Министерство образования и науки Российской Федерации

Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский**

**политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе №18.7**

Дисциплина: «Информатика»

Тема: Объектно-ориентированное программирование. Шаблоны классов

Вариант 8

Выполнил:

Студент группы РИС-20-1б

Тараканов Д. М.

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

Пермь

2021 год

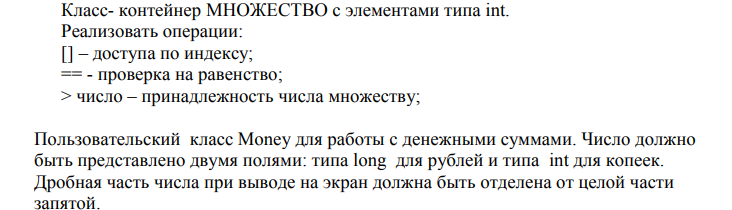
# **Цель работы**

1. Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.
2. Реализация шаблона класса-контейнера.

# **Постановка задачи**

1. Определитель шаблон класса-контейнера.
2. Реализовать конструкторы, деструкторы, операции ввода-вывода, операцию присваивания.
3. Перегрузить операции, указанные в варианте.
4. Инстанцировать шаблон для стандартных типов.
5. Реализовать пользовательский класс.
6. Перегрузить для пользовательского класса операции ввода и вывода.
7. Перегрузить операции необходимые для выполнения операций контейнерного класса.
8. Инстанцировать шаблон для пользовательского класса.
9. Написать демонстрационную программу, иллюстрирующую выполнение задачи.

(8 вариант)



# **Анализ задачи**

1. Определить какие действия предстоит выполнить:

* Разработать класс Money который хранит в себе значения рублей и копеек.
* Организовать методы для ввода данных, селекторы и модификатор, конструкторы и деструкторы, перегрузки операторов.

class Money {

private:

long int rubles;

int penny;

public:

Money();

Money(long int, int);

Money(const Money&);

virtual ~Money() {}

bool operator == (Money&);

bool operator != (Money&);

bool operator > (const Money&);

bool operator < (const Money&);

Money& operator = (const Money&);

Money& operator + (int);

friend istream& operator >> (istream&, Money&);

friend ostream& operator << (ostream&, const Money&);

};

* Разработать шаблон класс Set с полями data, size, max\_size, begin, end.
* Организовать методы для ввода данных, конструкторы и деструкторы, перегрузки операторов для класса Set.

template <class T>

class Set

{

private:

int size = 0;

int max\_size;

T\* data;

public:

Set();

Set(const Set<T>&);

~Set();

int get\_size() {

return size;

}

void Insert(T);

Set<T>& operator = (const Set<T>&);

T operator [] (int);

bool operator == (const Set <T>&);

bool operator > (T);

friend istream& operator >> <>(istream&, Set<T>&);

friend ostream& operator << <>(ostream&, const Set<T>&);

};

* Протестировать работу шаблон контейнера в функции main.

void main() {

system("chcp 1251");

cout << "INT";

Set<int> A, B;

cout << "\nЗаполнение множества A\n"; cin >> A; cout << "\nМножество A " << A;

B.Insert(12);

B.Insert(4);

B.Insert(10);

B.Insert(0);

cout << "\nМножество B " << B;

system("pause");

cout << "\nСравнение множеств\n";

if (A == B) {

cout << "\nМножества равны!\n";

}

else {

cout << "\nМножества не равны!\n";

}

system("pause");

int i\_element;

cout << "\nПоиск числа во множестве A\nЧисло\n> "; cin >> i\_element;

if (A > i\_element) {

cout << "\nЧисло во множестве есть!\n";

}

else {

cout << "\nЧисла во множестве нет!\n";

}

system("pause");

int number, size=A.get\_size();

cout << "\nДоступ к элементу множества A\n\nНомер элемента";

do {

cout << "\n> ";

cin >> number;

if (number > size) {

cout << "\nВведите индекс меньший длины множества\n";

}

if (number < 1) {

cout << "\nВведите положительный индекс\n";

