**Лабораторная работа №7**

Тема: "Шаблоны функций и шаблоны классов"

Задание 1.

**1. Какие ошибки допущены в следующих объявлениях?**

**template <class T, class T> T f(T x);**

**template <class T1, T2> void f(T1 x);**

**template <class T> T f(int x);**

**inline template <class T> T f(T x, T y);**

*Ответ:*

В каждом из данных объявлений функций содержатся ошибки, которые делают их некорректными. Вот исправления для каждого из них:

1. Ошибка в первом объявлении функции:

template <class T> T f(T x);

1. Ошибка во втором объявлении функции:

template <class T1, class T2> void f(T1 x);

1. Ошибка в третьем объявлении функции, так как ключевое слово "template" и возвращаемый тип должны идти перед списком параметров:

template <class T> T f(T x);

1. в четвёртом объявлении функции, так как ключевое слово "inline" должно идти перед ключевым словом "template":

template <class T> inline T f(T x, T y);

После внесения указанных исправлений, объявления функций будут выглядеть следующим образом:

template <class T> T f(T x);

template <class T1, class T2> void f(T1 x);

template <class T> T f(int x);

template <class T> inline T f(T x, T y);

**2. Написать тестовую программу для функции swap и попробовать ее**

**вызовы с различными типами аргументов (значения переменных - числа,**

**символы, строки).**

*Ответ:*

*#include <iostream>*

*#include <string>*

*// Функция для обмена значениями двух переменных разных типов*

*template <typename T>*

*void swap(T& a, T& b) {*

*T temp = a;*

*a = b;*

*b = temp;*

*}*

*int main() {*

*// Тестирование swap для чисел*

*int x = 5, y = 10;*

*std::cout << "Before swap: x = " << x << ", y = " << y << std::endl;*

*swap(x, y);*

*std::cout << "After swap: x = " << x << ", y = " << y << std::endl;*

*// Тестирование swap для символов*

*char char1 = 'A', char2 = 'B';*

*std::cout << "Before swap: char1 = " << char1 << ", char2 = " << char2 << std::endl;*

*swap(char1, char2);*

*std::cout << "After swap: char1 = " << char1 << ", char2 = " << char2 << std::endl;*

*// Тестирование swap для строк*

*std::string str1 = "Hello", str2 = "World";*

*std::cout << "Before swap: str1 = " << str1 << ", str2 = " << str2 << std::endl;*

*swap(str1, str2);*

*std::cout << "After swap: str1 = " << str1 << ", str2 = " << str2 << std::endl;*

