

**Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ *Робототехники и комплексной автоматизации*

КАФЕДРА *Системы автоматизированного проектирования (РК-6)*

# ОТЧЕТ О ВЫПОЛНЕНИИ РУБЕЖНОГО КОНТРОЛЯ

Студент Журавлев Николай

Группа РК6-42Б

Тип задания Рубежный контроль №1

Студент

**Журавлев Н.В.**

*подпись, дата фамилия, и.о.*

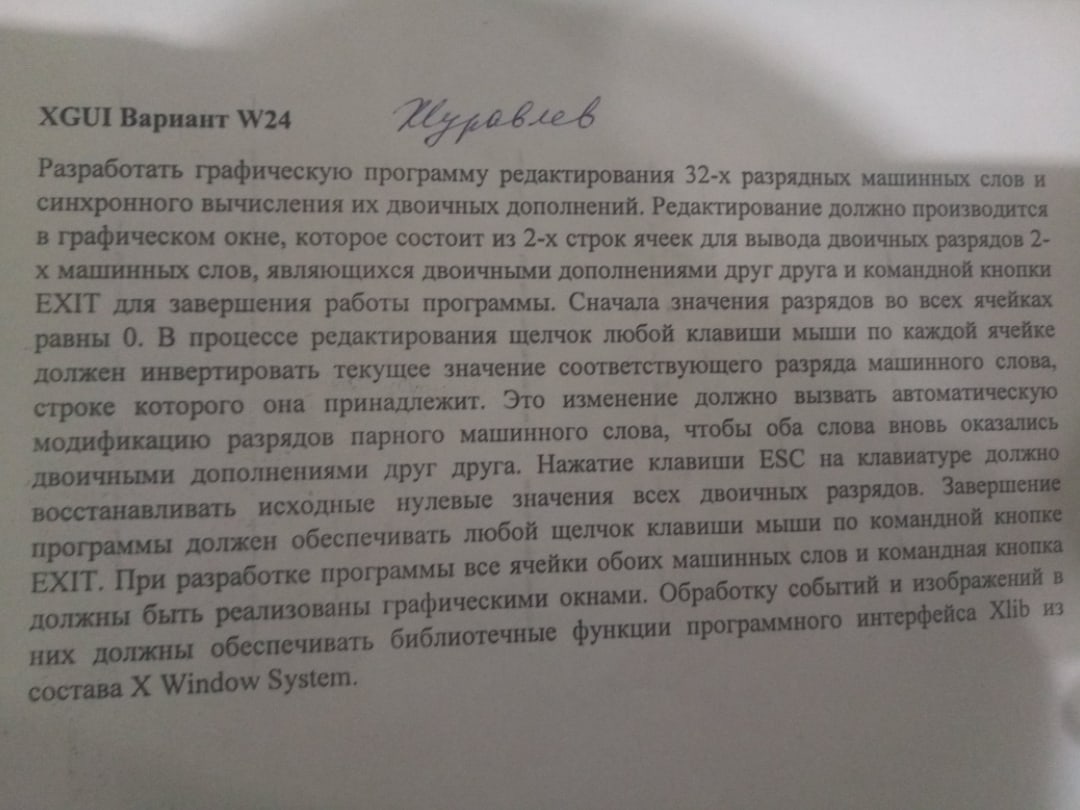
Преподаватель

**Родионов С.В.**

*подпись, дата фамилия, и.о.*

*Москва, 2021*

**Текст задания**



**Описание алгоритма**

В исходном коде программы 14 функция, а также основная функция main. В функциях xalloc() и dealloc() происходит выделение и очистка памяти соответственно. В assoc() происходит запись значений в поле структуры. В функции rebox() происходит перерисовка выявленных ячеек. Функция xcustom() создает и отрисовывает подокна с векторами и [0, 1] – матрицы их внешнего произведения. Функция kbdrive() вызывает при нажатии Esc возвращает исходные данные. Функция dispatch() проверяет на нажатие на ячейку, нажатие мышью, а также нажатие клавиш. Основная функция main() создает основное графическое окно, затем вызывает функцию xcustom(), создавая подокна, далее вызывает функцию dispatch(), которая ждет некоторого события. Исходный текст графического модуля подключает библиотеку Xlib. Используется структура, а также внешние статические переменные для размеров подокон, начального положения в окне и т.д. Также происходит подсчет геометрии ячеек, настройка окна, графического контекста и т.д.

