**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

Кафедра ВТ

отчет

**по лабораторной работе № 5**

**по дисциплине «Организация ЭВМиС»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 0301 |  | Мельник Д. А. |
| Студент гр. 0301 |  | Лепов А. В. |
| Преподаватель |  | Костичев С.В. |

Санкт-Петербург

2022 г.

**СОДЕРЖАНИЕ**

[1. ВВЕДЕНИЕ 3](#_Toc121086165)

[1.1. Цель лабораторной работы 3](#_Toc121086166)

[1.2. Задание для выполнения 3](#_Toc121086167)

[2. КРАТКИЕ СВЕДЕНИЯ 4](#_Toc121086168)

[2.1. Общие положения 4](#_Toc121086169)

[2.2. BIOS ISR 4](#_Toc121086170)

[2.3. Особые нажатия клавиш 6](#_Toc121086171)

[Alt-ввод 6](#_Toc121086172)

[2.4. Функции прерывания 8](#_Toc121086173)

[2.5. Функции С++ 9](#_Toc121086174)

[2.6. Интерфейс прерываний 16h 10](#_Toc121086175)

[3. ДЕМОНСТРАЦИЯ РАБОТЫ ПРОГРАММЫ 11](#_Toc121086176)

[4. ЛИСТИНГ ПРОГРАММНОГО КОДА 13](#_Toc121086177)

[5.1. Файл «5\_LAB-8-TEAM-F4-CapsLock-Toggling.cpp» 13](#_Toc121086178)

[5.2. Файл «5\_LAB-8-TEAM-F4-Printing-Uppercase\_(additional)» 16](#_Toc121086179)

1. **ВВЕДЕНИЕ**
   1. **Цель лабораторной работы**

Целью лабораторной работы является изучение студентами возможностей прямого доступа к клавиатуре, ознакомление со стандартными средствами библиотеки C++ и средствами системы прерываний DOS и BIOS, обслуживающими клавиатуру.

* 1. **Задание для выполнения**

Обработку прерывания для клавиш F3, F4 дополнить следующим:

* При нажатии на F3 на экран выводится символ «А» или «а» (в зависимости от регистра);
* При нажатии на F4 изменяется значение регистра на противоположное.
  1. **Формализация задания**

Исполняемый файл «5\_LAB-8-TEAM-F4-CapsLock-Toggling.cpp» полностью соответствует заданию: «F3» выводит символ, а «F4» переключает регистр.

В качестве дополнения, написан модуль «5\_LAB-8-TEAM-F4-Printing-Uppercase\_(additional).cpp». Отличается он тем, что при нажатии «F3» выводит строчный символ, а при нажатии «F4» - прописной.

# КРАТКИЕ СВЕДЕНИЯ

* 1. **Общие положения**

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.

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 скэн-код), 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.

* 1. **BIOS ISR**

Действия BIOS ISR при нажатии и отпускании одной и той же клавиши различны. Клавиши в зависимости от алгоритма обработки их скэн-кода можно разделить на:

* шифт-клавиши (Right-Shift, Left-Shift, Alt, Ctrl);
* триггерные клавиши (NumLock, ScrollLock, CapsLock);
* клавиши с буферизацией расширенного кода;
* специальные клавиши (клавиша PrnScr, комбинация Alt-Ctrl-Del, комбинация Ctrl-C (Ctrl-Break)).

За каждой шифт- или триггерной клавишей закреплен свой бит в ячейках памяти по адресам 40: 17h и 40: 18h.

Текущее состояние шифт- и тригтерных клавиш используется BIOS-обработчиком прерывания от клавиатуры при определении правил преобразования скэн-кодов от других клавиш. Большинство клавиш и их комбинаций с шифт-клавишами - это клавиши с буферизацией расширенного кода: при их нажатии в специальный буфер памяти помещается двухбайтовый код, называемый BIOS-кодом клавиши. Младший байт этого кода равен ASCII-коду символа, либо нулю. Стар­ший байт равен скэн-коду клавиатуры, либо так называемому расширенному скэн-коду. Комбинация "ASCII-код/ скэн-код клавиатуры" генерируется в следующих случаях:

Если нажата клавиша клавиатуры, помеченная символом, входящим в ASCII-таблицу (называемая далее ASCII-клавишей). Так как прописные и строчные буквы имеют разный ASCII-код, при генерации BIOS-кода учитывается текущее состояние триггерной клавиши CapsLock и клавиши Shift.

