**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра вычислительной техники**

отчет

**по лабораторной работе №4**

**по дисциплине «Организация ЭВМ и систем»**

Тема: К**лавиатура IBM PC. Использование прерываний**

|  |  |  |
| --- | --- | --- |
| Студент гр. 8363 |  | Нерсисян А.С. |
| Преподаватель |  | Гречухин М.Н. |

Санкт-Петербург

2019

**Цель работы.**

Изучение возможностей работы с клавиатурой, ознакомление со стандартными средствами библиотеки C++ и средствами системы прерываний DOS и BIOS, обслуживающими клавиатуру..

**Задание на лабораторную работу.**

1. Разработать, написать и отладить программу управления перемещением символа ("\*") в пределах заданного на экране окна. Для управления использовать функциональные клавиши Fl - F12. Для ввода использовать стандартные функции языка C++. Сохранить отлаженную программу.

2. Изменить программу, заменив стандартные функции библиотеки C++ своими. Для написания функций используйте заданное прерывание, если его возможностей достаточно. Если его возможностей недостаточно, то замените его по своему усмотрению. Сохраните отлаженную программу.

3. Отлаженные программы предъявить преподавателю.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № варианта | X1 | Y1 | X2 | Y2 | Вид движения | Клавиши управления | Номер прерывания |
| 11 | 20 | 10 | 60 | 20 | Постоянное | F3, F4 | INT 21h |

**Краткие сведения о подсистеме ввода информации с клавиатуры, используемых прерываниях, буфере клавиатуры и функциях обслуживания ввода с клавиатуры.**

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 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 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.

Если клавиша остается нажатой дольше некоторого времени задержки (delay value), микропроцессор клавиатуры начинает генерировать с заданной частотой (typematic rate) прямой скэн-код нажатой клавиши. Значения задержки и частоты повторения могут устанавливаться в нужные значения либо через порты клавиатуры, либо через функцию АН = 03h прерывания 16h BIOS. Когда скэн-код принят схемой ППИ, аппаратура компьютера генерирует аппаратное прерывание с номером 9.

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.

**Буфер клавиатуры**

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.

**Функции прерывания 21h MS-DOS для ввода информации с клавиатуры**

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. 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.

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 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.

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.

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

**Ввод информации с клавиатуры средствами BIOS**

Интерфейсом программ в персональном компьютере с клавиатурой является прерывание 16h BIOS. Далее приводится описание его функций.

АН = 00h - чтение с ожиданием двухбайтового кода из буфера клавиатуры. Прочитанный код возвращается в регистре АХ: младший байт - в регистре AL, старший - в АН. Если нажата ASCII-клавиша, в AL помещается ASCII-код символа, в АН - скэн-код. При нажатии специальных клавиш AL равен 0, а в АН возвращается расширенный скэн-код.

АН = 0lh - чтение без ожидания двухбайтового кода из буфера клавиатуры. Если буфер пуст, в 1 выставляется флаг нуля ZF. В противном случае в АХ возвращается двухбайтовый код из буфера клавиатуры, но продвижение указателя "головы" буфера не производится, т.е. код "остается" в буфере.

АН = 02h - определение состояния шифт- и триггерных клавиш. В регистре AL возвращается содержимое байта по адресу 40:17h (см. табл. 4.1).

Функция 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.

Функции АН = 10 - 12h являются аналогами функций 00 - 02h, но предназначены для использования в компьютерах с клавиатурой 101 /102 клавиши.

Функции АН = 00 - 02h прерывания 16h BIOS положены в основу функции bioskey() библиотеки Turbo С. Далее следует описание этой функции.

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

**Алгоритмы**

Для того, чтобы движение символа было постоянным используется два вложенных цикла while (с конструкцией switch-case): первая, чтобы программа выполнялась до нажатия кнопки ESC, вторая – для непрерывного движения символа ('\*') с условием нажатия кнопки, определяющая противоположное направление.

while(c != 27) // пока не нажали ESC

{

switch(c) // от нажатой клавиши

{

case 75: // <-

case 61: // F3

while(c != 77 || c != 61)

{

// сдвиг во одну сторону

}

break;

case 77: // ->

case 62: // F4

while(c != 75 || c != 62)

{

//сдвиг в другую сторону

}

break;

default:

break;

}

В программе кроме функциональных клавиш F3 и F4, предусмотрена управление символом клавишами из набора "стрелка вправо" и "стрелка влево".

