Министерство науки и образования РФ

Федеральное государственное автономное образовательное

учреждение высшего профессионального образования

«Санкт-Петербургский государственный электротехнический

университет «ЛЭТИ» им. В. И. Ульянова (Ленина)»

(СПбГЭТУ «ЛЭТИ»)

Факультет компьютерных технологий и информатики

Кафедра вычислительной техники

**Отчет по лабораторной работе №5**

**по дисциплине “Организация ЭВМ и систем”**

**на тему: “Исследование графического режима видеосистемы. Использование прерываний”**

Выполнил: Яловега Н.В. 9308

Принял: Жандаров В.В.

# **Содержание**

[Содержание 2](#__RefHeading___Toc407_355822346)

[1. Задание 3](#__RefHeading___Toc409_355822346)

[2. Краткие теоретические сведения 4](#__RefHeading___Toc411_355822346)

[3. Листинг программы 8](#__RefHeading___Toc413_355822346)

[4. Примеры запуска программ 22](#__RefHeading___Toc415_355822346)

[Вывод 24](#__RefHeading___Toc421_355822346)

# **1. Задание**

Написать программу, использующую аппаратные прерывания и графический режим видеосистемы.

В качестве такой программы была выбрана игра «Крестики-нолики», где необходимо в сетке 3x3 расположить X или O в ряд. Управление – числа на клавиатуре. Используемое прерывание – 16h.

# **2. Краткие теоретические сведения**

Использование графики в языке С++ - это многошаговый процесс. Прежде всего необходимо определить тип видеоадаптера. Затем устанавливается подходящий режим его работы и выполняется инициализация графической системы в выбранном режиме. После этого становятся доступными для исполь­зования функции графической библиотеки graphicx.h для построения основных графических примитивов: отрезков прямых линий, окружностей, эллипсов, прямоугольников, секторов, дуг и т.д., появляется возможность вывода текста с использованием различных шрифтов.

Весь код библиотеки графики разбивается на две части: немобильную, которая зависит от типа видеоадаптера и мобильную.

Немобильная часть представляет собой так называемый .BGI-драйвер (BGI - Borland Graphics Interface).

Основные функции, выполняемые .BGI-драйвером, сводятся к установке и обновлению ряда внешних переменных, которые могут изменяться как функциями системного обработчика прерывания 10h (например, при переключении видеорежима, изменении регистров палитры и т.п.), так и мобильными функциями библиотеки графики.

Графические режимы, поддерживаемые библиотекой графики, задаются символическими константами, описанными в заголовочном файле <graphics.h> в перечислимом типе graphics\_modes.

Функция detectgraph() вызывается автоматически из функции инициализации видеосистемы initgraph(), если последняя вызывается со значением для graphdriver, равным указателю на DETECT.

Защищенное от ошибок построение программы требует использования функции graphresult() после любого обращения к функциям detectgraph() и initgraph(). Далее следует описание функций обработки ошибок, сообщающих внутренние коды ошибок графической библиотеки (graphresult()) или формирующей строку диагностического сообщения (grapherrormsg()).

После того, как проведена инициализация графической системы, может быть установлен другой, не превосходящий максимального, режим видеоадаптера и выбраны цвета для пикселов. Установку режима выполняет функция setgraphmode(). Целая группа функций – getgraphmode(), getmaxmode(), getmodename() , getmoderange() - упрощает работу по определению текущего ус­тановленного режима

После инициализации системы графики и уста­новки нужного видеорежима возможен выбор необходимых цветов пикселов. Возможности по выбору цветов принципиально различны для CGA-, EGA- и VGA-адаптеров, что обусловлено различной логикой построения аппаратных средств.

void setcolor (int color)

Устанавливает цвет, используемый функциями графического вывода в значение, заданное аргументом color. До того момента, пока цвет не установлен, используется максимальный (из палитры) номер цвета. В случае, если color задает недопустимый номер цвета для текущей палитры, текущий цвет остается неизменным.

Окно экрана в графическом режиме, или графическое окно (viewport), - это прямоугольная область экрана, заданная пиксельными координатами левого верхнего и правого нижнего углов.

