**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: «Клавиатура IMB PC. Использование прерываний»**

|  |  |  |
| --- | --- | --- |
| Студенты гр. 3352 |  | Гультяев А.С.  Портнов В.В.  Шушков В.А. |
| Преподаватель |  | Ильин С. Е. |

Санкт-Петербург

2024

**Цель работы:** изучение работы с регистрами, прерыванием, а также записи с использованием BIOS.

# 1. Краткие сведения об аппаратных средствах персонального компьютера для ввода информации с клавиатуры, анализе и преобразовании скан-кода и ввода с клавиатуры средствами BIOS.

Подавляющее большинство программ выполняют ввод информации с клавиатуры. Ввод информации в компьютер может быть выполнен на трех уровнях:

1. обращением к функциям MS-DOS;
2. обращением к функциям BIOS;
3. физическим доступом к аппаратным средствам.

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.

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:

1. шифт-клавиши (Right-Shift, Left-Shift, Alt, Ctrl);
2. триггерные клавиши (NumLock, ScrollLock, CapsLock);
3. клавиши с буферизацией расширенного кода;
4. 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 (табл. 1).

Таблица 1 – Состояние шифт- и триггерных клавиш

|  |  |
| --- | --- |
| Бит | Состояние шифт- и триггерных клавиш |
|  | Байт 40:17h |
| 0 | Нажата и не отпущена клавиша Right Shift |
| 1 | Нажата и не отпущена клавиша Left Shift |
| 2 | Нажата и не отпущена клавиша Ctrl |
| 3 | Нажата и не отпущена клавиша Alt |
| 4 | Зафиксирован скроллинг экрана (ScrollLock - включен) |
| 5 | Включена цифровая клавиатура (NumLock - включен) |
| 6 | Зафиксирован верхний регистр (CapsLock - включен) |
| 7 | Включен режим вставки (хотя клавиша Ins не является триггерной, BIOS-обработчик фиксирует каждое ее нажатие, а код клавиши помещается еще и в буфер клавиатуры) |
|  | Байт40:18h |
| 0 | Нажата и не отпущена клавиша Left Ctrl |
| 1 | Нажата и не отпущена клавиша Left Alt |
| 2 | Нажата клавиша System Request (System) |
| 3 | Включен режим Pause(Ctrl-NumLock) |
| 4 | Нажата и не отпущена клавиша ScrollLock |
| 5 | Нажата и не отпущена клавиша NumLock |
| 6 | Нажата и не отпущена клавиша CapsLock |
| 7 | Нажата и не отпущена клавиша Ins |

Основные функции:

1. int getch (void) - выполняет ввод с клавиатуры через функцию MS-DOS АН=07h. Она не выполняет "эхо" вывода на экран. В этой связи полезна для организации интерфейса с пользователем, при котором нажатие той или иной клавиши вызывает немед­ленную реакцию программы без отображения введенного символа на экране.
2. 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. Перевод строки происходит при достижении правой вертикальной границы текущего активного окна.
3. char \*getpass(char \* prompt)- выводит на экран ASCII-строку, на начало которой указывает prompt, a затем принимает с клавиатуры без "эха" строку символов. Вводимые символы (не более 7) помещаются во внутреннюю статическую память. Функция возвращает указатель на внутреннюю статическую строку, переопределяемую каждым новым обращением к функции. Основное назначение данной функции - ввод паролей в программе без отображения их на экран.
4. int kbhit (void) - проверяет, пуст ли буфер клавиатуры. Если в буфере есть символы, функция возвращает ненулевое значение, в противном случае она возвращает 0. Использует функцию 0Bh MS-DOS. Является удобным средством предотвращения "зацикливания" при ожидании невозможного в данный момент события. Кроме того, при выполнении функции 0Bh осуществляется проверка нажатия комбинации клавиш Ctrl-Break, что позволяет выполнить аварийное завершение программы.

# 2. Задание на лабораторную работу.

1. Написать программу, в которой в окне с координатами (20, 8, 60, 18) с постоянным движением, номером прерывания INT 16h, и клавишами управления F9-F12 происходит перемещение символа.

2. Изменить программу, заменив стандартные функции библиотеки С++, такие как kbhit и getch своими, с использованием прерывания INT 16h.

# 3. Алгоритм и тексты отлаженных программ.

Первой представлена блок-схема функции getButton() (Рис. 1). В начале функции объявляется ключ, который будет возвращаться в случае не нажатой клавиши, а следом происходит ожидание в течении 200 миллисекунд. После чего следует проверка, была ли нажата какая-либо клавиша, если нет, то функция возвращает начальный ключ, если была нажата, то происходит проверка, какой ASCII код клавиши был нажат. После проверки на то, какая клавиша была нажата, возвращается необходимый символ.

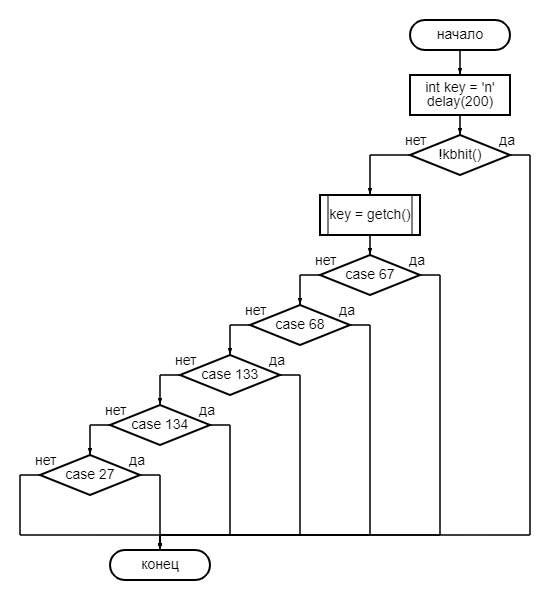


Рисунок 1 – Функция getButton()

Функция draw(int x, int y, char symbol) (Рис. 2) отвечает за то, чтобы просто поставить символ, который будет двигаться в необходимую точку.