Для улавливания нажатия кнопок в первой части лабораторной работы была использована функция bioskey() из библиотеки dos.h. Данная функция при аргументе 0 возвращает ASCII код нажатой клавиши, а при аргументе 1, проверяет на нажатие кнопок вообще и возвращает 0 при бездействии клавиатуры.

Во второй части лабораторной работы была использована собственная реализация проверки на нажатие и возвращение ASCII кода нажатой клавиши.

Ниже представлена реализация вышеупамянутой функции

char keyIsPressed(int x) {

union REGS r;

if (x==1) { // какая кнопка нажата

r.h.ah = 0x0B;

int86(0x21, &r, &r);

return r.h.al;

}

if(x==0) { // проверка на нажатие кнопок вообще

r.h.ah = 0x7

int86(0x21, &r, &r);//int86(номер прерывания,откуда,куда)

return r.h.al;

}

return 0;

}

**Текст программы №1**

#include <stdio.h>

#include <conio.h>

#include <bios.h>

#include <dos.h> //delay

//esc - 27

//up - 0 72

//left - 0 75

//right - 0 77

//down - 0 80

// F3 - 61

// F4 - 62

int main()

{

textbackground(0);

clrscr();

char c = 0;

int xMin = 20;

int xMax = 60;

int yMin = 10;

int yMax = 20;

window(xMin, yMin, xMax, yMax);

textbackground(4);

clrscr();

int winWidth = xMax - xMin + 1;

int winHeight = yMax - yMin + 1;

int curX = 1, curY = 1;

gotoxy(curX, curY);

printf("\*");

gotoxy(curX, curY);

while(c != 27) // пока не нажали ESC

{

c = 0;

unsigned int tmp = 0;

while(bioskey(1) == 0); // bioskey(1) проверка на нажатие кнопок вообще

tmp = bioskey(0); // bioskey(0) возвращает нажатую

c = tmp%256;

if(c == 0)

c = tmp/256;

clrscr();

switch(c)

{

case 75: // <-

case 61: // F3

while(c != 77 || c != 61)

{

gotoxy(curX, curY);

printf(" ");

--curX;

if(curX == 0)

{

curX = winWidth;

--curY;

}

if(curY == 0)

curY = winHeight;

gotoxy(curX, curY);

printf("\*");

gotoxy(curX, curY);

if (bioskey(1) != 0 ) break;

delay(70);

}

break;

case 77: // ->

case 62: // F4

while(c != 75 || c != 62)

{

gotoxy(curX, curY);

printf(" ");

++curX;

if(curX == winWidth+1)

{

curX = 1;

++curY;

}

if(curY == winHeight+1)

curY = 1;

gotoxy(curX, curY);

printf("\*");

gotoxy(curX, curY);

if (bioskey(1) != 0 ) break;

delay(70);

}

break;

default:

break;

}

}

return 0;

}

**Текст программы №2**

#include <stdio.h>

#include <conio.h>

#include <bios.h>

#include <dos.h>

char keyIsPressed(int x) {

union REGS r;

if (x==1) { // какая кнопка нажата

r.h.ah = 0x0B;

int86(0x21, &r, &r);

return r.h.al;

}

if(x==0) { // проверка на нажатие кнопок вообще

r.h.ah = 0x7;

int86(0x21, &r, &r);

return r.h.al;

}

return 0;

}

int main() {

textbackground(0);

clrscr();

char c = 0;

int xMin = 20;

int xMax = 60;

int yMin = 10;

int yMax = 20;

window(xMin, yMin, xMax, yMax);

textbackground(4);

clrscr();

int winWidth = xMax - xMin + 1;

int winHeight = yMax - yMin + 1;

int curX = 1, curY = 1;

gotoxy(curX, curY);

printf("\*");

gotoxy(curX, curY);

while(c != 27) // // пока не нажали ESC

{

c = 0;

unsigned int tmp = 0;

while(keyIsPressed(1) == 0); // проверка на нажатие кнопок вообще

tmp = keyIsPressed(0); // bioskey(0) возвращает нажатую

c = tmp%256;

if(c == 0)

c = tmp/256;

clrscr();

switch(c)