Графические координаты X и Y измеряются в пикселах экрана относительно координат левого верхнего угла текуще­го окна. Функции графического вывода изменяют эти координаты в соответствии с объемом выведенной на экран инфор­мации. Текущие координаты в окне доступны через функции getx() и gety(). Установку нужных значений координат текущей позиции выполняют функции moveto() и moverel(). Кроме того, некоторые функции графического вывода позволяют задать текущую позицию (см., например, outtextxy()).

int getx (void)

int gety (void)

Базовой функцией любой графической библиотеки является функция вывода в заданные координаты пиксела специфицированного цвета. С++ имеет в своем составе две функции манипуляции отдельными пикселами экрана: getpixel() - для определения кода цвета пиксела и putpixel () - для вывода пиксела текущим цветом.

unsigned getpixel( int x, int у)

Определяет, лежит ли пиксел с координатами (х, у) в текущем графическом окне, и, если лежит, возвращает код цвета этого пиксела. В противном случае возвращается 0.

void putpixel(int x, int у, int pixelcolor)

Целая группа функций библиотеки графики предназначена для вывода отрезков прямых линий. Далее приводится спецификация этих функций. Напомним, что на вывод отрезков прямых линий влияют режим вывода линии и стиль линии.

Выводимые отрезки прямых линий не пересекают границ текущего окна, если при описании окна включен режим "усечения" (clipping).

void line( int x1, int y1, int x2, int y2)

Выводит отрезок прямой линии между двумя явно специфицированными точками (x1, y1) и (х2, у2), используя текущие цвет, стиль, толщину и режим вывода линии. Координаты (x1, y1) и (х2, у2) задаются относительно левого верхнего угла текущего графического окна. Функция не изменяет текущую позицию.

Подавляющее большинство программ выполняют ввод информации с клавиатуры. Ввод информации в компьютер может быть выполнен на трех уровнях: обращением к функциям MS-DOS; обращением к функциям BIOS; физическим доступом к аппаратным средствам. Ввод информации на уровне MS-DOS позволяет "пропустить" клавиатурный ввод через инсталлируемые драйверы, обеспечивает отслеживание нажатия комбинации клавиш Ctrl-C (Ctrl-Break), стандартную для MS-DOS обработку ошибок. Доступ к клавиатуре на уровне BIOS позволяет программе отслеживать нажатие всех, а не только символьных клавиш, выполнять управление аппаратурой клавиатуры и пр. Интерфейсом Turbo С с BIOS является функция bioskey(). Непосредственный доступ к буферу клавиатуры резко повышает производительность программы. В некоторых случаях необходима имитация нажатий клавиш клавиатуры с записью кодов непосредственно в буфер. При этом физически нажатия клавиш не происходят. Так строятся многие демонстрационные 56 программы, которые открывают или закрывают окна меню, выполняют необходимый выбор, показывают работу программы в "автоматическом" режиме и т.п. На том же самом принципе имитации нажатий клавиш построены программы, способные переносить одним нажатием клавиши целые куски текста из одной программы в любой текстовый редактор. Примером такой программы является входящая в Turbo С резидентная Help-система THELP.COM.

Клавиатура персонального компьютера содержит специальный встроенный микропроцессор. Он при каждом нажатии и отпускании клавиши определяет ее порядковый номер и помещает его в порт 60h специальной электронной схемы - программируемого периферийного интерфейса (ППИ). Далее этот код будем называть скэн-кодом. Скэн-код в первых 7 битах содержит порядковый номер нажатой клавиши, а восьмой бит равен 0, если клавиша была нажата (прямой скэнкод), и равен 1, если клавиша была отпущена (обратный скэн-код). Когда скэн-код записан в порт 60h, схема ППИ выдает сигнал "подтверждения", уведомляя микропроцессор клавиатуры о принятии кода.

Стандартный обработчик прерывания 9 - это программа, входящая в состав BIOS (BIOS ISR). BIOS ISR анализирует скэн-код и по специальным правилам преобразует его. Отметим, что по скэн-коду всегда можно установить, вследствие чего ISR получила управление: из-за нажатия или из-за отпускания клавиши.

