**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

Расчетно-графическое задание

Дисциплина: Математическая логика и теория алгоритмов

по теме «Построение релейно-контактных схем по формулам»

Выполнил: ст. группы ВТ-22  
Макаров Даниил Сергеевич

Проверил: Куценко Д.А.

Белгород 2018Заголовочный файл

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#define BLOCK\_COUNT 50 //количество блоков

#define BLOCK\_SIZE 50 // размер блока в пикселях

#define BORDER\_SIZE 20 // отступы от края

void svg\_init(FILE \*file);

void svg\_debugmap(FILE \*file);

void svg\_close(FILE \*file);

void svg\_start\_icon(FILE \*file,unsigned x,unsigned y); //отрисовывает знак начала цепи в блоке x:y

void svg\_end\_icon(FILE \*file,unsigned x,unsigned y); //отрисовывает знак конца цепи в блоке x:y

void svg\_mid\_line(FILE \*file,unsigned x,unsigned y); //отрисовывает линию по середине блока

void svg\_right\_up\_edge\_semiline(FILE \*file,unsigned x,unsigned y); //отрисовывает верхную половину линию по правому краю блока

void svg\_right\_down\_edge\_semiline(FILE \*file,unsigned x,unsigned y); //отрисовывает нижнюю половину линию по правому краю блока

void svg\_left\_up\_edge\_semiline(FILE \*file,unsigned x,unsigned y); //отрисовывает верхную половину линию по левому краю блока

void svg\_left\_down\_edge\_semiline(FILE \*file,unsigned x,unsigned y); //отрисовывает нижнюю половину линию по левому краю блока

void svg\_right\_edge\_line(FILE \*file,unsigned x,unsigned y);//отрисовывает линию по правому краю блока

void svg\_left\_edge\_line(FILE \*file,unsigned x,unsigned y);//отрисовывает линию по левому краю блока

void svg\_block\_icon(FILE \*file,char \*str, unsigned x,unsigned y); //отрисовывает блок реле

void svg\_build\_brackets(FILE \*file,unsigned height,unsigned width);

void svg\_build\_graph(FILE \*file,char \*str,unsigned size);

void svg\_print\_text(FILE \*file,char \*str,unsigned x,unsigned y);

char\* str\_prep\_to\_out(char\* in);

unsigned count\_of\_dis(char\* str,unsigned size);

unsigned max\_of\_con(char\* str,unsigned size);

Исходный код модуля svg\_graph.c

#include "svg\_graph.h"

void svg\_init(FILE \*file){

fprintf(file,"<?xml version=%c1.0%c encoding=%cUTF-8%c standalone=%cno%c?>\n",34,34,34,34,34,34);

fprintf(file,"<svg version = \"1.1\" baseProfile = \"full\" xmlns=\"http://www.w3.org/2000/svg\" xmlns:xlink = \"http://www.w3.org/1999/xlink\" xmlns:ev=\"http://www.w3.org/2001/xml-events\" width=\"%d\" height=\"%d\">\n",BLOCK\_SIZE\*BLOCK\_COUNT,BLOCK\_SIZE\*BLOCK\_COUNT);

}

void svg\_debugmap(FILE \*file){

int x,y;

fprintf(file,"<!-- DEBUG SHEET -->");

for(x=1;x<BLOCK\_COUNT;x++){

for(y=1;y<BLOCK\_COUNT;y++){

fprintf(file," <rect x=\"%d\" y=\"%d\" width=\"%d\" height=\"%d\" fill=\"none\" stroke=\"black\" stroke-width=\"%d\"/>\n"

,BORDER\_SIZE+x\*BLOCK\_SIZE,BORDER\_SIZE+y\*BLOCK\_SIZE,BLOCK\_SIZE,BLOCK\_SIZE,BLOCK\_SIZE/20);

}

}

}

void svg\_start\_icon(FILE \*file,unsigned x,unsigned y){

fprintf(file," <line x1=\"%d\" y1=\"%d\" x2=\"%d\" y2=\"%d\" stroke=\"black\" stroke-width=\"%d\"/>\n",BORDER\_SIZE+x\*BLOCK\_SIZE+BLOCK\_SIZE/2,BORDER\_SIZE+y\*BLOCK\_SIZE+BLOCK\_SIZE/2,BORDER\_SIZE+x\*BLOCK\_SIZE+BLOCK\_SIZE,BORDER\_SIZE+y\*BLOCK\_SIZE+BLOCK\_SIZE/2,BLOCK\_SIZE/20);

fprintf(file," <circle cx=\"%d\" cy=\"%d\" r=\"%d\" fill=\"white\" stroke=\"black\" stroke-width=\"%d\"/>\n",BORDER\_SIZE+x\*BLOCK\_SIZE+BLOCK\_SIZE/2,BORDER\_SIZE+y\*BLOCK\_SIZE+BLOCK\_SIZE/2,BLOCK\_SIZE/5,BLOCK\_SIZE/20);

