Лабораторная работа №8
«Разработка шаблона класса»

1. Цель работы

Целью данной работы является изучение шаблонов классов языка C++.

2. Исходные данные

2.1 Некоторые контейнерные классы

В некоторых вариантах заданий размер данных, хранящихся в объекте класса, не фиксирован, и, тем самым, требуется динамическое выделение памяти. Чтобы избежать реализации конструктора копий, деструктора и перегруженной операции присваивания, целесообразно в своём классе организовать хранение данных в объектах контейнерных классов из стандартной библиотеки C++.

Динамические массивы в C++ представляются переменными класса vector, объявленного в заголовочном файле vector. Использование этого класса можно проиллюстрировать следующим примером (создание вектора целых чисел и заполнение его числами от 0 до 9):

```
vector a;
for (int i = 0; i < 10; i++) a.push_back (i);
```

К вектору можно применять операцию индексации. Размер вектора возвращает метод size. Поэтому, например, вывести элементы нашего вектора на печать можно следующим образом:

```
for(int i = 0 ; i < a.size( ); i++) cout << a [i];
```

Вектора, кроме того, предоставляют итераторы для доступа к своему содержимому. Это даёт возможность для перебора элементов вектора использовать специальную форму оператора for:

```
for(int x : a) cout << x;
```

Упорядоченный ассоциативный массив, реализованный через дерево, представлен в стандартной библиотеке C++ шаблоном класса map из заголовочного файла map.

Следующий фрагмент кода демонстрирует объявление ассоциативного массива, заполнение его значениями и вывод словарных пар на печать:

```
map<string, int>  m;
m["a"] = 1;
m["b"] = 2;
for (auto pair : m) cou t << pair.f i r s t << ", ␣" << pair.second << endl;
```

Обратите внимание на то, что применение ключевого слова auto в объявлении переменной pair заставляет компилятор выводить её тип автоматически. Эта возможность доступна только в режиме c++11.

Упорядоченное множество, также реализованное через дерево, представлено шаблоном класса set из заголовочного файла set. Объявление множества, добавление в него

элементов и вывод множества на печать можно проиллюстрировать следующим фрагментом кода:

```
set<string> s ;
s.insert("qwerty");
s.insert("abcd");
for (string x : s) cout << x << endl;
```

Ассоциативный массив и множество, реализованные через хеш-таблицу, представлены в стандартной библиотеке языка C++ шаблонами классов unordered_map и unordered_set, соответственно.

2.2 Специализация шаблонов в зависимости от значений их целочисленных параметров

Информация об объявлении и специализации шаблонов дана в лекционном курсе. Здесь мы рассмотрим только один вопрос, касающийся специализации шаблонов в зависимости от значений их целочисленных параметров.

Пусть шаблон некоторого класса Abs имеет целочисленный параметр N:

```
template <int N>
class Abs
{
	public :
		void print();
};

template <int N>
void Abs<N>:: print()
{
	cout << N << endl;
}
```

Допустим, мы хотим добиться, чтобы в этом совершенно искусственном примере метод print всегда печатал абсолютное значение числа N. При этом мы собираемся достигнуть этого результата через специализацию шаблона класса.

Добавим в шаблон дополнительный булевский параметр isNegative со значением по умолчанию, вычисляемым на основе значения параметра N:

```
template <int N, bool isNegative = (N > 0)>
class Abs
{
	public:
		void print();
};

template <int N, bool isNegative>
void Abs<N,isNegative>::print()
{
	cout << N << endl;
}
```

Очевидно, что при отрицательных значениях параметра N значение параметра isNegative будет истинным. Поэтому мы можем добавить специализированную версию шаблона для отрицательных N:

```
template <int N>
class Abs<N, true>
{
	public :
		void print();
};

template <int N>
void Abs<N, true> :: print()
{
	cout << −N << endl;
}
```

Теперь при инстанциации шаблона класса Abs с отрицательным N компилятор будет задействовать специализированную версию, в которой N печатается с изменением знака. В этом можно убедиться, добавив в нашу программу следующий код:

```
int main ()
{
	Abs<−5> t;
	t.print();
	return 0;
}
```

3 Задание

Согласно выбранному из таблиц 1–16 описанию требуется составить шаблон класса, разместив его в отдельном заголовочном файле. Проверку работоспособности класса требуется организовать в функции main, размещённой в файле «main.cpp».

