Выполнила

студентка группы КТбо1-2 Н. А. Свидич

Принял

доцент кафедры САиТ В. С. Лапшин

ЛАБОРАТОРНАЯ РАБОТА №1

по дисциплине «Основы алгоритмизации и программирования»

**«РАБОТА СО СТЕКОМ»**

Таганрог 2023

Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

(ФГАОУ ВО «ЮФУ»)

Институт компьютерных технологий и информационной безопасности

Кафедра системного анализа и телекоммуникаций

# Содержание

Содержание 2

Техническое задание 3

Цель задания 3

Задача 3

Ход работы 4

Алгоритм 4

Написание программы 5

Пример работы программы 5

Вывод 7

Листинг 8

StackLab.cpp 8

# Техническое задание

## Цель задания

Целью данной лабораторной работы является получение практических навыков работы со стеком на примере построения обратной польской записи арифметических выражений.

## Задача

Написать программу, реализующую перевод в ОПЗ простых арифметических и логических выражений. Ввод и вывод данных организовать во внешние файлы. Запрограммировать работу стека на основе двух функций push () и pop (), программно реализовать алгоритм Дейкстры.

# Ход работы

## Алгоритм

1. Получить входные данные
2. Перебрать строку входных данных на составляющие символы:
   1. Если символ является операндом: проверить операнд на приоритетность выполнения между другими операциями выражения, и, в зависимости от приоритета, заполнить стек
   2. Если символ является числом, заполнить стек
3. Вывести полученную строку

## Написание программы

Переменные:

char stack – стек для хранения значений.

int peek – идентификатор последнего элемента стека.

char line – строка с входными данными.

char result – строка с данными вывода.

int id – идентификатор элементов строк.

input – переменная, хранящая путь к файлу вводных данных.

output – переменная, хранящая путь к файлу выходных данных

Функции:

Push – добавление элемента в стек.

Pop – удаление элемента из стека.

Operation – проверка на приоритетность операнда.

Выполнение:

В начале работы выполняется инициализация всех переменных и проверка файлов на корректное открытие.

Затем начинается построчное считывание строк из исходного файла и перебор считанной строки по символам на наличие символов операций или чисел.

В зависимости от того, чем является считанный символ, заполняется стек: смежные числа добавляются в стек поочередно, пропуская операнды, если в исходной строке алгебраической записи над ними совершалось одно и тоже действие; операнды заполняют стек в зависимости от приоритетности и чисел, над которыми выполняются соответствующие им действия.

Далее происходит вывод полученной форматированной строки в файл вывода.

## Пример работы программы

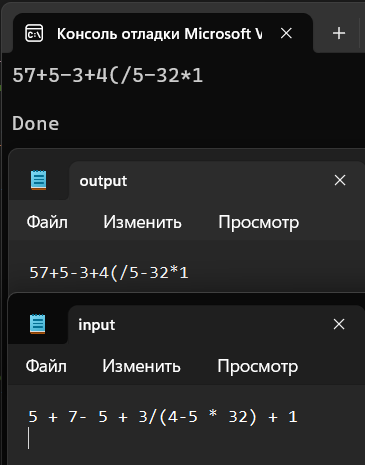


Рисунок 1 – Результат работы программы

# Вывод

По итогу лабораторной работы была получена программа, которая выполняет алгоритм преобразования арифметического выражения в форму обратной польской записи.

В ходе лабораторной работы я ознакомилась с понятием стека, методами стека pop и push, реализовала свой стек для решения поставленной задачи.

# Листинг

## StackLab.cpp

1. #include <string.h>
2. #include <cstdio>
3. #pragma warning(disable:4996)
4. #define MAX\_SIZE 256
5. #define INPUT\_FN "..\\input.txt"
6. #define OUTPUT\_FN "..\\output.txt"
7. // Стек и id последнего элемента
8. char stack[MAX\_SIZE];
9. int peek = 0;
10. // Добавить элемент в стек
11. void push(const char с)
12. {
13. if (peek >= MAX\_SIZE)
14. printf("Stack overflow\n");
15. else
16. {
17. stack[peek] = с;
18. peek++;
19. }
20. }
21. // Удалить элемент из стека
22. char pop()
23. {
24. if (peek == 0)
25. {
26. return -1;
27. }
28. peek--;
29. return stack[peek];
30. }
31. // Значение приоритета операции
32. int operation(char c)
33. {
34. switch (c)
35. {
36. case '(':
37. return 0;
38. case ')':
39. return 1;
40. case '+': case '|': case '-':
41. return 2;
42. case '\*': case '&': case '/': case '^':
43. return 3;
44. default:
45. return -1;
46. }
47. }
48. int main() {
49. FILE\* input = fopen(INPUT\_FN, "r");
50. FILE\* output = fopen(OUTPUT\_FN, "w");
51. if (input == NULL || output == NULL) {
52. printf("Files error\n");
53. return -1;
54. }
55. char line[MAX\_SIZE], result[MAX\_SIZE];
56. int id = 0;
58. while (fgets(line, MAX\_SIZE, input) != NULL) {
59. // Перебор строки-выражения по символам
60. for (int i = 0; i < strlen(line); ++i)
61. {
62. // Если знак - операнд, перебираем строку в поиске более приоритетных значений операндов. Если цифра - запись в буфер
63. if (line[i] == '+' || line[i] == '-' || line[i] == '\*' || line[i] == '/' || line[i] == '|' || line[i] == '&')
64. {
66. while (peek != 0 && stack[peek - 1] >= operation(line[i]))
67. {
68. result[id++] = pop();
69. }
70. push(line[i]);
71. }
72. else if (line[i] == '(')
73. {
74. push(line[i]);
75. }
76. else if (line[i] == ')')
77. {
78. // Пока не дойдём до скобки, добавляем символы в стек
79. while (peek != 0 && stack[peek - 1] != '(')
80. {
81. result[id++] = pop();
82. }
83. pop();
84. }
85. else if (line[i] != '\n' && line[i] != ' ')
86. {
87. result[id++] = line[i];
88. }
89. }
90. result[id++] = '\0';
91. printf("%s", result);
92. fputs(result, output);
93. }
94. fclose(input);
95. fclose(output);
96. printf("\nDone\n");
97. return 0;
98. }