}

void svg\_end\_icon(FILE \*file,unsigned x,unsigned y){

fprintf(file," <line x1=\"%d\" y1=\"%d\" x2=\"%d\" y2=\"%d\" stroke=\"black\" stroke-width=\"%d\"/>\n",BORDER\_SIZE+x\*BLOCK\_SIZE,BORDER\_SIZE+y\*BLOCK\_SIZE+BLOCK\_SIZE/2,BORDER\_SIZE+x\*BLOCK\_SIZE+BLOCK\_SIZE/2,BORDER\_SIZE+y\*BLOCK\_SIZE+BLOCK\_SIZE/2,BLOCK\_SIZE/20);

fprintf(file," <circle cx=\"%d\" cy=\"%d\" r=\"%d\" fill=\"white\" stroke=\"black\" stroke-width=\"%d\"/>\n",BORDER\_SIZE+x\*BLOCK\_SIZE+BLOCK\_SIZE/2,BORDER\_SIZE+y\*BLOCK\_SIZE+BLOCK\_SIZE/2,BLOCK\_SIZE/5,BLOCK\_SIZE/20);

}

void svg\_mid\_line(FILE \*file,unsigned x,unsigned y){

fprintf(file," <line x1=\"%d\" y1=\"%d\" x2=\"%d\" y2=\"%d\" stroke=\"black\" stroke-width=\"%d\"/>\n",BORDER\_SIZE+x\*BLOCK\_SIZE,BORDER\_SIZE+y\*BLOCK\_SIZE+BLOCK\_SIZE/2,BORDER\_SIZE+x\*BLOCK\_SIZE+BLOCK\_SIZE,BORDER\_SIZE+y\*BLOCK\_SIZE+BLOCK\_SIZE/2,BLOCK\_SIZE/20);

}

void svg\_right\_edge\_line(FILE \*file,unsigned x,unsigned y){

fprintf(file," <line x1=\"%d\" y1=\"%d\" x2=\"%d\" y2=\"%d\" stroke=\"black\" stroke-width=\"%d\"/>\n",BORDER\_SIZE+(x+1)\*BLOCK\_SIZE,BORDER\_SIZE+y\*BLOCK\_SIZE,BORDER\_SIZE+(x+1)\*BLOCK\_SIZE,BORDER\_SIZE+(y+1)\*BLOCK\_SIZE,BLOCK\_SIZE/20);

}

void svg\_left\_edge\_line(FILE \*file,unsigned x,unsigned y){

fprintf(file," <line x1=\"%d\" y1=\"%d\" x2=\"%d\" y2=\"%d\" stroke=\"black\" stroke-width=\"%d\"/>\n",BORDER\_SIZE+x\*BLOCK\_SIZE,BORDER\_SIZE+y\*BLOCK\_SIZE,BORDER\_SIZE+x\*BLOCK\_SIZE,BORDER\_SIZE+(y+1)\*BLOCK\_SIZE,BLOCK\_SIZE/20);

}

void svg\_right\_up\_edge\_semiline(FILE \*file,unsigned x,unsigned y){

fprintf(file," <line x1=\"%d\" y1=\"%d\" x2=\"%d\" y2=\"%d\" stroke=\"black\" stroke-width=\"%d\"/>\n",BORDER\_SIZE+(x+1)\*BLOCK\_SIZE,BORDER\_SIZE+y\*BLOCK\_SIZE+BLOCK\_SIZE/2,BORDER\_SIZE+(x+1)\*BLOCK\_SIZE,BORDER\_SIZE+(y+1)\*BLOCK\_SIZE,BLOCK\_SIZE/20);

}

void svg\_right\_down\_edge\_semiline(FILE \*file,unsigned x,unsigned y){

fprintf(file," <line x1=\"%d\" y1=\"%d\" x2=\"%d\" y2=\"%d\" stroke=\"black\" stroke-width=\"%d\"/>\n",BORDER\_SIZE+(x+1)\*BLOCK\_SIZE,BORDER\_SIZE+(y)\*BLOCK\_SIZE,BORDER\_SIZE+(x+1)\*BLOCK\_SIZE,BORDER\_SIZE+(y)\*BLOCK\_SIZE+BLOCK\_SIZE/2,BLOCK\_SIZE/20);