Если нажаты некоторые из ASCII-клавиш в комбинации с нажатой и не отпущенной клавишей Ctrl, а также при нажатии клавиш Backspace, ENTER (Ввод), Tab и Esc (Ключ). В этом случае младший байт BIOS-кода клавиши равен одному из управляющих ASCII-кодов. Это ASCII-коды со значениями 00 - 31, которые не входят в число печатаемых символов, а используются для управления периферийными устройствами. Например, нажатие клавиши ENTER порождает управляющий символ Carriage Return (Возврат каретки), нажатие клавиши Tab порождает управляющий символ горизонтальной табуляции, комбинация Ctrl-L - управляющий символ Form Feed (Перевод формы), комбинация Ctrl-B - управляющий символ Bell (Звонок). Нажатие комбинации Ctrl-M соответствует также управляющему символу Carriage Return, но полный BIOS-код этой клавиши равен 13/50, а в случае нажатия клавиши ENTER - 13/28.

* 1. **Особые нажатия клавиш**

Некоторые нажатия клавиш обрабатываются ISR BIOS особым образом. К их числу относятся:

* клавиша PrnScr, при нажатии которой ISR BIOS выполняет программное прерывание 5;
* комбинация Alt-Ctrl-Dei, обнаружив такую комбинацию, ISR BIOS передает управление программе начальной загрузки. Эта программа также входит в состав BIOS;
* комбинация Ctrl-C (Ctrl-Break); ISR BIOS записывает по абсолютному адресу памяти 00471h значение 80h. Оно используется как флаг, сигнализирующий о желании пользователя остановить выполнение текущей программы. Значение этого флага проверяют при своем выполнении функции MS-DOS, работающие с файлами stdin, stdout, stdprn и stdaux.

**Alt-ввод**

Особым образом обрабатывается так называемый Alt-ввод. Если нажимается и удерживается нажатой клавиша Alt и на цифровой клавиатуре набираются цифры, то после отпускания клавиши Alt в буфер клавиатуры помещается двухбайтовый код, старший байт которого равен нулю, а младший байт содержит набранный цифрами код.

Буфер BIOS для записи кодов клавиш занимает 32 байта оперативной памяти с адреса 40:lEh до 40:3Eh. Запись информации в буфер выполняет ISR BIOS прерывания 9, чтение - функции ISR BIOS прерывания 16h. Буфер клавиатуры рассчитан на 15 нажатий клавиш, генерирующих двухбайтовые коды и поэтому имеет 30 байт для кодов клавиш и еще два дополнительных байта, которые резервируются под двухбайтовый код для клавиши ENTER.

Буфер организуется как кольцевая очередь, доступ к которой осуществляется с помощью указателя «головы» (head pointer), адрес которого 40:1Ah, и указателя «хвоста» (tail pointer), адрес которого 40:1Ch. Указатель "хвоста" задает смещение до слова, где будет записан обработчиком прерывания 9 код буферизуемой клавиши, т.е. первое свободное слово буфера. Указатель "головы" задает смещение слова, которое будет возвращено запросу буфе­ризованного ввода с клавиатуры, сделанного операционной системой или BIOSoм.

При каждом нажатии клавиши, для которой генерируется двухбайтовый код, ISR BIOS прерывания 9,'используя текущее значение указателя "хвоста", записывает в память образованный двухбайтовый код. После этого указатель "хвоста" увеличивается на 2. Если указатель "хвоста" перед доступом к буферу указывает на верхнюю границу буфера (на слово 40:3Eh), указатель после записи в буфер "перепрыгивает" на начало буфера, т.е. ему присваивается значение 40:1Eh. Поэ­тому значение указателя "хвоста" может быть и меньше значения указателя "головы". Это значит, что указатель "хвоста" "перескочил" назад к нижней границе буфера. Когда указатель "хвоста" догонит указатель "головы", наступит пере­полнение буфера. В этом случае указатель "хвоста" задает смещение до "холостой" позиции. Каждое новое нажатие клавиши игнорируется BIOS-обработчиком; код клавиши не помещается в буфер, и звучит сигнал динамика.

Указатель "головы" используется BIOS-обработчиком прерывания 16h, которое вызывается непосредственно из приклад­ной программы или функциями MS-DOS ввода с клавиатуры.

Буфер клавиатуры - это классический пример использования кольцевого буфера для организации асинхронного взаимодействия двух программ по схеме "производитель-потребитель". Одна из программ (ISR BIOS прерывания 9) "производит" информацию или, как говорят, является процессом-производителем. Исполняемая программа через функцию АН= 00h прерывания 16h BIOS "потребляет" информацию или является процессом-потребителем. Асинхронность взаимодействия означает, что запись в буфер новой информации и чтение из него происходят в случайные, не связанные между собой моменты времени.