4. Дополнительная информация

1. Возможности C++ 11: https://habr.com/ru/post/182920/
2. Шаблоны классов: https://habr.com/ru/post/599801/

	Формулировка задачи	Студент	Группа
1	FenwickTree<T,N> – дерево Фенвика для последовательности длины N с элементами типа T, имеющее операцию вычисления исключающего ИЛИ элементов на заданном отрезке и операцию изменения указанного элемента. Дерево должно быть реализовано через массив размера N. В FenwickTree<bool, N> каждому элементу должен соответствовать 1 бит.		
2	SparseTable<T, N> – разреженная таблица для последовательности длины N с элементами типа T, имеющая операцию вычисления максимума на заданном отрезке. Таблица должна быть представлена в виде двумерного массива с константными размерами, вычисляемыми компилятором во время компиляции.		
3	Diraph<int N, bool Dense> – простой ориентированный граф, у которого N вершин и про который известно, будет ли он плотным или разреженным. Плотный граф должен быть реализован через матрицу смежности, а разреженный – через списки инцидентности. В графе должны быть реализованы операции: добавление дуги (принимает номера вершин, которые надо соединить) и проверка смежности двух вершин.		
4	RangeSet<T, Closed> – набор интервалов, имеющих вид (a, b) при Closed==false или [a, b] при Closed==true. Набор интервалов задаёт множество чисел типа T. Операции: проверка принадлежности числа множеству; добавление нового интервала. Если тип T – целочисленный, в классе должна быть дополнительная операция, вычисляющая количество чисел, принадлежащих задаваемому набором интервалов множеству.		
5	Points<T> – множество точек на плоскости, в которой каждая точка помечена значением типа T. Операции: добавление новой точки; удаление точек, попадающих внутрь указанного круга. Если T – числовой тип, то метка точки должна трактоваться как её масса, и в классе должна быть дополнительная операция, вычисляющая центр масс множества точек.		
6	Path<T,M> – траектория движения точки в M-мерном пространстве, заданная с помощью n положений точки в моменты времени 0, 1, . . . , n − 1. Координаты точки задаются числами типа		

T.
Операции: вычисление длины траектории; добавление нового положения точки в конец траектории.
В случае, если M = 1, в классе должна быть дополнительная операция, вычисляющая минимальную и максимальную координату точки на траектории.

7 Matrix<M,N> – целочисленная матрица размера M × N с операцией, возвращающей ссылку на указанный элемент. Если M = N, то для матрицы должна быть доступна операция возведения в квадрат.

8 Equality<L,N> – линейное равенство вида $a_1x_1 + a_2x_2 + \ldots + a_nx_n = b$, коэффициенты a_i и b которого заданы целыми числами, лежащими в диапазоне от L до H. В классе должна быть предусмотрена операция проверки, удовлетворяет ли указанный вектор значений переменных равенству.
Коэффициенты равенства должны храниться в виде $a_i - L$, причём для их представления должен использоваться целочисленный тип минимального размера, подходящий для представления числа H − L + 1.

9 PascalArray<int L,int R,class T> – массив с элементами типа T, индексируемыми от L до R, с перегруженной операцией индексации и операцией конкатенации двух массивов, которая допустима в случае, если правая граница первого массива на единицу меньше левой границы второго массива. Массив, в котором R = L − 1, считается пустым и не обладает операцией индексации.

10 Queue<class T,int N> – очередь с элементами типа T и максимальным размером N, который может быть не задан (равен 0). Для очереди должны быть реализованы операции enqueue, dequeue и is_empty. Очередь с ненулевым максимальным размером должна быть реализована через массив размера N, хранящийся в поле объекта очереди.