**Текст программы**

**main.c**

#include <X11/Xlib.h>

#include <X11/Xutil.h>

#include <X11/keysym.h>

#include <X11/keysymdef.h>

#include <stdio.h>

#include <stdlib.h>

static Display \* dpy; /\* X Display \*/

static GC gc[2]; /\* Graphic Context \*/

static Window root; /\* Main root window \*/

static Window \*\* box; /\* box (NYxNX) window array address \*/

static Window \* row; /\* NY row window box containe array \*/

static unsigned \*\* pos; /\* Marker position NYx2 array address \*/

static int X0 = 0; /\* main root wintow x-location \*/

static int Y0 = 0; /\* main root wintow y-location \*/

static unsigned NX = 32; /\* Default Boxes' number in each row \*/

static unsigned NY = 3; /\* Default Row's number in game desk \*/

static XRectangle cell; /\* Box cell \*/

static char \* mark[] = {

"0",

"1",

"EXIT"

};

static char\*\* last\_move;

int relink(void \*\* p, void \* r, void \*\* b) {

pos = (unsigned \*\* ) p;

row = (Window \* ) r;

box = (Window \*\* ) b;

return (0);

}

int rebox(XEvent \* ev) {

unsigned int i, j;

for (i = 0; i < NX ; i++) {

if (ev->xexpose.window == box[0][i]) {

XDrawString(dpy, box[0][i], gc[0], cell.x, cell.y, mark[0], 1);

last\_move[0][i] = \*mark[0];

}

}

for (j = 0; j < NX; j++) {

if (ev->xexpose.window == box[1][j]) {

XDrawString(dpy, box[1][j], gc[0], cell.x, cell.y, mark[1], 1);

last\_move[1][j] = \*mark[1];

}

}

XDrawString(dpy, box[2][0], gc[0], cell.x \* 1.65 \* NX, cell.y, mark[2], 4);

return (0);

}

int repos(unsigned i, unsigned j, int m) {

XDrawString(dpy, box[i][j], gc[1], cell.x, cell.y, mark[m], 1);

XDrawString(dpy, box[i][j], gc[0], cell.x, cell.y, mark[!m], 1);

XFlush(dpy);

last\_move[i][j] = \*mark[!m];

return (i);

}

int reset(XEvent \* ev) {

int m;

int i, j;

for (i = 0; i < NY; i++)

if (ev -> xbutton.window == row[i])

break;

if (i == NY) return 0;

for (j = 0; j < NX; j++)

if (ev -> xbutton.subwindow == box[i][j])

break;

if (j == NX) return 0;

if (i == 2 && j == 0) {

return 1;

}

switch (last\_move[i][j]) {

case '1': m = 1; break;

case '0': m = 0; break;

default: return 0;

}

if (repos(i, j, m) == NY)

return (NY);

if (i == 0) {

++i;

} else {

--i;

}

if (repos(i, j, !m) == NY)

return (NY);

if (ev -> xbutton.state & ControlMask)

return 0;

return 0;

}

int xcustom() {

static unsigned XSP = 4; /\* x-space between 2 boxes in any row \*/

static unsigned YSP = 4; /\* y-space between 2 rows \*/

static unsigned BW = 32; /\* Box width \*/

static unsigned BH = 32; /\* Box height \*/

int x, y; /\* windows' location \*/

unsigned w, h; /\* windows' size \*/

int dx, dy; /\* interwindow distance \*/

int src; /\* screen number \*/

int depth; /\* screen depth \*/

XSetWindowAttributes attr; /\* created Window attributes \*/

XSizeHints hint; /\* WM connection structure \*/

XFontStruct \* fn; /\* Font parameters structure \*/

char \* fontname = "9x15"; /\* default font name \*/

int i, j;

src = DefaultScreen(dpy);

depth = DefaultDepth(dpy, src);

gc[0] = DefaultGC(dpy, src);

if ((fn = XLoadQueryFont(dpy, fontname)) == NULL)

return (puts("Incorrect FontStruct id"));

