИЯ 15](#_Toc504573248)

[Приложение А 15](#_Toc504573249)

[Приложение Б 16](#_Toc504573250)