}

} while (number > size || number < 1);

cout << "\nA[" << number << "] = " << A[number - 1] << endl;

system("pause");

cout << "DOUBLE";

Set<double> C, D;

cout << "\nЗаполнение множества C\n"; cin >> C; cout << "\nМножество C " << C;

D.Insert(1.75);

D.Insert(4.21);

D.Insert(1.405);

D.Insert(0.0);

cout << "\nМножество D " << D;

system("pause");

cout << "\nСравнение множеств\n";

if (C == D) {

cout << "\nМножества равны!\n";

}

else {

cout << "\nМножества не равны!\n";

}

system("pause");

double d\_element;

cout << "\nПоиск числа во множестве C\nЧисло\n> "; cin >> d\_element;

if (C > d\_element) {

cout << "\nЧисло во множестве есть!\n";

}

else {

cout << "\nЧисла во множестве нет!\n";

}

system("pause");

size = C.get\_size();

cout << "\nДоступ к элементу множества C\n\nНомер элемента";

do {

cout << "\n> ";

cin >> number;

if (number > size) {

cout << "\nВведите индекс меньший длины множества\n";

}

if (number < 1) {

cout << "\nВведите положительный индекс\n";

}

} while (number > size || number < 1);

cout << "\nC[" << number << "] = " << C[number - 1] << endl;

system("pause");

cout << "CHAR";

Set<char> E, F;

cout << "\nЗаполнение множества E\n"; cin >> E; cout << "\nМножество E " << E;

F.Insert('f');

F.Insert('1');

F.Insert('/');

F.Insert('0');

cout << "\nМножество F " << F;

system("pause");

cout << "\nСравнение множеств\n";

if (E == F) {

cout << "\nМножества равны!\n";

}

else {

cout << "\nМножества не равны!\n";

}

system("pause");

char c\_element;

cout << "\nПоиск символ во множестве E\nСимвол\n> "; cin >> c\_element;

if (E > c\_element) {

cout << "\nСимвол во множестве есть!\n";

}

else {

cout << "\nСимвола во множестве нет!\n";

}

system("pause");

size = E.get\_size();

cout << "\nДоступ к элементу множества E\n\nНомер элемента";

do {

cout << "\n> ";

cin >> number;

if (number > size) {

cout << "\nВведите индекс меньший длины множества\n";

}

if (number < 1) {

cout << "\nВведите положительный индекс\n";

}

} while (number > size || number < 1);

cout << "\nE[" << number << "] = " << E[number - 1] << endl;

system("pause");

cout << "MONEY\nПростейшие методы с пользовательским типом";

Money g;

cout << "\nСумма \n"; cin >> g; cout << "\nСумма: " << g << endl;

system("pause");

int x;

cout << "\nСколько копеек прибавить\n>"; cin >> x;

g + x;

cout << "\nСумма 1: " << g << endl;

system("pause");

Set<Money> G, H;

cout << "\nЗаполнение множества G\n"; cin >> G; cout << "\nМножество G " << G;

cout << "\nЗаполнение множества H\n"; cin >> H; cout << "\nМножество H " << H;

system("pause");

cout << "\nСравнение множеств\n";

if (G == H) {

cout << "\nМножества равны!\n";

}

else {

cout << "\nМножества не равны!\n";

}

system("pause");

Money m\_element;

cout << "\nПоиск суммы во множестве G\nСумма\n> "; cin >> m\_element;

if (G > m\_element) {

cout << "\nСумма во множестве есть!\n";

}

else {

cout << "\nСуммы во множестве нет!\n";

}

system("pause");

size = G.get\_size();

cout << "\nДоступ к элементу множества G\n\nНомер элемента";

do {

cout << "\n> ";

cin >> number;

if (number > size) {

cout << "\nВведите индекс меньший длины множества\n";

}

if (number < 1) {

cout << "\nВведите положительный индекс\n";

}

} while (number > size || number < 1);

cout << "\nG[" << number << "] = " << G[number - 1] << endl;

system("pause");

}

1. C какими типами данных предстояло работать:

* Для хранения данных реализована переменная data типа T\*.