* 1. **Функции прерывания**

MS-DOS имеет целую группу функций прерывания 21h для выполнения ввода 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.

Характеристика функций MS-DOS, используемых для ввода с клавиатуры:

1. 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.
2. 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.
3. 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.
4. AH=08h - подобна АН=07h, за исключением того, что если обнаруживается нажатие комбинации клавиш Ctrl-Break, вы­зывается прерывание 23h.
5. 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 Ctrl-Break, 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.
6. AH=0Ch - ввод с клавиатуры с очисткой буфера. Значение в регистре AL 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.
   1. **Функции С++**

* int getch (void) - выполняет ввод с клавиатуры через функцию MS-DOS АН=07h. Она не выполняет "эхо" вывода на экран.
* int getche (void) - выполняет небуферизуемый ввод с клавиатуры через функцию MS-DOS AH=07h, но в отличие от предыдущей функции обеспечивает вывод введенного символа на экран.
* char \*getpass(char \* prompt) - выводит на экран ASCII-строку, на начало которой указывает prompt, a затем принимает с клавиатуры без "эха" строку символов.
* int kbhit (void) - проверяет, пуст ли буфер клавиатуры.
  1. **Интерфейс прерываний 16h**

Интерфейсом программ в персональном компьютере с клавиатурой является прерывание 16h BIOS.

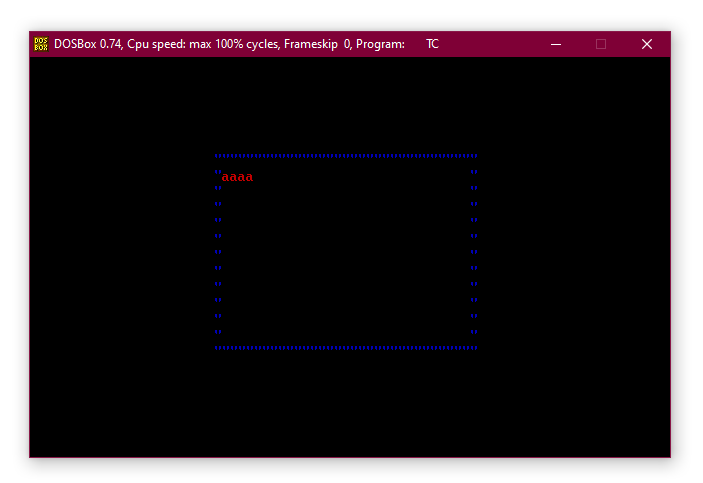
* АН = 00h - чтение с ожиданием двухбайтового кода из буфера клавиатуры.
* АН = 0lh - чтение без ожидания двухбайтового кода из буфера клавиатуры.
* АН = 02h - определение состояния шифт- и триггерных клавиш.
* АН = 05h не имеет аналогов в библиотеке Turbo С и может использоваться для имитации нажатии клавиш в демонстрационных программах, программах переноса текста и т.д.
* АН = 10 - 12h являются аналогами функций 00 - 02h, но предназначены для использования в компьютерах с клавиатурой 101 /102 клавиши.
* АН = 00 - 02h прерывания 16h BIOS положены в основу функции bioskey() библиотеки Turbo С.

int bioskey(int cmd) - обращается в зависимости от значения в cmd к функциям АН = 00 - 02h прерывания 16h. Возвращаемое функцией значение повторяет значение регистра АХ при выходе из прерывания.

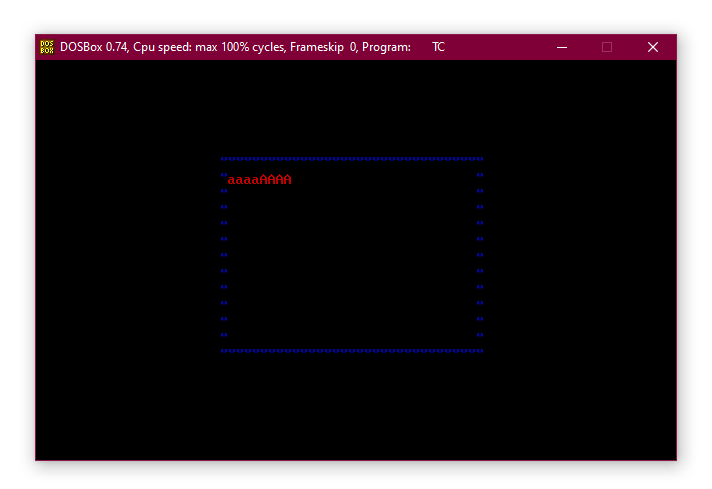
Функции:

|  |  |
| --- | --- |
| getch (void) | Считать клавишу без «эха» |
| getche (void) | Считать клавишу |
| kbhit (void) | Проверка, пуст ли буфер |
| bioskey(int cmd) | Считывание через прерывания |