XSetFont(dpy, gc[0], fn -> fid);

cell.width = fn -> max\_bounds.width;

cell.height = fn -> max\_bounds.ascent + fn -> max\_bounds.descent;

cell.x = (BW - fn -> max\_bounds.width) / 2;

cell.y = BH / 2 + (fn -> max\_bounds.ascent - fn -> max\_bounds.descent) / 2;

w = NX \* BW + (NX + 1) \* XSP + 2 \* NX + 2;

h = NY \* BH + (NY + 1) \* YSP + 4 \* NY;

attr.override\_redirect = False;

attr.background\_pixel = BlackPixel(dpy, src);

x = X0;

y = Y0;

root = XCreateWindow(dpy, DefaultRootWindow(dpy), x, y, w, h,

1, depth, InputOutput, CopyFromParent,

(CWOverrideRedirect | CWBackPixel), & attr);

hint.flags = (PMinSize | PMaxSize | PPosition);

hint.min\_width = hint.max\_width = w;

hint.min\_height = hint.max\_height = h;

hint.x = x;

hint.y = y;

XSetNormalHints(dpy, root, & hint);

attr.override\_redirect = True;

attr.background\_pixel = BlackPixel(dpy, src);

attr.event\_mask = (ButtonPressMask | KeyPressMask);

w = NX \* BW + (NX - 1) \* XSP + 2 \* NX + 2;

h = BH + 2 + 2;

x = XSP;

dy = h + YSP;

for (i = 0, y = YSP; i < NY; i++, y += dy) {

row[i] = XCreateWindow(dpy, root, x, y, w, h, 0, depth,

InputOutput, CopyFromParent,

(CWOverrideRedirect | CWBackPixel | CWEventMask), &attr);

}

attr.override\_redirect = True;

attr.background\_pixel = 0x00FF00; /\* WhitePixel(dpy, src); \*/

attr.event\_mask = (KeyPressMask | ExposureMask);

w = BW;

h = BH;

dx = w + XSP + 2;

for (i = 0, y = 0; i < NY; i++)

for (j = 0, x = 0; j < NX; j++, x += dx) {

if (i != 2) {

box[i][j] = XCreateWindow(dpy, row[i], x, y, w, h, 1,

depth, InputOutput, CopyFromParent,

(CWOverrideRedirect | CWBackPixel | CWEventMask), &attr);

} else {

if (j == 0) {

attr.background\_pixel = 0xFF0000;

box[i][j] = XCreateWindow(dpy, row[i], x , y , w \* 37.75, h, 1,

depth, InputOutput, CopyFromParent,

(CWOverrideRedirect | CWBackPixel | CWEventMask), &attr);

}

}

}

XMapWindow(dpy, root);

XMapSubwindows(dpy, root);

for (i = 0; i < NY; i++)

XMapSubwindows(dpy, row[i]);

XStoreName(dpy, root, "RK1");

gc[1] = XCreateGC(dpy, root, 0, 0);

XCopyGC(dpy, gc[0], GCFont, gc[1]);

XSetForeground(dpy, gc[1], 0x00FF00);

return (0);

}

int kbdrive(XEvent \* ev) {

KeySym sym;

XLookupString((XKeyEvent \* ) ev, NULL, 0, & sym, NULL);

switch (sym) {

case XK\_Escape:

restore();

break;

default:

break;

}

return (0);

}

int dispatch() {

XEvent event;

int flag = 0;

while (flag == 0) {

XNextEvent(dpy, & event);

switch (event.type) {

case Expose:

rebox( & event);

break;

case ButtonPress:

flag = reset( & event);

break;

case KeyPress:

flag = kbdrive( & event);;

break;

default:

break;

}

}

return (0);

}

int main(int argc, char \* argv[]) {

alloc(NX, NY);

last\_move = (char\*)malloc(NY \* sizeof(char));

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

last\_move[i] = (char\*)malloc(NX \* sizeof(char));

}

dpy = XOpenDisplay(NULL);

if (xcustom() > 0)

return (1);

dispatch();

XDestroySubwindows(dpy, root);

XDestroyWindow(dpy, root);

XCloseDisplay(dpy);

dealloc(pos, box, row);

return (0);

