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

**ЛАБОРАТОРНАЯ РАБОТА № 1**

по дисциплине «Основы алгоритмизации и программирования»

на тему:

«Работа со стеком»

Выполнил

студент группы КТбо1-2 И. Н. Акимов

Принял

доцент кафедры САиТ В. С. Лапшин

Таганрог 2023

Содержание

1 Техническое задание 3

1.1 Цель работы 3

1.2 Задание 3

2 Ход работы 4

2.1 Алгоритм 4

2.2 Блок-схема 6

2.3 Написание программы 7

3 Вывод 9

4 Приложение А 10

4.1 Листинг программы 10

# Техническое задание

## Цель работы

Цель данной лабораторной работы выработка практических навыков работы со стеком на примере построения обратной польской записи арифметических и логических выражений.

## Задание

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. Считать очередной символ входной строки, пока не будет

достигнут конец выражения:

1.1. Если символ операнд, то он перемещается в

выходную строку.

1.2. Если символ операция и стек пуст, то он

проталкивается в стек.

1.3. Если символ операция и стек не пуст, а приоритет

символа операции меньше, чем у символа операции

вершины стека, то прочитанный символ проталкивается и

стек.

1.4. Если символ операция и стек не пуст, а приоритет

символа вершины стека, меньше приоритета входного

символа, то из стека в выходную строку выталкиваются

все символы операций с меньшим или равным

приоритету входного символа, после чего входной

символ операции заносится в стек.

1.5. Если входной символ левая открывающая скобка, то

она проталкивается в стек.

1.6. Если входной символ правая закрывающая скобка, то

она выталкивает из стека в выходную строку все символы

до левой открывающей скобки. сами скобки

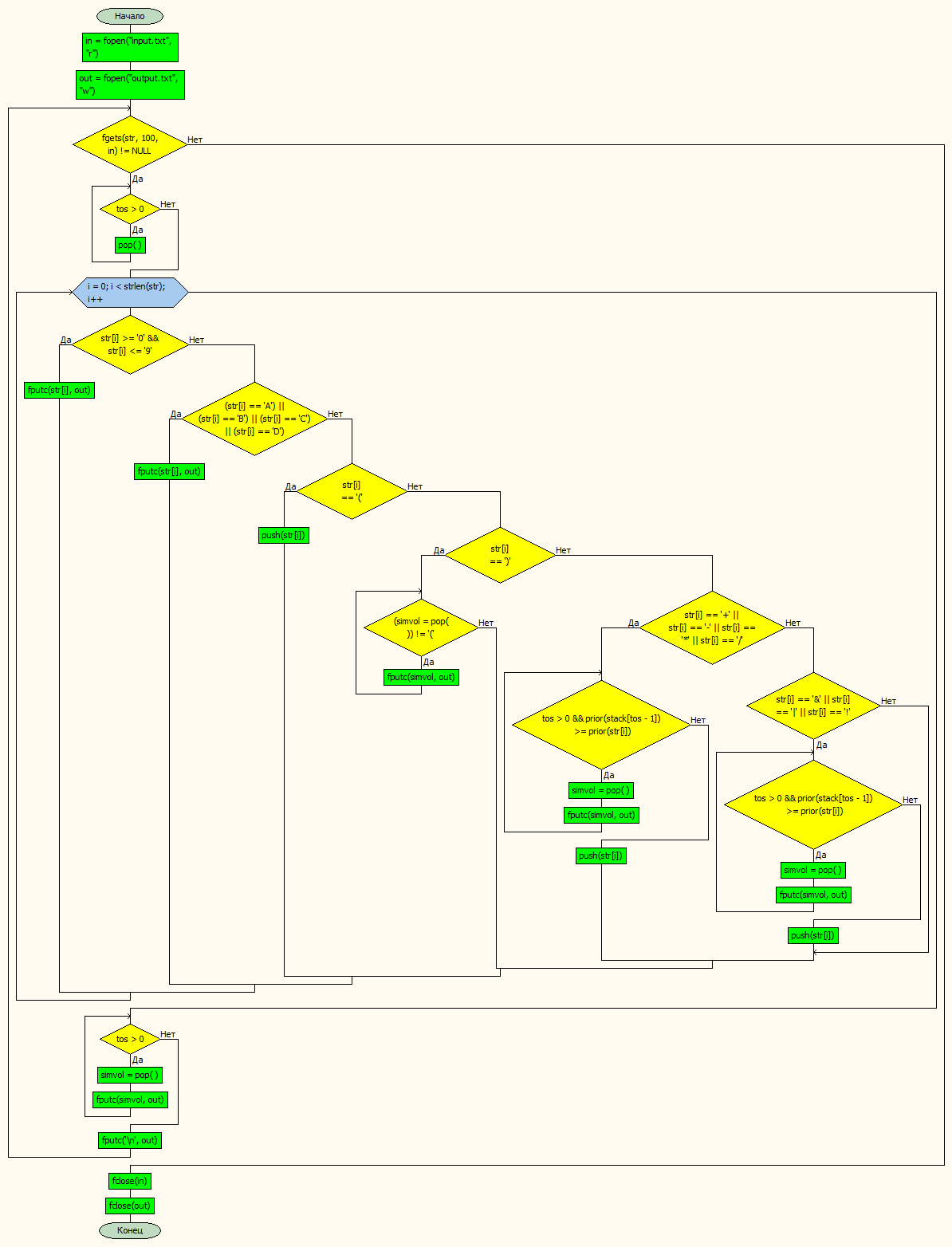
уничтожаются и в выходную строку не попадают.