T\* data;

* Для хранения длины множества реализована переменная size типа int.

int size;

* Для хранения максимальной длины множества реализована переменная max\_size типа int.

int max\_size;

* Для хранения значения итератора реализована переменная element типа int\*.

int\* size;

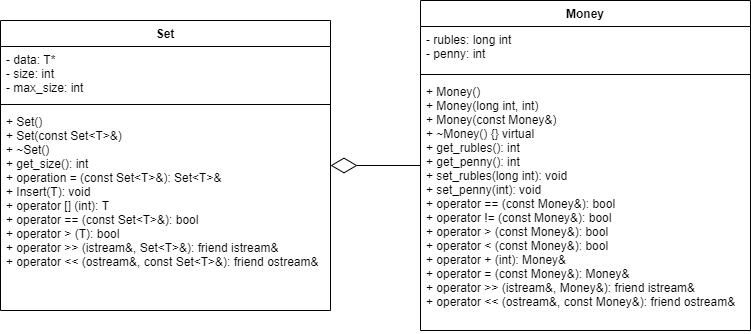
* Для хранения кол-ва рублей реализована переменная rubles типа long int.

long int rubles;

* Для хранения кол-ва копее реализована переменная penny типа int.

int penny;

# **UML диаграмма**



# **Код программы на языке C++**

Заголовочный файл Set.h

#pragma once

#include <iostream>

#include <algorithm>

using namespace std;

const int Lenght = 10;

template <class T>

class Set

{

private:

int size = 0;

int max\_size;

T\* data;

public:

Set();

Set(const Set<T>&);

~Set();

int get\_size() {

return size;

}

void Insert(T);

Set<T>& operator = (const Set<T>&);

T operator [] (int);

bool operator == (const Set <T>&);

bool operator > (T);

friend istream& operator >> <>(istream&, Set<T>&);

friend ostream& operator << <>(ostream&, const Set<T>&);

};

template <class T>

Set<T>::Set() {

size = 0;

max\_size = Lenght;

data = new T[max\_size];

}

template <class T>

Set<T>::Set(const Set& temp\_Set) {

size = temp\_Set.size;

max\_size = temp\_Set.max\_size;

data = new T[max\_size];

for (int i = 0; i < size; i++) {

data[i] = temp\_Set.data[i];

}

}

template <class T>

Set<T>::~Set() {

delete[] data;

data = 0;

}

template <class T>

void Set<T>::Insert(T value) {

int i;

for (i = 0; i < size; i++) {

if (data[i] == value) {

i = size;

}

}

if (i != size + 1) {

data[size] = value;

if (i > 0) {

sort(data, data + size + 1);

}

size++;

if (size == max\_size) {

max\_size += max\_size;

T\* temp\_data = new T [max\_size];

for (i = 0; i < size; i++) {

temp\_data[i] = data[i];

}

delete[] data;

data = temp\_data;

}

}

}

template<class T>

Set<T>& Set<T>::operator = (const Set<T>& temp\_Set) {

if (\*this == temp\_Set) {

return \*this;

}

size = temp\_Set.size;

max\_size = temp\_Set.max\_size;

delete[] data;

data = temp\_Set.data;

return \*this;

}

template<class T>

T Set<T>::operator [] (int index) {

return data[index];

}

template<class T>

bool Set<T>::operator == (const Set<T>& temp\_Set) {

if (size == temp\_Set.size) {

int i;

for (i = 0; i < size; i++) {

if (data[i] != temp\_Set.data[i]) {

i = size;

}

}

if (i == size) {

return true;

}

}

return false;

}

template<class T>

bool Set<T>::operator > (T value) {

for (int i = 0; i < size; i++) {

if (data[i] == value) {

return true;

}

}

return false;

}

template <class T>

istream& operator >> (istream& in, Set<T>& temp\_Set) {

T number;

int quantity;

cout << "\nКол-во элментов - "; in >> quantity;

for (int i = 0; i < quantity; i++) {

cout << i + 1 << ") ";

in >> number;

temp\_Set.Insert(number);