MS-DOS имеет целую группу функций прерывания 21h для выполнения ввода 62 информации с клавиатуры. Последовательность действий системы при вводе с клавиатуры такова. Функция MS-DOS вызывает драйвер клавиатуры, передавая ему запрос на ввод одного символа из буфера клавиатуры. Драйвер, выполняя запрос, обращается к нужной функции прерывания 16h BIOS. ISR BIOS прерывания 16h читает из буфера клавиатуры нужное слово и передает в драйвер. Драйвер возвращает байт (обычно младший) в MS-DOS. Таким образом, функции MS-DOS и опирающиеся на них функции библиотеки Turbo С слабо зависят от особенностей аппаратуры, поскольку система от нее изолирована двумя слоями программного обеспечения - драйверами и BIOSом. Далее приводится характеристика функций MS-DOS, используемых для ввода с клавиатуры.

AH=01h - ввод с ожиданием со стандартного устройства ввода (клавиатуры). Выполняется "эхо" на экран вводимых символов. ASCII-код прочитанного символа помещается в AL. Если нажимается специальная клавиша, в AL возвращается 0, а второе обращение к функции возвращает расширенный скэн-код клавиши.

AH=06h - ввод-вывод с консоли. Если DL = FFh, выполняется ввод со стандартного устройства ввода без ожидания. Если буфер пуст, функция сообщает об этом установленным в 1 флагом нуля (ZF). В противном случае в регистре AL возвращается ASCII-код прочитанного символа.

AH=07h - ввод с консоли с ожиданием без "эха" на экран. ASCII-код прочитанного символа возвращается в AL. Если нажимается специальная клавиша, передаваемое в AL значение равно нулю, а второе обращение к функции возвращает расширенный скэн-код клавиши. Функция не выполняет "фильтрацию" ввода с клавиатуры. Это значит, что нажатие клавиши Backspace не стирает символ на экране, а только сдвигает курсор. Нажатие ENTER не переводит строку, а только перемещает курсор на начало строки.

AH=08h - подобна АН=07h, за исключением того, что если обнаруживается нажатие комбинации клавиш Ctrl-Break, вызывается прерывание 23h.

AH=0Bh - проверка состояния стандартного ввода. Возвращает в регистре AL значение FFh, если буфер клавиатуры не пуст, и 0 в противном случае. Функцию следует использовать перед выполнением функций АН=01h, 07h и 08h для того, чтобы избежать ожидания ввода, если он отсутствует. Кроме того, функция используется как средство проверки того, нажата ли комбинация клавиш CtrlBreak, если программа долгое время выполняет работу, не связанную с обращением к функциям MS-DOS. Периодическое выполнение функции позволяет аварийно завершить программу, например, в случае ее зацикливания.

AH=0Ch - ввод с клавиатуры с очисткой буфера. Значение в регистре AL 63 содержит номер выполняемой функции: 01, 06, 07, 08 или 0Ah. Поведение функции и возвращаемые значения описаны ранее в спецификации функций АН=01, 06, 07,08 или 0Ah.

Рассмотренные функции MS-DOS для ввода с клавиатуры могут вызываться напрямую из программы через функции geninterrupt(), int86(), intr() и т.п., либо неявно другими функциями ввода.

# **3. Листинг программы**

#include <iostream.h>

#include <conio.h>

#include <graphics.h>

#include <stdlib.h>

#include <time.h>

#include <dos.h>

#define FIRST 0

#define SECOND 1

#define NO\_ONE 3

int count;

// game functions

void draw\_board(char\*);

char winner(char\*);

void get\_computer\_move(char\*);

void get\_players\_move(char\*);

void announce\_winner(char);

void tictactoe\_multiplayer();

void tictactoe\_singleplayer();

char get\_fig();

void information();

// help functions

char get\_key();

int main()

{

int graph\_driver,

graph\_mode,

graph\_error\_code;

char menu\_item;

detectgraph(&graph\_driver, &graph\_mode);

initgraph(&graph\_driver, &graph\_mode, "C:\\TURBOCPP\\BGI");

graph\_error\_code = graphresult();

if (graph\_error\_code != grOk)