11 Polygon<T,N> – N-угольник на плоскости, заданный координатами вершин. Координаты вершин представлены числами типа T.
Операции: вычисление периметра (возвращает double); добавление новой вершины (в результате формируется новый (N + 1)-угольник). Если N = 3, в классе должна быть дополнительная операция вычисления площади

треугольника (возвращает double).

12 Seq<class T,bool Unique> – последовательность с элементами типа T, которая может допускать или не допускать наличие повторяющихся элементов. Для последовательности должна быть перегружена операция индексации (она должна возвращать константную ссылку в случае последовательности с уникальными элементами, чтобы элемент по ссылке нельзя было изменить). Кроме того, должна быть реализована операция добавления нового элемента в последовательность.

13 IntStack<L,N> – стек целых чисел из диапазона от L до H, имеющий стандартный для стека набор операций. Если размер диапазона не превышает 256, для представления стека должен использоваться массив char'ов. В случае диапазона, имеющего размер, не превышающий 65536, должен использоваться массив short'ов.

14 Polynom<int L, int H, int N> – полином степени N с элементами из диапазона от L до H, имеющий следующие операции: дифференцирование полинома, в результате которого должен формироваться новый полином типа Polynom<L2, H2, N2>, где L2, H2 и N2 вычисляются на базе значений L, H и N; вычисление значения полинома в точке. Если размер диапазона не превышает 256, для представления полинома должен использоваться массив char'ов.

15 Polygon<T> – многоугольник на плоскости, заданный координатами вершин. Координаты вершин представлены числами типа T. Операции: вычисление периметра (возвращает double); добавление новой вершины. В случае, если T – double, в классе должна быть дополнительная операция поворота N-угольника относительно его первой вершины на такой угол, чтобы его первое ребро стало параллельно оси OX.

16 Matrix<T,N> – квадратная матрица размера N, элементы которой имеют тип T. Матрица должна иметь следующие операции:

1. запись значения в элемент с индексами (i, j);
2. чтение значения из элемента с индексами (i, j);
3. построение новой матрицы путём удаления i-той строки и j-того столбца.

В Matrix<bool,N> при N ≤ 8 матрица должна быть представлена 64-разрядным целым числом, в котором каждому элементу соответствует один бит.

17 Matrix<L, H, M, N> – целочисленная матрица размера M × N, элементы которой принадлежат диапазону от L до H. Матрица должна иметь следующие операции:

1. запись значения в элемент с индексами (i, j);
2. чтение значения из элемента с индексами (i, j);
3. транспонирование.

Если размер диапазона не превышает 256, для представления матрицы должен использоваться массив char'ов.

18 Stack<T> – стек с элементами типа T, имеющий в дополнение к обычным стековым операциям операцию переворачивания стека. Все операции должны работать за константное время. Для представления стека нужно использовать двунаправленный список. В Stack<int> дополнительно должна быть реализована операция, сообщающая о наличии в стеке нулевого элемента и работающая за константное время.

19 SegmentTree<T,N> – дерево отрезков для последовательности длины N с элементами типа T, имеющее операцию вычисления суммы элементов на заданном отрезке и операцию изменения указанного элемента. Отметим, что в случае, если элементами последовательности являются строки, от дерева отрезков нет никакой пользы, так как всё равно сумма строк вычисляется за время, пропорциональное длине результирующей строки. Соответственно, SegmentTree<string, N> должно быть реализовано через обычный массив строк размера N.

20 Seq<class T,bool Sorted> – последовательность с элементами типа T, которая может быть отсортирована или не отсортирована. Для последовательности должна быть перегружена операция индексации (она должна возвращать константную ссылку в случае отсортированной последовательности, чтобы элемент по ссылке нельзя было изменить). Кроме того, должны быть операции добавления и удаления элемента последовательности.