}

void svg\_left\_up\_edge\_semiline(FILE \*file,unsigned x,unsigned y){

fprintf(file," <line x1=\"%d\" y1=\"%d\" x2=\"%d\" y2=\"%d\" stroke=\"black\" stroke-width=\"%d\"/>\n",BORDER\_SIZE+x\*BLOCK\_SIZE,BORDER\_SIZE+y\*BLOCK\_SIZE+BLOCK\_SIZE/2,BORDER\_SIZE+x\*BLOCK\_SIZE,BORDER\_SIZE+(y+1)\*BLOCK\_SIZE,BLOCK\_SIZE/20);

}

void svg\_left\_down\_edge\_semiline(FILE \*file,unsigned x,unsigned y){

fprintf(file," <line x1=\"%d\" y1=\"%d\" x2=\"%d\" y2=\"%d\" stroke=\"black\" stroke-width=\"%d\"/>\n",BORDER\_SIZE+x\*BLOCK\_SIZE,BORDER\_SIZE+(y)\*BLOCK\_SIZE,BORDER\_SIZE+x\*BLOCK\_SIZE,BORDER\_SIZE+(y)\*BLOCK\_SIZE+BLOCK\_SIZE/2,BLOCK\_SIZE/20);

}

void svg\_block\_icon(FILE \*file,char \*str, unsigned x,unsigned y){

fprintf(file," <rect x=\"%d\" y=\"%d\" width=\"%d\" height=\"%d\" fill=\"none\" stroke=\"black\" stroke-width=\"%d\"/>\n",BORDER\_SIZE+x\*BLOCK\_SIZE,BORDER\_SIZE+y\*BLOCK\_SIZE,BLOCK\_SIZE,BLOCK\_SIZE,BLOCK\_SIZE/20);

fprintf(file," <text x=\"%d\" y=\"%d\" text-anchor=\"middle\" alignment-baseline=\"middle\" style=\"font-size:%dpx\">%s</text>\n",BORDER\_SIZE+x\*BLOCK\_SIZE+BLOCK\_SIZE/2,BORDER\_SIZE+y\*BLOCK\_SIZE+BLOCK\_SIZE/2,BLOCK\_SIZE/2,str);

}

void svg\_print\_text(FILE \*file,char \*str,unsigned x,unsigned y){

fprintf(file," <text x=\"%d\" y=\"%d\" text-anchor=\"middle\" alignment-baseline=\"middle\" style=\"font-size:%dpx\">%s</text>\n",BORDER\_SIZE+(x)\*BLOCK\_SIZE+BLOCK\_SIZE/2,BORDER\_SIZE+y\*BLOCK\_SIZE+BLOCK\_SIZE/2,BLOCK\_SIZE/2,str);

}

void svg\_close(FILE \*file){

fprintf(file,"</svg>\n");

fclose(file);

}

void svg\_build\_brackets(FILE \*file,unsigned width,unsigned height){

if(height==1){

svg\_start\_icon(file,1,1);

svg\_end\_icon(file,width\*2+3,1);

}

else{

svg\_start\_icon(file,1,height);

svg\_end\_icon(file,2\*width+5,height);

svg\_right\_up\_edge\_semiline(file,2,1);

svg\_left\_up\_edge\_semiline(file,2\*width+4,1);

svg\_mid\_line(file,2,height);

svg\_mid\_line(file,2\*width+4,height);

svg\_right\_down\_edge\_semiline(file,2,2\*height-1);

svg\_left\_down\_edge\_semiline(file,2\*width+4,2\*height-1);

for(int i=2;i<2\*height-1;i++){

svg\_right\_edge\_line(file,2,i);

svg\_left\_edge\_line(file,2\*width+4,i);

}

}

}

unsigned count\_of\_dis(char\* str,unsigned size){

unsigned counter=0;

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

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

counter++;

}

}

return counter;

}

unsigned max\_of\_con(char\* str,unsigned size){

unsigned current\_count=0,max=0;

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

switch (str[i]) {

case '(' :

current\_count=0;

break;

case '\*' :

current\_count++;

break;

case ')':

if(current\_count>max){

max=current\_count;

}

break;

}

}

return max;