}

**xpat1.c**

#include <stdlib.h>

#include <stdio.h>

static unsigned \*\* pos; /\* Marker position (NYx2) array address \*/

static unsigned NX; /\* Default Boxes' number in each row \*/

static unsigned NY; /\* Default Row's number in game desk \*/

int alloc(unsigned \_nx, unsigned \_ny) {

void \*\* p; /\* position array pointer \*/

void \* r; /\* row array pointer \*/

void \*\* b; /\* box array pointer \*/

int i; /\* row & position array index \*/

NX = \_nx;

NY = \_ny;

p = calloc(NY, sizeof(unsigned \* ));

r = calloc(NY, sizeof(unsigned long));

b = calloc(NY, sizeof(void \* ));

for (i = 0; i < NY; i++) {

b[i] = calloc(NX, sizeof(unsigned long));

p[i] = calloc(2, sizeof(unsigned));

}

relink(p, r, b);

for (i = 0, pos = (unsigned \*\* ) p; i < NY; i++) {

pos[i][0] = 0;

pos[i][1] = NX - 1;

}

return (0);

}

int dealloc(void \*\* p, void \*\* b, void \* r) {

int i; /\* row index \*/

for (i = 0; i < NY; i++) {

free(b[i]);

free(p[i]);

}

free(p);

free(b);

free(r);

return (0);

}

int restore() {

int i;

for (i = 0; i < NX; i++) {

repos(0, i, 1);

repos(1, i, 0);

}

return (0);

}

int remark(int i, int j, int \* m) {

if (j < pos[i][0])

( \* m) = 0;

if (j > pos[i][1])

( \* m) = 1;

if (j != pos[i][( \* m)])

return (repos(i, j, \* m));

return (NY);

}

int rpat() {

int i; /\* row index \*/

for (i = 0; i < NY; i++)

if (pos[i][0] < (NX - 1 - 1))

break;

return (i);

}

int lpat() {

int i; /\* row index \*/

for (i = 0; i < NY; i++)

if (pos[i][1] > 1)

break;

return (i);

}

int lose(int m) {

int i; /\* row index \*/

int d = (m == 0) ? 1 : -1;

for (i = 0; i < NY; i++)

if ((pos[i][1] - pos[i][0]) > 1)

break;

if (i < NY)

return (repos(i, pos[i][m] + d, m));

i = (m == 0) ? lpat() : rpat();

if (i < NY)

repos(i, pos[i][m] - d, m);

return (i);

}

int nim(int m) {

unsigned b = 8 \* sizeof(unsigned); /\* 1 bit \*/

unsigned d = 0; /\* (X-0) distance \*/

unsigned s = 0; /\* NIMM summa \*/

int i;

for (i = 0; i < NY; i++)

s ^= (pos[i][1] - pos[i][0] - 1);

if (s == 0)

return (NY);

for (; b > 0; b >>= 1)

if (b & s)

break;

for (i = 0; i < NY; i++)

if ((d = (pos[i][1] - pos[i][0] - 1)) & b)

break;

for (; b > 0; b >>= 1)

if (b & s)

d = (d & b) ? (d ^ b) : (d | b);

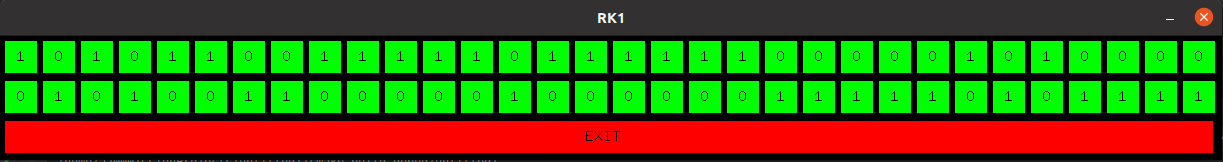
d = (m > 0) ? (d + 1) : -(d + 1);

repos(i, pos[i][(m + 1) % 2] + d, m);

return (i);

}

**Результат работы**



**Литература**

1. Материалы учебного плана по предмету «Программирование графических приложений»
2. Волосатова Т.М., Родионов С. В., Беломойцев Д.Е. «Разработка графических приложений в среде X Window System»
3. Документация по библиотеке Xlib <https://tronche.com/gui/x/xlib/> http://www.asvcorp.ru/tech/linux/xwinprg