Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования (ОАиП)

ОТЧЕТ

по лабораторной работе №11

Тема работы: Рекурсия

Выполнил

студент: гр. 151004 Данилов Ф.А.

Проверил: Фадеева Е.П.

Минск 2022

Содержание

[1 Постановка задачи 4](#_Toc103456016)

[2 Описание алгоритмов решения задачи 5](#_Toc103456017)

[3 Структура данных 6](#_Toc103456018)

[3.1 Структура данных основной программы 6](#_Toc103456019)

[3.2 Структура данных алгоритма replace 6](#_Toc103456020)

[3.3 Структура данных алгоритма eval 6](#_Toc103456021)

[3.4 Структура данных алгоритма parseBoolExp 7](#_Toc103456022)

[4 Схема алгоритма решения задачи по ГОСТ 19.701-90 8](#_Toc103456023)

[4.1 Схема алгоритма подпрограммы replace 9](#_Toc103456024)

[4.2 Схема алгоритма подпрограммы eval 10](#_Toc103456025)

[4.3 Схема алгоритма подпрограммы parseBoolExp 11](#_Toc103456026)

[5 Результаты расчетов программы 13](#_Toc103456027)

[Приложение А 14](#_Toc103456028)

[Приложение Б 18](#_Toc103456029)

[Приложение В 19](#_Toc103456030)

# Постановка задачи

Без ошибок записано некое логическое выражение следующего вида:

<Логическое\_выражение>::= True | False | (<Операция> (<Операнды>)).

<Операция >::= Not | And | Or.

<Операнды>::= <Операнд> {,<Операнд>}.

<Операнд>::= <Логическое\_выражние>.

Ввести это выражение и вычислить его значение. У операций And и Or может быть любое число операндов, у Not – только один операнд.

# Описание алгоритмов решения задачи

Таблица 1 – Описание алгоритмов решения задачи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  п.п. | Наименование алгоритма | Назначение  алгоритма | Формальные параметры | Предпологаемый тип реализации |
|  | Основной алгоритм | Ввод строки и проверка правильности. Вызывает подпрограммы:  parseBoolExp(s,  result),  replace(str, result) |  |  |
|  | replace  (str,  result) | Упрощение строки путем  замены слов на символы) | str – получает от фактического параметра адрес;  result – получает от фактического параметра адрес, возвращаемый функцией параметр | Функция. result – возвращаемый параметр |
|  | eval  (bools,  oper,  result) | Вычисление  логических  подвыражений | bools, oper – получают от фактического параметра адрес;  result – получает от фактического параметра адрес, возвращаемый функцией параметр | Функция. result – воз-вращаемый параметр |
|  | parseBoolExp (s,  result) | Вычисление всего  логического  выражения. Вызывает под-программы:  eval(bools, oper, result) | s – получает от фактического параметра адрес;  result – получает от фактического параметра адрес, возвращаемый функцией параметр | Функция. result – воз-вращаемый параметр |

# Структура данных

## Структура данных основной программы

Таблица 2 – Структура данных основной программы

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| inputStr | string | Вводимая пользователем строка |
| cond | boolean | Конечный результат расчетов |

## Структура данных алгоритма replace

Таблица 3 – Структура данных алгоритма replace(str, result)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| words | array [1..5] of string | Массив слов, которые требуется заменить. Принимает значения ('And', 'Or', 'Not', 'True', 'False') | Локальный |
| chars | array [1..5] of string | Массив символов. Принимает значения ('&', '|', '!', 't', 'f') | Локальный |
| str | string | Изменяемая строка | Формальный |
| result | string | Результат преобразования | Формальный |
| i | integer | Итератор и индекс | Локальный |
| k | integer | Позиция замены | Локальный |

## 3.3 Структура данных алгоритма eval

Таблица 4 – Структура данных алгоритма eval(bools, oper, result)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| bools | array [0..N-1] of boolean | Операнды выражения | Формальный |
| oper | char | Оператор выражения | Формальный |
| i | integer | Итератор и индекс | Локальный |
| result | boolean | Результат операции | Формальный |

## Структура данных алгоритма parseBoolExp

Таблица 5 – Структура данных алгоритма parseBoolExp(s, result)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| s | string | Вычисляемая строка | Формальный |
| result | boolean | Результат операции | Формальный |
| с | char | Символ строки | Локальный |
| op | char | Текущий оператор | Локальный |
| i | integer | Итератор и индекс | Локальный |
| num | integer | Число операндов | Локальный |
| isEnd | boolean | Условия остановки вычиления значения строки | Локальный |
| boolsArr | array [0..N-1] of boolean | Передаваемый для вычисления массив операндов | Локальный |