1.7. Если входной символ - маркер конца строки, то из

стека в выходную строку выталкиваются все символы.

2. Вывод результата выходной строки

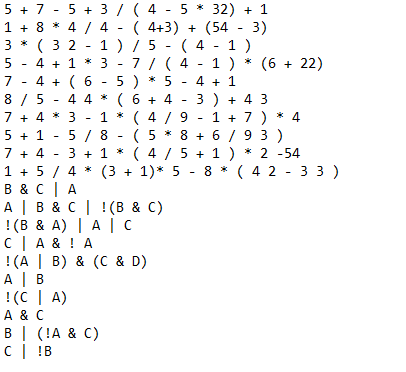
## Блок-схема



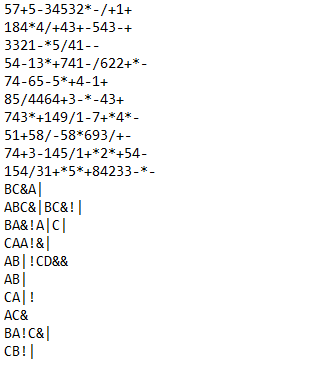
## Написание программы

1. Объявляем структуру стека
2. Объявляем функции для работы со стеком
3. Объявляем вспомогательные функции для перевода в обратную польскую нотацию
4. Объявляем функцию перевода в обратную польскую нотацию. Выполняем в ней все действия данного алгоритма, описанные в методичке.
5. Объявляем функцию main
6. Создаем переменные для работы с файлами: первая представляет собой объект входного потока данных (файл входных данных), вторая – выходного (файл выходных данных)
7. Открываем файлы и начинаем чтение выражений из файла построчно при помощи функции «fgets(string, n, filestream)», передаем их в функцию перевода и записываем результат в выходной файл

Закрываем файлы и завершаем выполнение программы



1. – Скриншот входных данных



1. – Скриншот результатов выполнения программы

# Вывод

Выполнив лабораторную работу, я выработал практические навыки работы со стеком на примере построения обратной польской записи арифметических и логических выражений.

# Приложение А

## Листинг программы

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <Windows.h>

#include <ctype.h>

int const MAX = 100;

int stack[MAX];

int tos = 0;

int prior(char str)

{

switch (str)

{

case '!':

return 4;

case '\*':

case '/':

case '&':

return 3;

case '-':

case '+':

case '|':

return 2;

case '(':

return 1;

}

}

void push(int i) {

if (tos >= MAX) {

printf("Стек полон\n"); return;

}

stack[tos] = i;

tos++;

}

int pop() {

tos--;

if (tos < 0) {

printf("Стек пуст\n"); return 0;

}

return stack[tos];

}

int main() {

FILE\* in = fopen("input.txt", "r");

FILE\* out = fopen("output.txt", "w");

char str[100], element;

while (fgets(str, 100, in) != NULL) {

while (tos > 0) {

pop();

}

for (int i = 0; i < strlen(str); i++) {

if (str[i] >= '0' && str[i] <= '9') {

fputc(str[i], out);

}

else if ((str[i] == 'A') || (str[i] == 'B') || (str[i] == 'C') || (str[i] == 'D')) {

fputc(str[i], out);

}

else if (str[i] == '(') {

push(str[i]);

}

else if (str[i] == ')') {

while ((element = pop()) != '('){

fputc(element, out);

}

}

else if (str[i] == '+' || str[i] == '-' || str[i] == '\*' || str[i] == '/') {

while (tos > 0 && prior(stack[tos - 1]) >= prior(str[i])) {

element = pop();

fputc(element, out);

}

push(str[i]);

}

else if (str[i] == '&' || str[i] == '|' || str[i] == '!') {

while (tos > 0 && prior(stack[tos - 1]) >= prior(str[i])) {

element = pop();

fputc(element, out);

}

push(str[i]);

}

}

while (tos > 0) {

element = pop();

fputc(element, out);

}

fputc('\n', out);

}

fclose(in);

fclose(out);

}