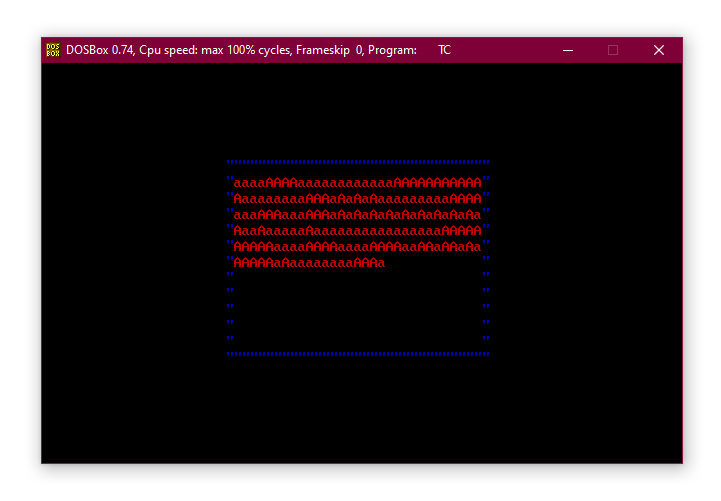
# ДЕМОНСТРАЦИЯ РАБОТЫ ПРОГРАММЫ

**

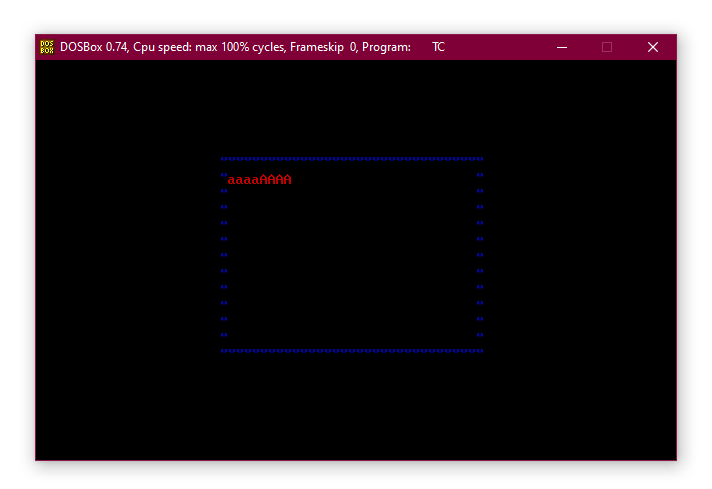
1. *Результат работы программы при   
   четырехкратном нажатии «F3»*

**

1. *Результат работы программы при   
   четырехкратном нажатии «F3», одного нажатия на «F4»   
   и снова четырехкратном нажатии «F3»*

**

1. *Результат работы программы при   
   случайном наборе символов*

**

1. *Результат работы программы* модуля   
   «*5\_LAB-8-TEAM-F4-Printing-Uppercase\_(additional)*.cpp*»   
   при четырехкратном нажатии «F3» и затем «F4»*

# ЛИСТИНГ ПРОГРАММНОГО КОДА

**5.1. Файл «5\_LAB-8-TEAM-F4-CapsLock-Toggling.cpp»**

#include <stdio.h>

#include <conio.h>

#include <dos.h>

//#include <WinAble.h> // didn't work with turboc++

///////////////////////////////////

//                               //

//    System kbhit interrupts    //

//                               //

///////////////////////////////////

int keyPressHandler(int &work)

{

    union REGS rg;

    rg.h.ah = 2;

    int86(0x16, &rg, &rg);

    char ch;

    char far \*memory1=(char far \*)0x417;

    char far \*memory2=(char far \*)0x418;

    if (kbhit())

    {

        ch = getch();

        switch(ch)

        {

            case 61:

            // In case F3 pressed (3d in hex = 61 in dec)

            // printing 'A'/'a' symbol

                // if <CapsLock> not toggled

                if((rg.h.al & 0x40) == 0) cprintf("a");

                else cprintf("A");

                //::keybd\_event(VkKeyScan('A'),0x9e, 0, 0);                 // 'A' Pressing     - by <WinAble.h> lib

                //::keybd\_event(VkKeyScan('A'),0x9e, KEYEVENTF\_KEYUP,0);    // 'A' Releasing    - by <WinAble.h> lib

                break;

            case 62:

            // In case F4 pressed (3e in hex = 62 in dec)

            // works as CapsLock

                \*memory1=\*memory1 ^ 64;