{

closegraph();

cputs("Graph mode error.\n");

return 255;

}

srand(time(NULL));

while (1)

{

count = 0;

clrscr();

cleardevice();

setbkcolor(BLACK);

setcolor(BLUE);

settextstyle(10,0,1);

outtextxy(210,0,"TIC-TAC-TOE");

outtextxy(210,60,"1. Singleplayer");

outtextxy(210,90,"2. Multiplayer ");

outtextxy(210,120,"3. Information ");

outtextxy(210,150,"0. Exit ");

line(400,200,400,500);

line(500,200,500,500);

line(300,300,600,300);

line(300,400,600,400);

line(530,230,570,270);

line(530,270,570,230);

circle(450,450,20);

menu\_item = get\_key();

switch (menu\_item)

{

case 1:

tictactoe\_singleplayer();

break;

case 2:

tictactoe\_multiplayer();

break;

case 3:

information();

break;

case 0:

exit(0);

break;

}

}

}

void tictactoe\_multiplayer()

{

char board[9];

for(int i = 0; i < 9; i++)

board[i]='a';

draw\_board(board);

while (winner(board) == NO\_ONE)

{

get\_players\_move(board);

draw\_board(board);

}

announce\_winner(winner(board));

}

void tictactoe\_singleplayer()

{

char human;

char board[9]; // доска

for(int i = 0; i < 9; i++)

board[i]='a';

human = rand()%2;

draw\_board(board);

while (winner(board) == NO\_ONE)

{

if (count%2 == human)

get\_players\_move(board);

else

{

get\_computer\_move(board);

draw\_board(board);

}

}

draw\_board(board);

announce\_winner(winner(board));

}

void draw\_board(char a[])

{

clrscr();

cleardevice();

setcolor(BLUE);

cout<<"\n\n\n\n 1 2 3\n\n\n\n\n";

cout<<"\n 4 5 6\n\n\n\n\n";

cout<<"\n 7 8 9\n\n\n\n\n\n";

line(100,50,100,350);

line(100,50,400,50);

line(400,50,400,350);

line(400,350,100,350);

line(200,50,200,350);

line(300,50,300,350);

line(100,150,400,150);

line(100,250,400,250);

if(a[0] == 'x')

{

line(130,80,170,120);

line(130,120,170,80);

}

else if(a[0] == 'o')

circle(150,100,20);

if(a[1] == 'x')

{

line(230,80,270,120);

line(230,120,270,80);

}

else if(a[1] == 'o')

circle(250,100,20);

if(a[2] == 'x')

{

line(330,80,370,120);

line(330,120,370,80);

}

else if(a[2] == 'o')

circle(350,100,20);

if(a[3] == 'x')

{

line(130,180,170,220);

line(130,220,170,180);

}

else if(a[3] == 'o')

circle(150,200,20);

if(a[4] == 'x')

{

line(230,180,270,220);

line(230,220,270,180);

}

else if(a[4] == 'o')

circle(250,200,20);

if(a[5] == 'x')

{

line(330,180,370,220);

line(330,220,370,180);

}

else if(a[5] == 'o')

circle(350,200,20);

if(a[6] == 'x')

{

line(130,280,170,320);

line(130,320,170,280);

}

else if(a[6] == 'o')

circle(150,300,20);

if(a[7] == 'x')

{

line(230,280,270,320);

line(230,320,270,280);

}

else if(a[7] == 'o')

circle(250,300,20);

if(a[8] == 'x')

{

line(330,280,370,320);

line(330,320,370,280);

}

else if(a[8] == 'o')

circle(350,300,20);

}

char winner(char g[])