}

return in;

}

template <class T>

ostream& operator << (ostream& out, const Set<T>& temp\_Set) {

out << "{ ";

for (int i = 0; i < temp\_Set.size; i++) {

out << temp\_Set.data[i] << " ";

}

out << "}\n";

return out;

}

Заголовочный файл Money.h

#pragma once

#include <iostream>

using namespace std;

class Money {

private:

long int rubles;

int penny;

public:

Money();

Money(long int, int);

Money(const Money&);

virtual ~Money() {}

bool operator == (Money&);

bool operator != (Money&);

bool operator > (const Money&);

bool operator < (const Money&);

Money& operator = (const Money&);

Money& operator + (int);

friend istream& operator >> (istream&, Money&);

friend ostream& operator << (ostream&, const Money&);

};

Обьявление класса в Money.cpp

#include "Money.h"

Money::Money() {

rubles = 0;

penny = 0;

}

Money::Money(long int r, int p) {

rubles = r;

penny = p;

}

Money::Money(const Money& temp\_Money) {

rubles = temp\_Money.rubles;

penny = temp\_Money.penny;

}

bool Money:: operator == (Money& temp\_Money) {

long int temp1 = rubles \* 100 + penny;

long int temp2 = temp\_Money.rubles \* 100 + temp\_Money.penny;

if (temp1 == temp2) {

return true;

}

return false;

}

bool Money:: operator != (Money& temp\_Money) {

long int temp1 = rubles \* 100 + penny;

long int temp2 = temp\_Money.rubles \* 100 + temp\_Money.penny;

if (temp1 != temp2) {

return true;

}

return false;

}

bool Money::operator>(const Money& temp\_Money) {

return (rubles \* 100 + penny > temp\_Money.rubles \* 100 + temp\_Money.penny);

}

bool Money::operator<(const Money& temp\_Money)

{

return (rubles \* 100 + penny < temp\_Money.rubles \* 100 + temp\_Money.penny);

}

Money& Money::operator = (const Money& temp\_Money) {

rubles = temp\_Money.rubles;

penny = temp\_Money.penny;

return \*this;

}

Money& Money::operator + (int x) {

long int temp = rubles \* 100 + penny;

temp += x;

rubles = temp / 100;

penny = temp % 100;

return \*this;

}

istream& operator >> (istream& in, Money& temp\_Money) {

do {

cout << "\nРубли: "; in >> temp\_Money.rubles;

} while (temp\_Money.rubles < 0);

do {

cout << "Копейки: "; in >> temp\_Money.penny;

} while (temp\_Money.penny > 99 || temp\_Money.penny < 0);

return in;

}

ostream& operator<<(ostream& out, const Money& temp\_Money) {

if (temp\_Money.penny < 10) {

return (out << temp\_Money.rubles << ",0" << temp\_Money.penny);

}

else {

return (out << temp\_Money.rubles << "," << temp\_Money.penny);

}

}

Main файл

#include <iostream>

#include "Set.h"

#include "Money.h"

using namespace std;

void main() {

system("chcp 1251");

cout << "INT";

Set<int> A, B;

cout << "\nЗаполнение множества A\n"; cin >> A; cout << "\nМножество A " << A;

B.Insert(12);

B.Insert(4);

B.Insert(10);

B.Insert(0);

cout << "\nМножество B " << B;

system("pause");

cout << "\nСравнение множеств\n";

if (A == B) {

cout << "\nМножества равны!\n";

}

else {

cout << "\nМножества не равны!\n";

}

system("pause");

int i\_element;

cout << "\nПоиск числа во множестве A\nЧисло\n> "; cin >> i\_element;

if (A > i\_element) {

cout << "\nЧисло во множестве есть!\n";

}

else {

cout << "\nЧисла во множестве нет!\n";

}

system("pause");

int number, size=A.get\_size();

cout << "\nДоступ к элементу множества A\n\nНомер элемента";

do {

cout << "\n> ";

cin >> number;

if (number > size) {

cout << "\nВведите индекс меньший длины множества\n";