# Схема алгоритма решения задачи по ГОСТ 19.701-90

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1 – Схема алгоритма решения задачи по ГОСТ 19.701-90

## Схема алгоритма подпрограммы replace

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – Схема алгоритма подпрограммы replace по ГОСТ 19.701-90

## Схема алгоритма подпрограммы eval

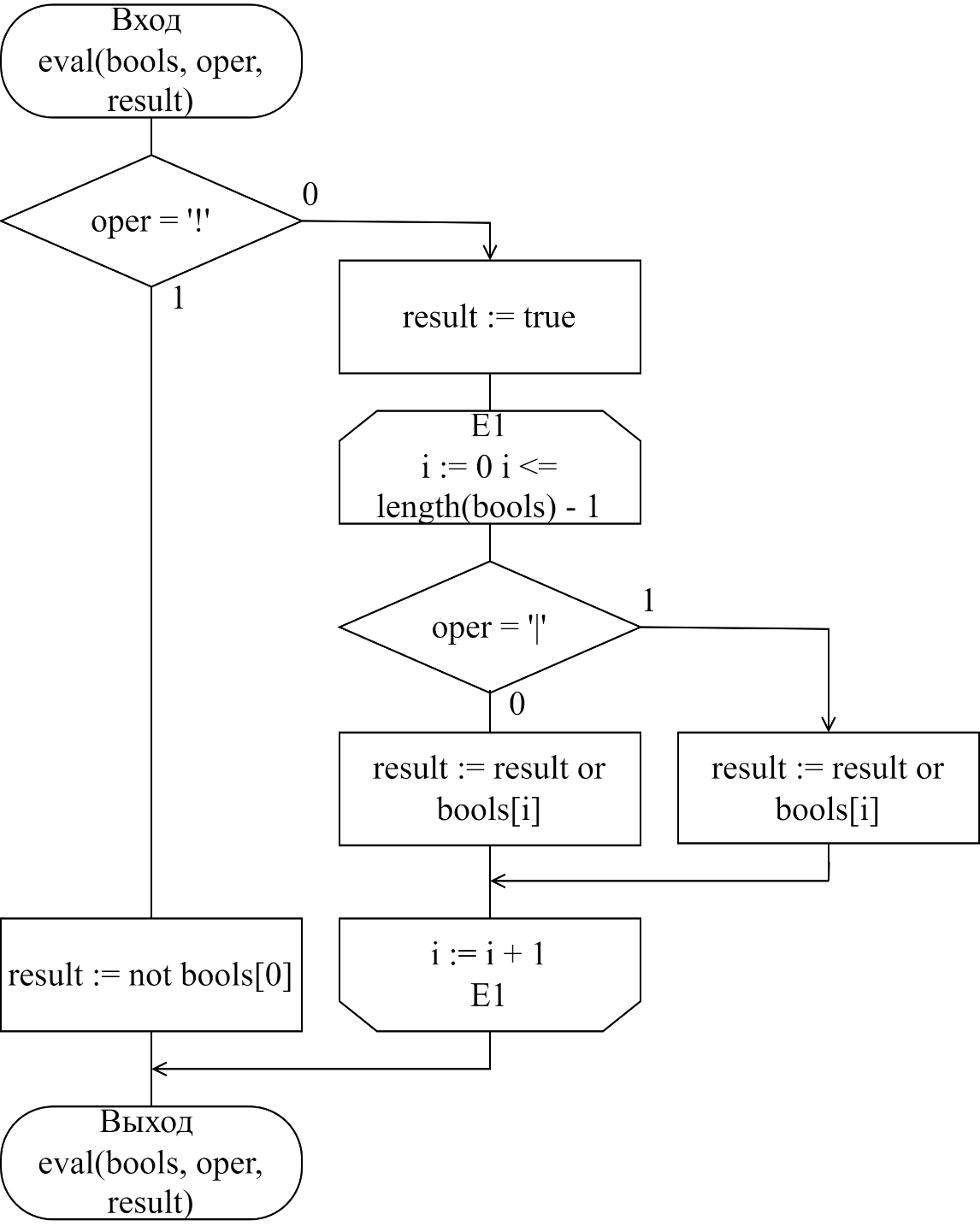


Рисунок 3 – Схема алгоритма подпрограммы eval по ГОСТ 19.701-90

## Схема алгоритма подпрограммы parseBoolExp

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4 – Схема алгоритма подпрограммы parseBoolExp по ГОСТ 19.701-90

