МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа № 9

по дисциплине: ООП
тема: «Использование стандартной библиотеки шаблонов STL»

Выполнил: ст. группы ПВ202
Аладиб язан
Проверил:
Буханов Д.Г.

Белгород 2022

Цель работы: знакомство со стандартной библиотекой шаблонов в C++; получение навыков использования классов контейнеров, итераторов, алгоритмов.

Содержание работы

Разработать программное обеспечения для решения соответствующего варианта. Оформить отчет. Для реализации поставленных задач требуется использовать следующие библиотеки классов: list, vector, queue, iostream, algorithm, set, iterator, map, stack.

Задание Варианта :

Разработать программное обеспечение для решения следующей задачи: преобразование математического выражения в обратную польскую запись, для хранения использовать stack, выполнить вычисления. Реализовать вычисление сложения, вычитания, и других арифметических операций над stack.

Выполнение работы:

```
#include <iostream>
#include <stack>
#include <string>
#include <cctype>

using namespace std;

int getPriority(char operation) {
  if (operation == '+' || operation == '-')
    return 1;
  else if (operation == '*' || operation == '/')
    return 2;
  else
    return 0;
}

int Calculate(int op1, int op2, char op) {
  if (op == '+')
    return op1 + op2;
  else if (op == '-')
```

```
    return op1 - op2;
  else if (op == '*')
    return op1 * op2;
  else if (op == '/')
    return op1 / op2;
  else
    return 0;
}

string getReversePolish(const string& infixInputString) {
  stack<char> operatorStack;
  string postfix;
  for (char current : infixInputString) {
    if (isdigit(current)) {
      postfix += current;
    } else if (current == '(') {
      operatorStack.push(current);
    } else if (current == ')') {
      while (!operatorStack.empty() && operatorStack.top() != '(') {
        postfix += operatorStack.top();
        operatorStack.pop();
      }
      if (!operatorStack.empty() && operatorStack.top() == '(') {
        operatorStack.pop();
      }
    } else {
      while (!operatorStack.empty() && getPriority(current) <=
                                       getPriority(operatorStack.top())) {
        postfix += operatorStack.top();
        operatorStack.pop();
      }
      operatorStack.push(current);
    }
  }
  while (!operatorStack.empty()) {
    postfix += operatorStack.top();
    operatorStack.pop();
  }
  return postfix;
}

int EvaluatePostfixExpression(const string& postfix) {
  stack<int> operandStack;
  for (char current : postfix) {
    if (isdigit(current)) {
      operandStack.push(current - '0');
    } else {
      int op2 = operandStack.top();
      operandStack.pop();
      int op1 = operandStack.top();
      operandStack.pop();
```

```
      int result = Calculate(op1, op2, current);
      operandStack.push(result);
    }
  }
  return operandStack.top();
}

void getSolution(string &infix) {
  cout << "normal exp: " << infix << endl;
  string postfix = getReversePolish(infix);
  cout << "reverse polish exp: " << postfix << endl;
  int result = EvaluatePostfixExpression(postfix);
  cout << "result: " << result << endl << endl;
}

int main() {
  string infix = "1+2*3-7";
  getSolution(infix);
  infix = "(5+6)*8/2";
  getSolution(infix);
  infix = "2+2*2";
  getSolution(infix);
  return 0;
}
```