{

if ((g[0]=='x'&&g[3]=='x'&&g[6]=='x')||

(g[1]=='x'&&g[4]=='x'&&g[7]=='x')||

(g[2]=='x'&&g[5]=='x'&&g[8]=='x')||

(g[0]=='x'&&g[1]=='x'&&g[2]=='x')||

(g[3]=='x'&&g[4]=='x'&&g[5]=='x')||

(g[6]=='x'&&g[7]=='x'&&g[8]=='x')||

(g[0]=='x'&&g[4]=='x'&&g[8]=='x')||

(g[2]=='x'&&g[4]=='x'&&g[6]=='x'))

return 0;

else if ((g[0]=='o'&&g[3]=='o'&&g[6]=='o')||

(g[1]=='o'&&g[4]=='o'&&g[7]=='o')||

(g[2]=='o'&&g[5]=='o'&&g[8]=='o')||

(g[0]=='o'&&g[1]=='o'&&g[2]=='o')||

(g[3]=='o'&&g[4]=='o'&&g[5]=='o')||

(g[6]=='o'&&g[7]=='o'&&g[8]=='o')||

(g[0]=='o'&&g[4]=='o'&&g[8]=='o')||

(g[2]=='o'&&g[4]=='o'&&g[6]=='o'))

return 1;

else if(g[0]!='a'&&g[1]!='a'&&g[2]!='a'&&g[3]!='a'&&g[4]!='a'&&g[5]!='a'&&g[6]!='a'&&g[7]!='a'&&g[8]!='a'&&count!=0)

return 2;

else

return 3;

}

void get\_players\_move(char g[])

{

char a, end;

do {

draw\_board(g);

if(count%2==0)

cout<<" Player 1: \n";

else

cout<<" Player 2: \n";

cout<<"\n Enter 0 to exit or Enter the place to drop ";

if(count%2==0)

cout<<"x";

else

cout<<"o";

a = get\_key();

if(a==0)

exit(0);

else

{

if(count%2==0 && g[a-1]=='a')

{

g[a-1]='x';

count++;

end = 0;

}

else if(count%2!=0 && g[a-1]=='a')

{

g[a-1]='o';

count++;

end = 0;

}

else

end = 1;

}

} while(end != 0);

}

void get\_computer\_move(char board[])

{

const int BEST\_MOVES[] = {4, 0, 2, 6, 8, 1, 3, 5, 7};

int move = 0, i = 0;

char found = 0;

while(!found && move < 9)

{

if (board[move] == 'a')

{

board[move] = get\_fig();

found = winner(board) == count%2;

board[move] = 'a';

}

if (!found)

++move;

}

if (!found)

{

move = 0;

++count;

while(!found && move < 9)

{

if (board[move] == 'a')

{

board[move] = get\_fig();

found = winner(board) == !count%2;

board[move] = 'a';

}

if (!found)

++move;

}

--count;

}

if (!found)

{

move = 0;

while(!found && i < 9)

{

move = BEST\_MOVES[i];

if (board[move] == 'a')

found = 1;

++i;

}

}

board[move] = get\_fig();

++count;

}

void announce\_winner(char winner)

{

if(winner == FIRST)

cout<<"\n\n X Wins!! Congradulations !!";

else if (winner == SECOND)

cout<<"\n\n O Wins!! Congradulations !!";

else

cout<<"\n\n The match is draw!!!";

get\_key();

}

char get\_fig()

{

if (count%2 == 0)

return 'x';

else

return 'o';

}

void information()

{

clrscr();

cleardevice();

setbkcolor(BLACK);

setcolor(BLUE);

settextstyle(10,0,1);

outtextxy(210,0,"Information");

cout<<"\n\n\n The goal of game is to be the first player to get 3 in a row on a 3x3 grid.";

cout<<"\n\n The player who is playing X always goes first.";

cout<<"\n Players placing X and O on the board until other player has 3 in a row";

cout<<"\n horizontally, vertically, or diagonally or until all squares are filled.";

cout<<"\n\n If a player is able to draw three X or three O in a row, then he wins.";

cout<<"\n If all squares are filled and noboby has made a row, then its a draw.";

line(400,200,400,500);

line(500,200,500,500);

line(300,300,600,300);

line(300,400,600,400);

line(530,230,570,270);

line(530,270,570,230);

circle(450,450,20);

get\_key();

}

// help functions

char get\_key()

{

union REGS in, out;