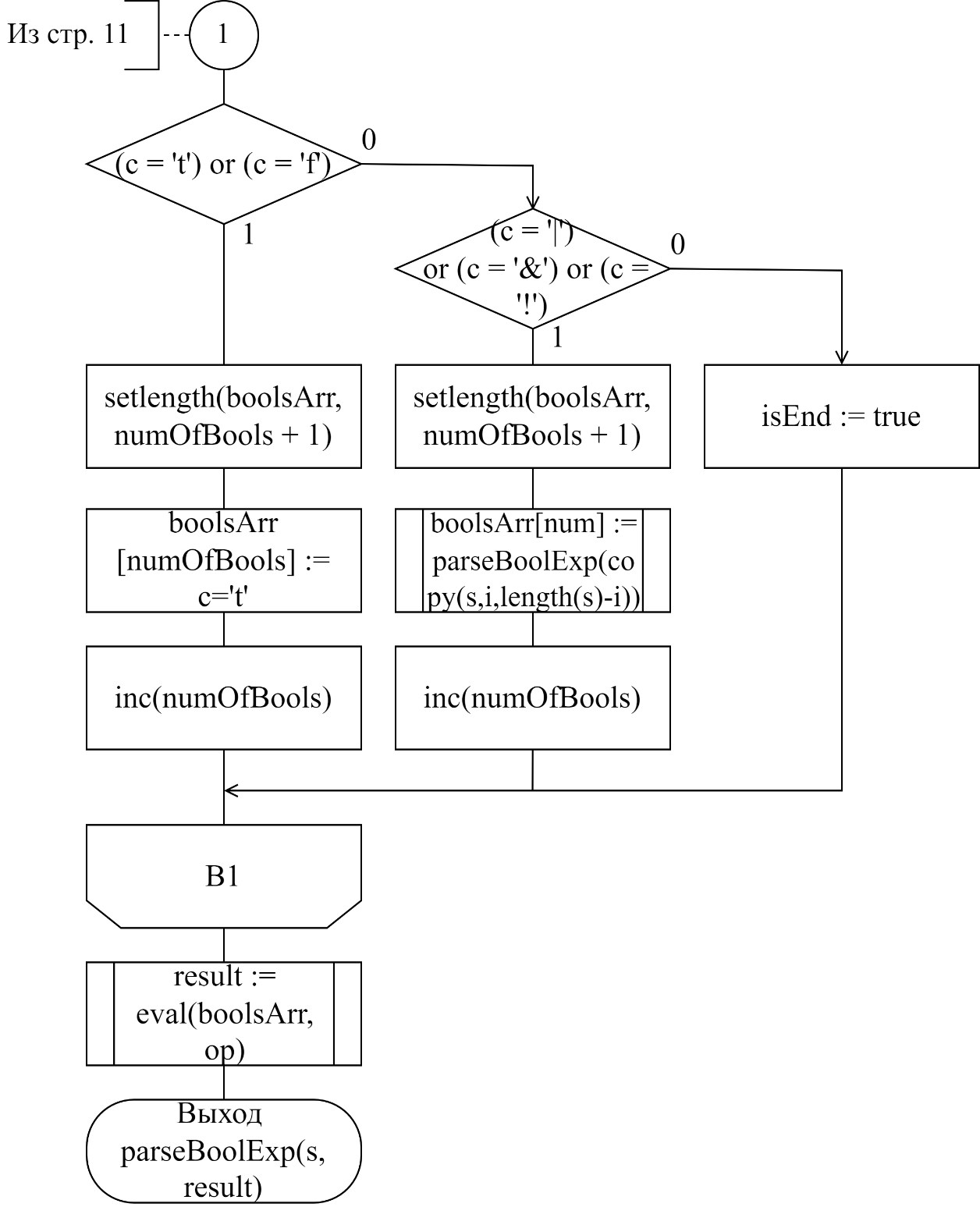


Рисунок 5 – Схема алгоритма подпрограммы parseBoolExp по ГОСТ 19.701-90

# Результаты расчетов программы



Рисунок 6 – Результаты расчётов программы

Приложение А

(обязательное)

Исходный код программы

program LAB11;