            // other methods

                // if((rg.h.al & 0x40) == 0)    // if <CapsLock> not toggled

                // {

                //     outp(0x40,0xed);         // set keyboard processor to alter indicators

                //     outp(0x40,0);            // turn on indication

                //     cprintf("NO");   //debbug

                // }

                // else

                // {

                //     outp(0x40,0xed);         // set keyboard processor to alter indicators

                //     outp(0x40,0);            // turn off indication

                //     cprintf("YES");  //debbug

                // }

            // other methods

                //::keybd\_event(VK\_CAPITAL, 0x45, KEYEVENTF\_EXTENDEDKEY, 0 );                   // Pressing down    - by <WinAble.h> lib

                //::keybd\_event(VK\_CAPITAL, 0x45, KEYEVENTF\_EXTENDEDKEY | KEYEVENTF\_KEYUP, 0);  // Releasing key    - by <WinAble.h> lib

                break;

            case 27:

                work = 0;

                break;

            default:

                break;

        }

        return 1;

    }

    return 0;

}

//////////////////////////////////////////

//                                      //

//    drawing the application window    //

//                                      //

//////////////////////////////////////////

void WidnowDrawing(int x1, int y1, int x2, int y2, int textback, int textcol)

{

    window (x1-1, y1-1, x2+1, y2+2);

    textbackground(textback);

    textcolor(textcol);

    clrscr();

    for (int i = 0; i < x2 - x1 + 3; i++) cprintf("\"");

    for(int j = 0; j < (y2 - y1 + 1); j++)

    {

        cprintf("\"");

        for(int k = 0; k < (x2 - x1 + 1); k++) cprintf(" ");

        cprintf("\"");

    };

    for (int l = 0; l < x2 - x1 + 3; l++) cprintf("\"");

    window(x1,y1,x2,y2);

    textbackground(textback);

    textcolor(textcol);

    clrscr();

}

//////////////////////////////////////

//                                  //

//    application initialization    //

//                                  //

//////////////////////////////////////

main()

{

    int work = 1;

    int x1 = 25, y1 = 8, x2 = 55, y2 = 18;

    clrscr();

    textbackground(BLACK);

    WidnowDrawing(x1, y1, x2, y2, BLACK, BLUE);

    \_setcursortype (\_NOCURSOR);

    textcolor(RED);

    gotoxy(1,1);

    while (work)

    {

        keyPressHandler(work);

    }

    return 0;

}

**5.2. Файл «5\_LAB-8-TEAM-F4-Printing-Uppercase\_(additional)»**

#include <stdio.h>

#include <conio.h>

///////////////////////////////////

//                               //

//    System kbhit interrupts    //

//                               //

///////////////////////////////////

int keyPressHandler(int &work)

{

    char ch;

    int capslock = 0;

    if (kbhit())

    {

        ch = getch();

        switch (ch)

        {

            case 61: // In case F3 pressed (3d in hex = 61 in dec) // printing 'A' symbol

                cprintf("A");

                break;

            case 62: // In case F4 pressed (3e in hex = 62 in dec) // printing 'a' symbol

                cprintf("a");

                break;

            case 27:

                work = 0;

                break;

            default:

                break;

        }

        return 1;

    }

    return 0;

}

//////////////////////////////////////////

//                                      //

//    drawing the application window    //

//                                      //

//////////////////////////////////////////

void WidnowDrawing(int x1, int y1, int x2, int y2, int textback, int textcol)

{

    window (x1-1, y1-1, x2+1, y2+2);

    textbackground(textback);

    textcolor(textcol);

    clrscr();

    for (int i = 0; i < x2 - x1 + 3; i++) cprintf("\"");

    for(int j = 0; j < (y2 - y1 + 1); j++)

    {

        cprintf("\"");

        for(int k = 0; k < (x2 - x1 + 1); k++) cprintf(" ");

        cprintf("\"");

    };

    for (int l = 0; l < x2 - x1 + 3; l++) cprintf("\"");

    window(x1,y1,x2,y2);

    textbackground(textback);

    textcolor(textcol);

    clrscr();

}

//////////////////////////////////////

//                                  //

//    application initialization    //

//                                  //

//////////////////////////////////////

main()

{

    int work = 1;

    int x1 = 25, y1 = 8, x2 = 55, y2 = 18;

    clrscr();

    textbackground(BLACK);

    WidnowDrawing(x1, y1, x2, y2, BLACK, BLUE);

    \_setcursortype (\_NOCURSOR);

    textcolor(RED);

    gotoxy(1,1);

    while (work)

    {

        keyPressHandler(work);

    }

    return 0;

}