}

if (number < 1) {

cout << "\nВведите положительный индекс\n";

}

} while (number > size || number < 1);

cout << "\nA[" << number << "] = " << A[number - 1] << endl;

system("pause");

cout << "DOUBLE";

Set<double> C, D;

cout << "\nЗаполнение множества C\n"; cin >> C; cout << "\nМножество C " << C;

D.Insert(1.75);

D.Insert(4.21);

D.Insert(1.405);

D.Insert(0.0);

cout << "\nМножество D " << D;

system("pause");

cout << "\nСравнение множеств\n";

if (C == D) {

cout << "\nМножества равны!\n";

}

else {

cout << "\nМножества не равны!\n";

}

system("pause");

double d\_element;

cout << "\nПоиск числа во множестве C\nЧисло\n> "; cin >> d\_element;

if (C > d\_element) {

cout << "\nЧисло во множестве есть!\n";

}

else {

cout << "\nЧисла во множестве нет!\n";

}

system("pause");

size = C.get\_size();

cout << "\nДоступ к элементу множества C\n\nНомер элемента";

do {

cout << "\n> ";

cin >> number;

if (number > size) {

cout << "\nВведите индекс меньший длины множества\n";

}

if (number < 1) {

cout << "\nВведите положительный индекс\n";

}

} while (number > size || number < 1);

cout << "\nC[" << number << "] = " << C[number - 1] << endl;

system("pause");

cout << "CHAR";

Set<char> E, F;

cout << "\nЗаполнение множества E\n"; cin >> E; cout << "\nМножество E " << E;

F.Insert('f');

F.Insert('1');

F.Insert('/');

F.Insert('0');

cout << "\nМножество F " << F;

system("pause");

cout << "\nСравнение множеств\n";

if (E == F) {

cout << "\nМножества равны!\n";

}

else {

cout << "\nМножества не равны!\n";

}

system("pause");

char c\_element;

cout << "\nПоиск символ во множестве E\nСимвол\n> "; cin >> c\_element;

if (E > c\_element) {

cout << "\nСимвол во множестве есть!\n";

}

else {

cout << "\nСимвола во множестве нет!\n";

}

system("pause");

size = E.get\_size();

cout << "\nДоступ к элементу множества E\n\nНомер элемента";

do {

cout << "\n> ";

cin >> number;

if (number > size) {

cout << "\nВведите индекс меньший длины множества\n";

}

if (number < 1) {

cout << "\nВведите положительный индекс\n";

}

} while (number > size || number < 1);

cout << "\nE[" << number << "] = " << E[number - 1] << endl;

system("pause");

cout << "MONEY\nПростейшие методы с пользовательским типом";

Money g;

cout << "\nСумма \n"; cin >> g; cout << "\nСумма: " << g << endl;

system("pause");

int x;

cout << "\nСколько копеек прибавить\n>"; cin >> x;

g + x;

cout << "\nСумма 1: " << g << endl;

system("pause");

Set<Money> G, H;

cout << "\nЗаполнение множества G\n"; cin >> G; cout << "\nМножество G " << G;

cout << "\nЗаполнение множества H\n"; cin >> H; cout << "\nМножество H " << H;

system("pause");

cout << "\nСравнение множеств\n";

if (G == H) {

cout << "\nМножества равны!\n";

}

else {

cout << "\nМножества не равны!\n";

}

system("pause");

Money m\_element;

cout << "\nПоиск суммы во множестве G\nСумма\n> "; cin >> m\_element;

if (G > m\_element) {

cout << "\nСумма во множестве есть!\n";

}

else {

cout << "\nСуммы во множестве нет!\n";

}

system("pause");

size = G.get\_size();

cout << "\nДоступ к элементу множества G\n\nНомер элемента";

do {

cout << "\n> ";

cin >> number;

if (number > size) {

cout << "\nВведите индекс меньший длины множества\n";

}

if (number < 1) {

cout << "\nВведите положительный индекс\n";

