**МИНОБРНАУКИ РОССИИ**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 №1

Дисциплина: Математическая логика и теория алгоритмов

по теме «Логика высказываний»

Выполнил: ст. группы ВТ-22  
Макаров Даниил Сергеевич

Проверил: Куценко Д.А.

Белгород 2018

Задание к работе

Задание варианта 7

Разработать программу, решающую задачи согласно своему варианту. Программа должна считывать формулу логики высказываний в конъюнктивной нормальной форме.

2. Программа должна доказать является ли данная формула общезначимой.

5. Программа должна отыскивать все интерпретации, на которых введенная формула принимает ложное значение.

Модуль r\_c

**Заголовочный файл r\_c.h**

#ifndef REV\_NOTE

#define REV\_NOTE

#include "stack.h"

#include <stdio.h>

#include <stdlib.h>

enum operatons{INV='!',MLT='\*',SUM='+',OBR='(',CBR=')'};

int str\_lenght(char\* str);

char strsearch(char\* str,char c);

int revers\_note(char\* str,char\* result);

#endif

**Исходный код r\_c.с**

#include "r\_n.h"

#include <string.h>

char strsearch(char\* str,char c){

int i=0;

int size=strlen(str);

while((str[i]!=c)&(i<size)) i++;

return (i<size);

}

int revers\_note(char\* str,char\* result){

char arr[30];

stack buff;

char t,f1;

stack \*buffer=&buff;

stack\_init(buffer);

int i=0,j=0,k=0;

while(str[i]!=0){

switch(str[i]){

case SUM :

if(stack\_top(buffer)==MLT) {

result[j]=stack\_pop(buffer);

j++;

stack\_push(buffer,SUM);}

else {

stack\_push(buffer,SUM);}

break;

case MLT :

stack\_push(buffer,MLT);

break;

case INV :

stack\_push(buffer,INV);

f1=1;

break;

case OBR :

stack\_push(buffer,OBR);

break;

case CBR :

while((t=stack\_pop(buffer))!=OBR){

if(t!=OBR){

result[j]=t;

j++;}}

break;

default:

if (!(strsearch(arr,str[i]))){

arr[k]=str[i];

k++;

};

result[j]=str[i];

j++;

if(stack\_top(buffer)==INV){

result[j]=stack\_pop(buffer);

j++;

f1=0;}

}

i++;

}

while(!(stack\_empty(buffer))){

result[j]=stack\_pop(buffer);

j++;

};

result[j]='\0';

return k;

}

**Заголовочный файл stack.h**

#ifndef STACK

#define STACK

#define STACK\_SIZE 100

#define BASE\_T char

int stack\_error;

extern const int STACK\_OVERFLOW,STACK\_EMPTY;

typedef struct stk {

BASE\_T elem[STACK\_SIZE];

int top;

}stack;

void stack\_init(stack \*stk);

void stack\_push(stack \*stk, BASE\_T f);

BASE\_T stack\_pop(stack \*stk);

BASE\_T stack\_top(stack \*stk);

int stack\_empty(stack \*stk);

#endif

**Исходный код stack.c**

#include "stack.h"

#include <stdio.h>

#include <stdlib.h>

const int STACK\_OVERFLOW=100;

const int STACK\_EMPTY=101;

void stack\_init(stack \*stk) {

stk->top = 0;

}

void stack\_push(stack \*stk, BASE\_T f) {

if(stk->top < STACK\_SIZE) {

stk->elem[stk->top] = f;

stk->top++;

} else{

stack\_error=STACK\_OVERFLOW;

printf("STACK OVERFLOW");

exit(STACK\_OVERFLOW);}

}

BASE\_T stack\_pop(stack \*stk) {

BASE\_T elem;

if((stk->top) > 0) {

stk->top--;

elem = stk->elem[stk->top];

return(elem);