{

case 75: // <-

case 61: // F3

while(c != 77 || c != 61)

{

gotoxy(curX, curY);

printf(" ");

--curX;

if(curX == 0)

{

curX = winWidth;

--curY;

}

if(curY == 0)

curY = winHeight;

gotoxy(curX, curY);

printf("\*");

gotoxy(curX, curY);

if (keyIsPressed(1) != 0)

break;

delay(70);

}

break;

case 77: // ->

case 62: // F4

while(c != 75 || c != 62)

{

gotoxy(curX, curY);

printf(" ");

++curX;

if(curX == winWidth+1)

{

curX = 1;

++curY;

}

if(curY == winHeight+1)

curY = 1;

gotoxy(curX, curY);

printf("\*");

gotoxy(curX, curY);

if (keyIsPressed(1) != 0)

break;

delay(70);

}

break;

default:

break;

}

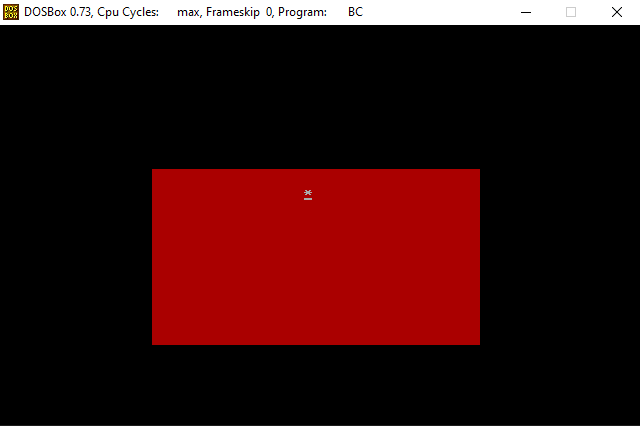
}

return 0;

}

**Примеры запуска программы**

На рисунке демонстрирована работа программы. Программа в ограниченной окне выводит на экран символ '\*', движением которого можно управлять функциональными клавишами F3 и F4, а также клавишами из набора "стрелка вправо" и "стрелка влево".

****

**Вывод**

В ходе выполнения данной лабораторной работы была изучена работа с клавиатурой: ознакомились со стандартными средствами библиотеки C++ и средствами системы прерываний DOS и BIOS, обслуживающими клавиатуру.

**Структурная схема аппаратных средств, используемых при выполнении программы с необходимой степенью детализации содержимого блоков.**

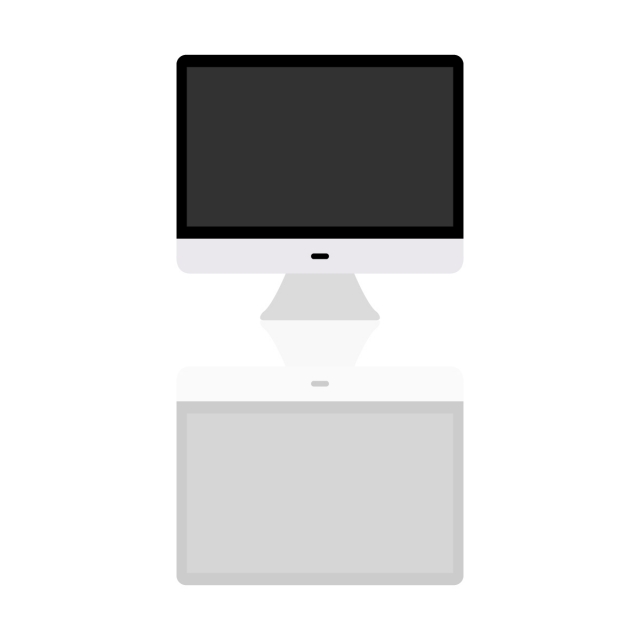
**ПЗУ**

**(RОM)**

**ОЗУ**

**(RAM)**

Контроллер



**ИП**

Монитор

Контроллер

**ЦП**

**(CPU)**

СИСТЕМНАЯ ШИНА



Клавиатура

Контроллер