// Еhe program that calculates boolean string

// expression

// Console app

{$APPTYPE CONSOLE}

// Types declaration

type

TBoolsArr = array of boolean;

// Vars declaration

var

inputStr: string;

cond: boolean;

{ inputString - string that user inputs

cond – result of the evaluation }

// A function that makes expression easier to parse

function replace(var str: string): string;

{ s - formal parameter, string we replace }

// Consts declaration

const

words: array [1 .. 5] of string = ('And', 'Or',

'Not', 'True', 'False');

chars: array [1 .. 5] of string = ('&', '|', '!',

't', 'f');

{ words - words we replace

chars - words-replacers }

// Vars declaration

var

i, k: integer;

{ i - iterator and index

k - position }

begin

// Start of the cycle A1, replacing every word

for i := 1 to length(words) do

begin

// Start of the cycle A2, deleting and inserting

while pos(words[i], str) > 0 do

begin

k := pos(words[i], str);

delete(str, k, length(words[i]));

insert(chars[i], str, k);

end;

// End of the cycle A2

end;

// End of the cycle A1

result := str;

end;

// A function that calculates smaller expressions

function eval(const bools: TBoolsArr; const oper:

char): boolean;

{ bools - formal parameter, operands

oper - formal, operator }

// Vars declaration

var

i: integer;

{ i - iterator and index }

begin

// If not there are one operand

if oper = '!' then

result := not bools[0]

else

// Calculating other possible ways

begin

result := true;

for i := 0 to length(bools) - 1 do

if oper = '|' then

result := result or bools[i]

else

result := result and bools[i];

end;

end;

// A main function that parses expression

function parseBoolExp(const s: string): boolean;

{ s - formal parameter, string we parse }

// Vars declaration

var

c, op: char;

boolsArr: TBoolsArr;

i, num: integer;

isEnd: boolean;

{ c - char we read

op - current operator

i - iterator and index

num - number of operands

isEnd - stop condition }

begin

// Assigning varibles to the original values

i := 1;

op := s[i];

inc(i);

num := 0;

isEnd := false;

// Start of the cycle B1, parsing remaining part

while (i <= length(s)) and (not isEnd) do

begin

inc(i);

c := s[i];

// If operand we add them to the list

if (c = 't') or (c = 'f') then

begin

setlength(boolsArr, num + 1);

boolsArr[num] := c = 't';

inc(num);

end

// If operator we recursive call the function

else if (c = '|') or (c = '&') or (c = '!') then

begin

setlength(boolsArr, num + 1);

boolsArr[num] := parseBoolExp(copy(s, i,

length(s) - i));

inc(num);

end

// End of the operation condition

else if c = ')' then

isEnd := true;

end;

// End of the cycle B1

// Calculating result

result := eval(boolsArr, op);

end;

begin

write('Input your boolean expression: ');

readln(inputStr);

cond := parseBoolExp(replace(inputStr));

writeln('Result: ', cond);

readln;

end.

Приложение Б

(обязательное)

Тестовые наборы

Таблица 6 – Тестовые наборы

|  |  |  |
| --- | --- | --- |
| Тест | Исходные данные и ожидаемый результат | Полученный результат |
| 1. | And(Or(False, Not(False)),  True, Not(True))  Ожидается: FALSE |  |
| 2. | True  Ожидается: TRUE |  |
| 3. | Not(True)  Ожидается: FALSE |  |
| 4. | Or(True, False)  Ожидается: TRUE |  |
| 5. | And(True, False)    Ожидается: FALSE |  |
| 6. | Not(Not(Not(True)))  Ожидается: FALSE |  |
| 7. | And(And(True, False), False)  Ожидается: FALSE |  |
| 8. | Or(True, Or(True, Not(False), True, False), Not(True))  Ожидается: TRUE |  |

Приложение В

**(обязательное)**

**Выполнение действий на рекурсивном спуске и возврате**

Выражение: And(Or(False, Not(False)), True, Not(True)).

Эквивалентное выражение: &(|(f, !(f)), t, !(t)).

Or(False, Not(False)), True, Not(True).

Таблица 7 – Выполнение действий на рекурсивном спуске и возврате

|  |  |  |
| --- | --- | --- |
| Уровень рекурсии | Спуск (значение s) | Возврат (результат) |
| 0 | &(|(f, !(f)), t, !(t)) | False |
| 1 | !(f)), t, !(t) | True |
| 2 | |(f, !(f)), t, !(t) | True |
| 3 | !(f)), t, !(t | True |