}

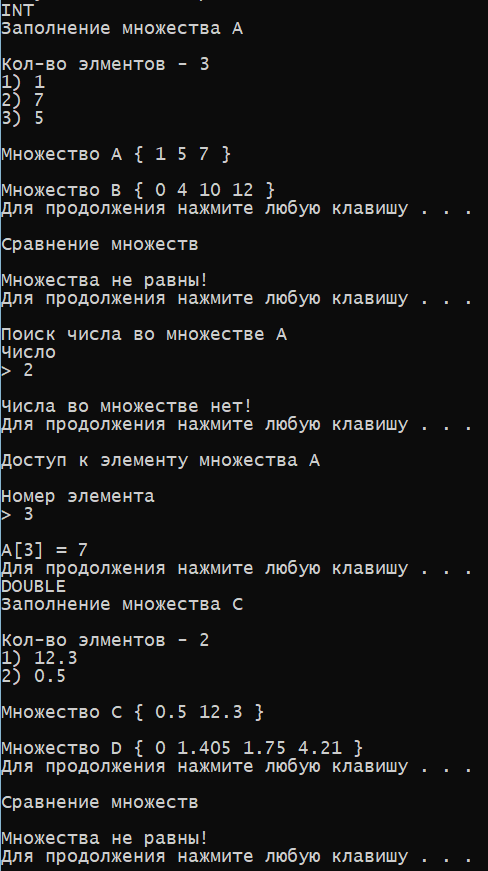
} while (number > size || number < 1);

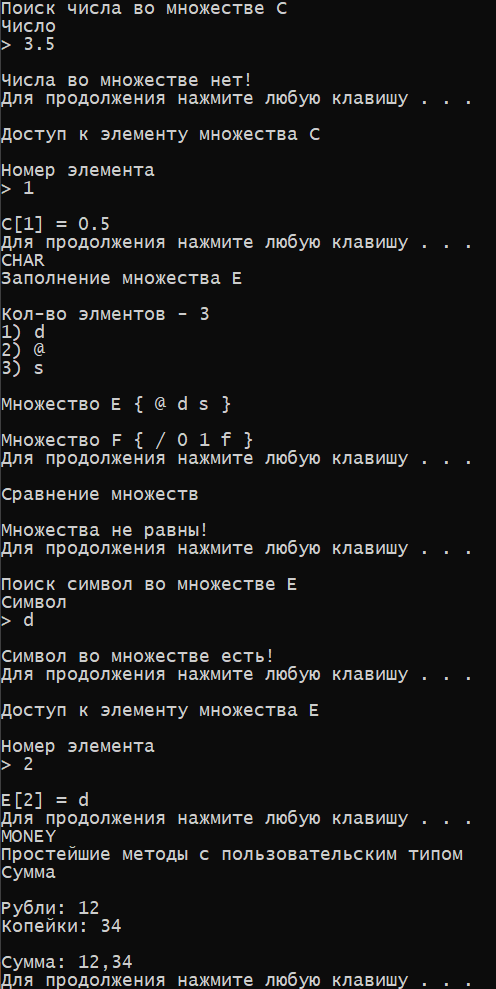
cout << "\nG[" << number << "] = " << G[number - 1] << endl;

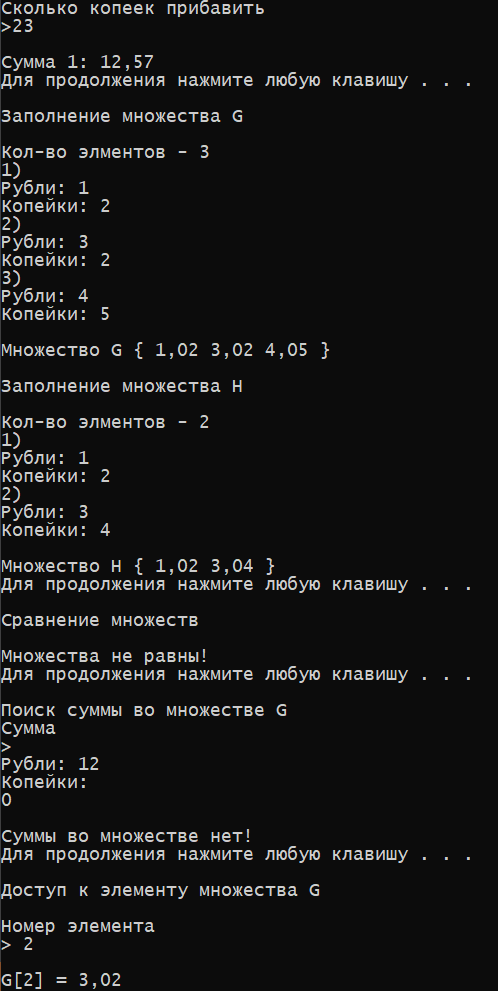
system("pause");