*return 0;*

*}*

**3. Написать программу, в которой определяется шаблон для функции**

**max(x,y), возвращающей большее из значений x и y. Написать**

**специализированную версию функции max(char\*,char\*), возвращающую**

**"большую" из передаваемых ей символьных строк. В каждой из функций**

**предусмотреть вывод сообщения о том, что вызвана шаблонная или**

**специализированная функция и вывод найденного большего. Проверить**

**работу программы на трех примерах**

**max('a','1'), max(0,1), max("Hello","World").**

*Ответ:*

*#include <iostream>*

*#include <cstring>*

*template <typename T>*

*T max(T x, T y) {*

*std::cout << "Вызвана шаблонная функция max" << std::endl;*

*return x > y ? x : y;*

*}*

*template <>*

*const char\* max(const char\* x, const char\* y) {*

*std::cout << "Вызвана специализированная функция max" << std::endl;*

*return std::strcmp(x, y) > 0 ? x : y;*

*}*

*int main() {*

*char a = 'a', b = '1';*

*int c = 0, d = 1;*

*const char\* str1 = "Hello";*

*const char\* str2 = "World";*

*std::cout << "max('a', '1') = " << max(a, b) << std::endl;*

*std::cout << "max(0, 1) = " << max(c, d) << std::endl;*

*std::cout << "max(\"Hello\", \"World\") = " << max(str1, str2) << std::endl;*

*return 0;*

*}*

**Задание 2.**

**1. Реализовать класс комплексных чисел complex. Переопределить**

**операции сложения, вычитания и ввод/вывод в поток.**

*Ответ:*

*#include <iostream>*

*class Complex {*

*private:*

*double real;*

*double imaginary;*

*public:*

*// Конструктор по умолчанию*

*Complex() : real(0), imaginary(0) {}*

*// Конструктор с параметрами*

*Complex(double r, double i) : real(r), imaginary(i) {}*

*// Переопределение операции сложения*

*Complex operator+(const Complex& other) const {*

*return Complex(real + other.real, imaginary + other.imaginary);*

*}*

*// Переопределение операции вычитания*

*Complex operator-(const Complex& other) const {*

*return Complex(real - other.real, imaginary - other.imaginary);*

*}*

*// Переопределение операции ввода*

*friend std::istream& operator>>(std::istream& in, Complex& complex) {*

*std::cout << "Введите действительную часть: ";*

*in >> complex.real;*

*std::cout << "Введите мнимую часть: ";*

*in >> complex.imaginary;*

*return in;*

*}*

*// Переопределение операции вывода*

*friend std::ostream& operator<<(std::ostream& out, const Complex& complex) {*

*if (complex.imaginary >= 0) {*

*out << complex.real << " + " << complex.imaginary << "i";*

*} else {*

*out << complex.real << " - " << -complex.imaginary << "i";*

*}*

*return out;*

*}*

*};*

*int main() {*

*Complex a, b, result;*

*std::cout << "Введите первое комплексное число:\n";*

*std::cin >> a;*

*std::cout << "Введите второе комплексное число:\n";*

*std::cin >> b;*

*// Сложение и вычитание*

*result = a + b;*

*std::cout << "Сумма: " << result << std::endl;*

*result = a - b;*

*std::cout << "Разность: " << result << std::endl;*

*return 0;*

*}*

**2. Создать шаблон класса матриц. Переопределить операции**

**сложения, вычитания, присваивания и ввод/вывод в поток.**

*Ответ:*

*#include <iostream>*

*#include <vector>*

*template <typename T>*

*class Matrix {*

*private:*

*std::vector<std::vector<T>> data;*

*size\_t rows;*

*size\_t cols;*

*public:*

*Matrix(size\_t rows, size\_t cols) : rows(rows), cols(cols) {*

*data.resize(rows, std::vector<T>(cols, 0));*

*}*

*// Переопределение оператора сложения*

*Matrix<T> operator+(const Matrix<T>& other) {*

*if (this->rows != other.rows || this->cols != other.cols) {*

*throw std::invalid\_argument("Матрицы имеют разные размеры");*

*}*

*Matrix<T> result(this->rows, this->cols);*

*for (size\_t i = 0; i < rows; ++i) {*

*for (size\_t j = 0; j < cols; ++j) {*

*result.data[i][j] = this->data[i][j] + other.data[i][j];*

*}*

*}*

*return result;*

*}*

*// Переопределение оператора вычитания*

*Matrix<T> operator-(const Matrix<T>& other) {*

*if (this->rows != other.rows || this->cols != other.cols) {*

*throw std::invalid\_argument("Матрицы имеют разные размеры");*

*}*

*Matrix<T> result(this->rows, this->cols);*

*for (size\_t i = 0; i < rows; ++i) {*

*for (size\_t j = 0; j < cols; ++j) {*

*result.data[i][j] = this->data[i][j] - other.data[i][j];*

*}*

*}*

*return result;*

*}*

*// Переопределение оператора присваивания*

*Matrix<T>& operator=(const Matrix<T>& other) {*

*if (this == &other) {*

*return \*this; // Самоприсваивание*

*}*

*if (this->rows != other.rows || this->cols != other.cols) {*

*throw std::invalid\_argument("Матрицы имеют разные размеры");*

*}*

*for (size\_t i = 0; i < rows; ++i) {*

*for (size\_t j = 0; j < cols; ++j) {*

*this->data[i][j] = other.data[i][j];*

*}*

*}*

*return \*this;*

*}*

*// Переопределение оператора ввода*

*friend std::istream& operator>>(std::istream& in, Matrix<T>& matrix) {*

*for (size\_t i = 0; i < matrix.rows; ++i) {*

*for (size\_t j = 0; j < matrix.cols; ++j) {*

*in >> matrix.data[i][j];*

*}*

*}*

*return in;*

*}*

*// Переопределение оператора вывода*

*friend std::ostream& operator<<(std::ostream& out, const Matrix<T>& matrix) {*

*for (size\_t i = 0; i < matrix.rows; ++i) {*

*for (size\_t j = 0; j < matrix.cols; ++j) {*

*out << matrix.data[i][j] << " ";*

*}*

*out << std::endl;*

*}*

*return out;*

*}*

*};*

*int main() {*

*Matrix<int> matrix1(2, 2);*

*Matrix<int> matrix2(2, 2);*

*std::cout << "Введите элементы матрицы 1:\n";*

*std::cin >> matrix1;*

*std::cout << "Введите элементы матрицы 2:\n";*

*std::cin >> matrix2;*

*Matrix<int> sum = matrix1 + matrix2;*

*Matrix<int> diff = matrix1 - matrix2;*

*std::cout << "Сумма матриц:\n" << sum;*

*std::cout << "Разность матриц:\n" << diff;*

*return 0;*

*}*

**3. Проверить работоспособность шаблона класса матриц для данных**

**int, float, complex.**

**Задание 3.**

**1. Реализовать шаблон класса стек, в котором размер стека**

**задается параметром шаблона. Протестировать работоспособность класса.**

*Ответ:*

#include <iostream>

#include <stdexcept>

template <typename T, int Size>

class Stack {

private:

T data[Size];

int top;

public:

Stack() : top(-1) {}

void push(const T& value) {

if (top < Size - 1) {

data[++top] = value;

} else {

throw std::overflow\_error("Stack is full");

}

}

void pop() {

if (top >= 0) {

--top;

} else {

throw std::underflow\_error("Stack is empty");

}

}

T& peek() {

if (top >= 0) {

return data[top];

} else {

throw std::underflow\_error("Stack is empty");

}

}

bool isEmpty() {

return top == -1;

}

bool isFull() {

return top == Size - 1;

}

};

int main() {

Stack<int, 5> intStack;

try {

intStack.push(1);

intStack.push(2);

intStack.push(3);

intStack.push(4);

intStack.push(5);

while (!intStack.isEmpty()) {

std::cout << intStack.peek() << " ";

intStack.pop();

}

std::cout << std::endl;

intStack.pop(); // This should throw an underflow error

} catch (const std::exception& e) {

std::cerr << "Exception: " << e.what() << std::endl;

}

return 0;

}

Этот код определяет шаблон класса Stack, который принимает два параметра: тип данных T и размер стека Size. Класс реализует стек с заданным размером и предоставляет методы для добавления элементов (push), удаления элементов (pop), просмотра верхнего элемента (peek), а также проверки на пустоту (isEmpty) и переполнение (isFull) стека.

Вы можете использовать этот шаблон класса Stack для разных типов данных и с разными размерами стека, как показано в функции main.