21 Stack<class T,int N> – стек с элементами типа T и максимальным размером N, который может быть не задан (равен 0). Для стека должны быть реализованы обычные стековые операции. Стек с ненулевым максимальным размером должен быть реализован через массив размера N, хранящийся в поле объекта стека.

22 Queue<T,N> – «неизменяемая» очередь с элементами типа T и максимальным размером N, имеющая обычные для очереди операции. «Неизменяемость» очереди заключается в том, что операции Enqueue и Dequeue вместо изменения очереди, для которой они вызваны, создают и возвращают новую очередь, отличающуюся от исходной на один элемент. При этом исходная очередь полностью сохраняет своё состояние и работоспособность. Операции Enqueue и Dequeue нужно реализовать так, чтобы они работали за амортзированное константное время. Для этого нужно использовать два стека размера N. Операция Enqueue записывает новый элемент в первый стек, а операция Dequeue забирает элемент из второго стека. Фокус заключается в том, что стеки выделяются в динамической памяти и являются общими для очереди, на которой сработала операция, и для очереди, которая была создана в результате работы операции. Отметим, что новая пара стеков создаётся в момент создания новой пустой очереди, а также тогда, когда операция Dequeue обнаруживает, что второй стек пуст и нужно копировать содержимое первого стека во второй. Для освобождения памяти, занимаемой парой стеков, которая потенциально может разделяться сразу несколькими очередями, требуется воспользоваться подсчётом ссылок на эту пару стеков (можно воспользоваться шаблоном shared_ptr).

23 List<T> – последовательность значений типа T, реализованная через двунаправленный список с операциями поиска значения, вставки значения в указанное место, удаления значения. В List<int> дополнительно должна присутствовать операция вычисления суммы значений.

24 Matrix<T,M,N> – матрица размера M × N, элементы которой имеют тип T. Матрица должна иметь следующие операции:

1. запись значения в элемент с индексами (i, j);
2. чтение значения из элемента с индексами (i,

j);
3. транспонирование.

В Matrix<bool,M,N> при M ≤ 8, N ≤ 8 матрица должна быть представлена 64-разрядным целым числом, в котором каждому элементу соответствует один бит

25 Seq<T,N> – неизменяемая упорядоченная последовательность длины N элементов типа T, имеющая операцию поиска первого вхождения в неё подпоследовательности длины M. В операции поиска должен использоваться алгоритм Кнута–Морриса–Пратта. Операция поиска должна создаваться шаблоном, имеющим параметр M. Этот шаблон должен быть специализирован для случая M = N, при котором алгоритм Кнута–Морриса–Пратта не нужен, а достаточно просто сравнить две последовательности.

26 Stack<T> – стек с элементами типа T, имеющий в дополнение к обычным стековым операциям операцию вычисления максимального элемента, работающую за константное время. Stack<string> должен дополнительно иметь операцию переворачивания всех строк, находящихся в стеке, работающую за константное время.

27 Polynom<T,N> – полином порядка N с коэффициентами типа T, имеющий операцию вычисления значения. В Polynom<bool, N> i-ый коэффициент означает наличие или отсутствие в полиноме члена x^i .

28 Operation <char O,class L,class R>– некоторое действие над двумя операндами. Операнды имеют типы L и R, каждый из которых – либо Operation, либо int (обозначает константу), либо int& (обозначает переменную). Константа O задаёт смысл действия: ’+’ – сложение, ’∗ ’ – умножение, ’=’ – присваивание. В классе надо реализовать операцию выполнения действия.

29 Stack<T,N> – стек с элементами типа T и максимальным размером N, имеющий обычные стековые операции. Стек должен быть реализован через массив размера N. В Stack<bool,N> каждый элемент должен быть представлен одним битом.

30 Queue<T, N> – очередь с элементами типа T и максимальным размером N, реализованная через двойной стек и имеющая в дополнение к обычным для очереди операциям операцию

переворачивания очереди, работающую за константное время.
В Queue<int, N> дополнительно должна быть реализована операция вычисления суммы элементов очереди, работающая за константное время.