}else {

stack\_error=STACK\_EMPTY;

printf("STACK IS EMPTY");

exit(STACK\_EMPTY);}

}

BASE\_T stack\_top(stack \*stk) {

char t = stack\_pop(stk);

stack\_push(stk,t);

return(t);}

int stack\_empty(stack \*stk) {

if((stk->top) == 0) return(1);

else return(0);

}

**Заголовочный файл task.h**

#ifndef TASK

#define TASK

int expn(int e);

void fill\_table(char \*\*table, int column, int row, int count);

char\*\* table\_Init(int size);

void create\_calcStr(char\* str,char\*\* table, int line,char\* r);

void task\_2(char\* str,int size);

void task\_5(char\* str,int size);

char rn\_Calc(char\* str);

#endif

**Исходный код task.h**

#include "stack.h"

#include "r\_n.h"

#include "task.h"

#include <string.h>

#include <stdlib.h>

int expn(int e){

return 2<<e;

}

void create\_calcStr(char\* str,char\*\* table, int line,char \*r){

int j=0;

while (j<strlen(str)){

if (str[j]>='A' && str[j]<='Z'){

r[j]=table[line][str[j]-'A']+'0';

}

else r[j]=str[j];

j++;

}

r[j]='\0';

}

void fill\_table(char \*\*table, int column, int row, int count){

if (count == 1) return;

for (int i = 0; i < (count/2); i++){

table[i+row][column] = 0;

}

fill\_table (table, column + 1, row, count/2);

for (int i = (count/2); i < count; i++){

table[i+row][column] = 1;

}

fill\_table (table, column + 1, row+count/2, count/2);

}

char\*\* table\_Init(int size){

char\*\* r;

int m=1<<size-1;

r=(char\*\*)malloc(sizeof(char\*)\*m);

for(int i=0;i<m;i++){

r[i]=(char\*)malloc(sizeof(char)\*(size+1));

}

return r;

};

char rn\_Calc(char\* str){

stack buff;

stack \*buffer=&buff;

int i=0;

char t1,t2;

stack\_init(buffer);

while(str[i]!=0){

switch(str[i]){

case SUM:

t1=stack\_pop(buffer);

t2=stack\_pop(buffer);

stack\_push(buffer,t1||t2);

break;

case MLT:

t1=stack\_pop(buffer);

t2=stack\_pop(buffer);

stack\_push(buffer,t1&&t2);

break;

case INV :

t1=stack\_pop(buffer);

stack\_push(buffer,!t1);

break;

default :

stack\_push(buffer,str[i]-'0');

}

i++;

}

return stack\_pop(buffer);

}

void task\_2(char\* str,int size){

char\*\* table;

char t[200];

char f=1;

int s=1<<size;

table=table\_Init(size+1);

fill\_table(table,0,0,1<<size);

for(int i=0;i<(1<<size);i++){

create\_calcStr(str,table,i,t);

table[i][size+1]=rn\_Calc(t);

if(table[i][size+1]==0){

f=0;

}

}

if (f==1){

printf("function is validity\n");

}

else{

printf("function isn't validity\n");

}

}

void task\_5(char\* str,int size){

char\*\* table;

char t[200];

int s=1<<size;

table=table\_Init(size+1);

fill\_table(table,0,0,1<<size);

for(int i=0;i<(1<<size);i++){

create\_calcStr(str,table,i,t);

table[i][size+1]=rn\_Calc(t);

if(table[i][size+1]==0){

for(int j=0;j<size;j++){

printf("%d ",table[i][j]);

}

printf("\n");

}

}

}

**Исходный код main.c**

#include <stdio.h>

#include <stdlib.h>

#include "stack.h"

#include "r\_n.h"

#include "task.h"

int main(){

char i\_arr[100];

char\* result;

int count;

result=(char\*)malloc(sizeof(char)\*100);

scanf("%s",i\_arr);

count=revers\_note(i\_arr,result);

printf("\n%s\n",result);

printf("task 2\n");

task\_2(result,count);

printf("task 5\n");

task\_5(result,count);

}