}

void svg\_build\_graph(FILE \*file,char \*str,unsigned size){

//временная строка для вывода переменной в блок j-индекс для позиционирования в этой строке

int j=0,k=0;

char \*buffer=(char\*)malloc(sizeof(char)\*50);

//подсчет кол-ва дизъюнкций

unsigned height=count\_of\_dis(str,size);

//поиск максимального кол-ва конъюнкций в дизъюнктах

unsigned width=max\_of\_con(str,size);

//построение левых и правых вертикальных соединительных линий

svg\_build\_brackets(file,width+1,height+1);

unsigned x=0,y=1;

char block\_state=2;

//если высота = 1 то просто строим линейную схему

if(height==0){

for(int i=2;i<(width+1)\*2+3;i+=2){

svg\_mid\_line(file,i,y);

}

x=3;

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

if((str[i]=='\*')||(str[i]==')')){

buffer[j]='\0';

switch (block\_state) {

case 2:

svg\_block\_icon(file,buffer,x,y);

break;

case 1:

svg\_mid\_line(file,x,y);

break;

}

block\_state=2;

x+=2;

j=0;

}

else{

switch (str[i]) {

case '0':

block\_state=0;

break;

case '1':

block\_state=1;

break;

case '!':

buffer[j]='&'; j++; buffer[j]='#'; j++; buffer[j]='1'; j++; buffer[j]='7'; j++; buffer[j]='2'; j++; buffer[j]=';'; j++;

break;

default:

buffer[j]=str[i];

j++;

}

}

}

svg\_print\_text(file,str\_prep\_to\_out(str),width+3,height+3);

}

//иначе строим линейную схему для каждого дизъюнкта выравнивая кол-во элементов

else{

unsigned count\_var=0;

y=0;

int start\_of\_bracket,end\_of\_bracket;

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

switch (str[i]) {

case '(' :

count\_var=0;

start\_of\_bracket=i+1;

y++;

break;

case '\*' :

count\_var++;

break;

case ')' :

end\_of\_bracket=i;

x=3;

//начальные горизонтальные линии

for(int i=0;i<width-count\_var+1;i++){

svg\_mid\_line(file,x+i,y);

svg\_mid\_line(file,2\*width+5-i,y);

}

x=4+width-count\_var;

k=start\_of\_bracket;

//построение блоков реле

for(int i=0;i<=2\*count\_var;i+=2){

j=0;

//заполнение блоков текстом

while((str[k]!='\*')&&(str[k]!=')')){

switch (str[k]) {

case '0':

block\_state=0;

break;

case '1':

block\_state=1;

break;

case '!':

buffer[j]='&'; j++; buffer[j]='#'; j++; buffer[j]='1'; j++; buffer[j]='7'; j++; buffer[j]='2'; j++; buffer[j]=';'; j++;

break;

default:

buffer[j]=str[k];

j++;

break;

}

k++;

}

buffer[j]='\0';

k++;

switch (block\_state) {

case 2:

svg\_block\_icon(file,buffer,x+i,y);

break;

case 1:

svg\_mid\_line(file,x+i,y);

break;

}

//k++;

block\_state=2;

//}

//если проверяет состояние блока и отрисовываем его в соответствии с состоянием

}

//построение соединительных горизонтальных линий

for(int i=1;i<=2\*count\_var;i+=2){

svg\_mid\_line(file,x+i,y);

}

//переход на следующую строку блока

y++;

break;

}

}

svg\_print\_text(file,str\_prep\_to\_out(str),2\*width-width/2+3,2\*height+3);

}

}

char\* str\_prep\_to\_out(char\* in){

unsigned sizeof\_str=strlen(in);

char\* out=(char\*)malloc(sizeof\_str\*sizeof(char));

int j=0;

for(int i=0;i<sizeof\_str;i++){

switch (in[i]) {

case '!':

out[j]='&'; j++; out[j]='#'; j++; out[j]='1'; j++; out[j]='7'; j++; out[j]='2'; j++; out[j]=';'; j++;

break;

case '\*':

out[j]='&'; j++; out[j]='#'; j++; out[j]='8'; j++; out[j]='7'; j++; out[j]='4'; j++; out[j]='3'; j++; out[j]=';'; j++;

break;

case '+':

out[j]='&'; j++; out[j]='#'; j++; out[j]='8'; j++; out[j]='7'; j++; out[j]='4'; j++; out[j]='4'; j++; out[j]=';'; j++;

break;

default:

out[j]=in[i];

j++;

}

}

out[j]='\0';

j++;

return out;

}