}

# **Скриншоты тестов**

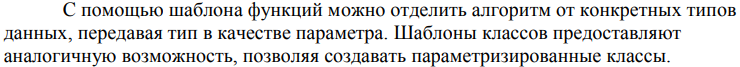




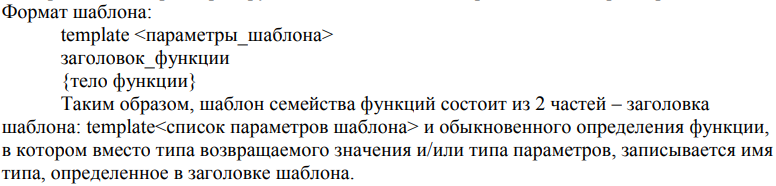


**Контрольные вопросы**

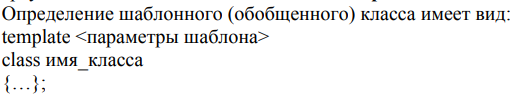














Параметры шаблона функции – это параметризированные типы данных, в зависимости от передаваемых данных в параметр.

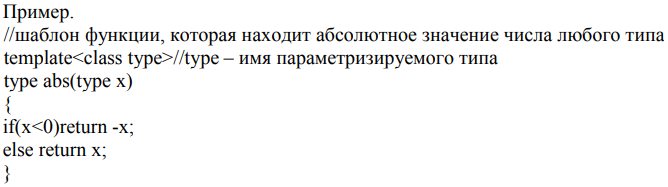


Шаблон функции объявляется один раз. Шаблон семейства функции состоит из 2 частей – заголовка шаблона и обыкновенного определения функции.

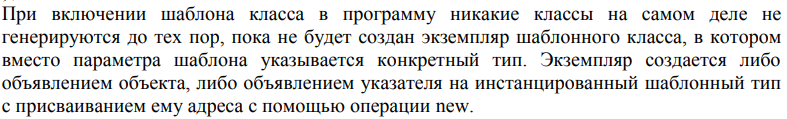


Параметр шаблона записывается в определении шаблона: template<typename Type>, далее в описании переменных и методов вместо типа данных необходимо записывать Type.



Да, можно. 







Компонентные функции параметризованного класса автоматически являются параметризованными.



Дружественные функции, которые описываются в параметризованном классе, не являются автоматически параметризованными функциями, то есть по умолчанию такие функции являются дружественными для всех классов, которые организуются по данному шаблону.



Шаблоны функций, которые являются членами классов, нельзя описывать как virtual.



Реализация компонентной функции шаблона класса, которая находится вне определения шаблона класса, должна включать дополнительно следующие два элемента:

* Определение должно начинаться с ключевого слова template, за которым следует такой же список\_параметров\_типов в угловых скобках, какой указан в определении шаблона класса.
* За именем\_класса, предшествующим операции области видимости (::), должен следовать список\_имен\_параметров шаблона.

template<список\_типов>

тип\_возвр\_значения имя\_класса<список\_имен\_параметров>::имя\_функции(список\_параметров)

{ ... }



Процесс генерации компилятором определения конкретного класса по шаблону класса и аргументам шаблона называется инстанцированием шаблона.



На этапе создания объекта перегруженного класса создаётся определение класса по шаблону.