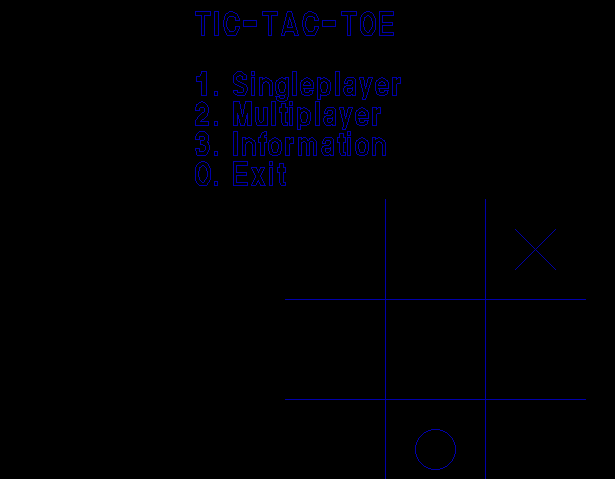
in.h.ah = 0x10; // функция 10h ожидание ввода символа

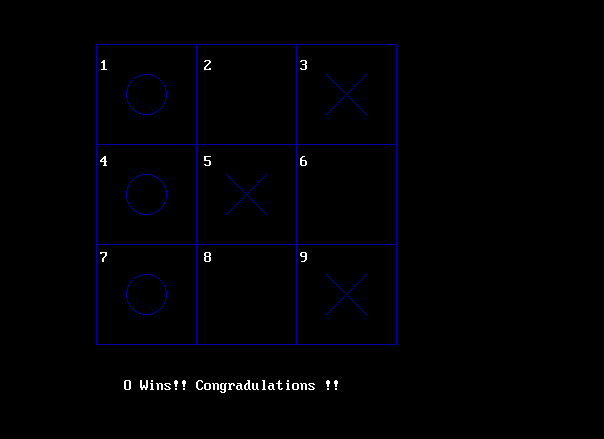
int86(0x16, &in, &out);

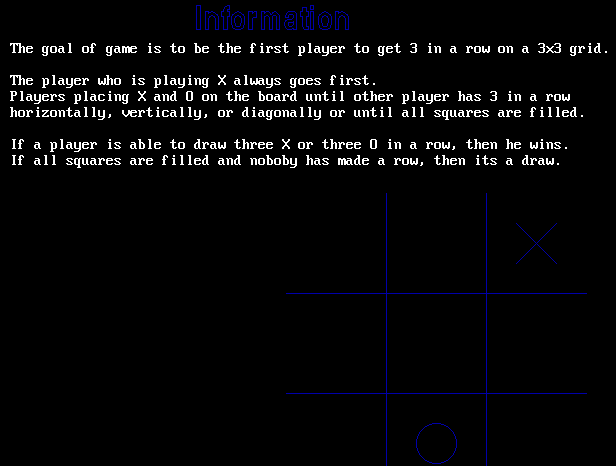
return out.h.al - '0';

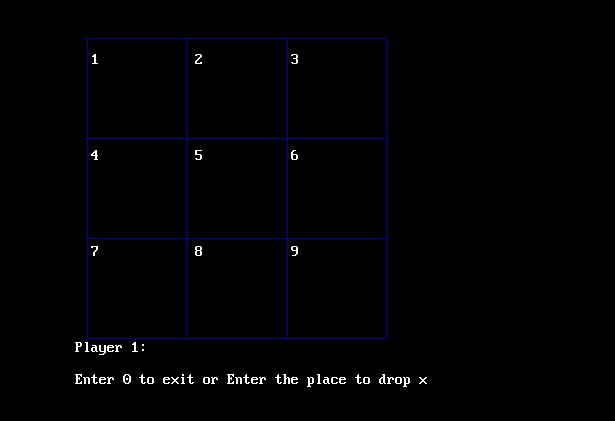
}

# **4. Примеры запуска программ**

Рисунок 1: Главное меню

Рисунок 2: Пример игры

Рисунок 3: Справка

Рисунок 4: Пример игры

# **Вывод**

В ходе лабораторной работы научились пользоваться графическим режимом, были приобретены навыки работы с клавиатурой, стандартными средствами библиотеки C++ средствами системы прерываний DOS и BIOS, обслуживающими клавиатуру. Задание выполнено в полном объёме.