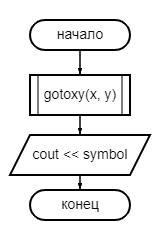


Рисунок 2 – Функция draw()

Далее следует функция move() (Рис. 3). Она отвечает за основную работу программы, а именно изначально ставит символ в центр окна, после чего начинается бесконечный цикл, который вызывает функции getButton и если функция возвращает определенное значение, то текущий ключ приравнивается к предыдущему. Далее начинается проверка на то, какое значение имеет ключ и в зависимости от значения ключа(нажатой клавиши), символ передвигается в необходимую сторону, при этом идет проверка на то, что если символ достиг пределов окна, то он переносится в начало параллельной стороны.

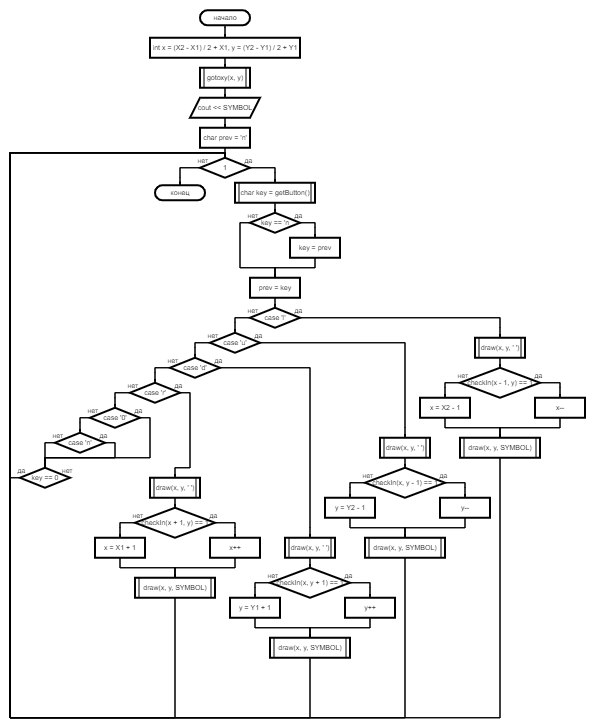


Рисунок 3 – Функция move()

В функции drawBorder() (Рис. 4) у нас в зависимости от заданных координат рисуется область окна, в котором происходит движение символа.

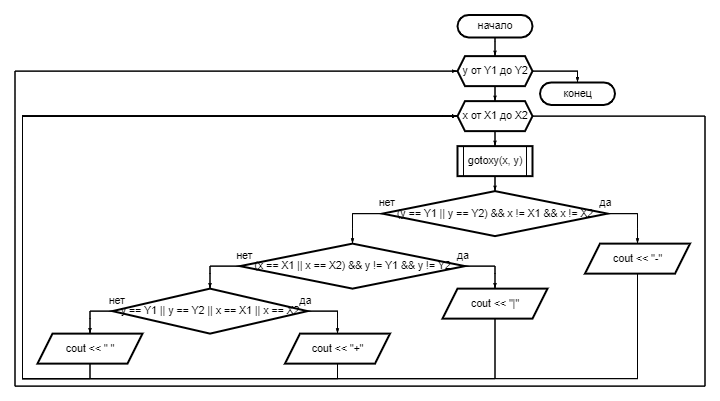


Рисунок 4 – Функция drawBorder()

В функции main() (Рис. 5) у нас происходит вызов функций, которые были описаны ранее.

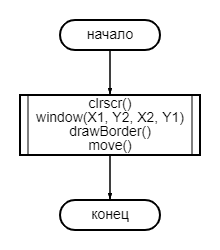


Рисунок 5 – Функция main()

Текст программы (версия с использованием функций kbhit, getch)

#include <iostream.h>

#include <conio.h>

#include <dos.h>

#include <time.h>

const int X1 = 20, Y1 = 8, X2 = 60, Y2 = 18;

const char SYMBOL = '\*';

char getButton() {

int key = 'n';

delay(200);

if (!kbhit()) {

return 'n';

}

key = getch();

switch (key) {

case 67: return 'l';

case 68: return 'u';

case 133: return 'd';

case 134: return 'r';

case 27: return '0';

default: return 'n';

}

}

void draw(int x, int y, char symbol) {

gotoxy(x, y);

cout << symbol;

}

int checkIn(int x, int y) {

if (x > X1 && x < X2 && y > Y1 && y < Y2) return 1;

return 0;

}

void move() {

int x = (X2 - X1) / 2 + X1, y = (Y2 - Y1) / 2 + Y1;

gotoxy(x, y);

cout << SYMBOL;

char prev = 'n';

while (1) {

char key = getButton();

if (key == 'n') key = prev;

prev = key;

switch (key) {

case 'l':

draw(x, y, ' ');

if (checkIn(x - 1, y) == 1) x--;

else x = X2 - 1;

draw(x, y, SYMBOL);

break;

case 'u':

draw(x, y, ' ');

if (checkIn(x, y - 1) == 1) y--;

else y = Y2 - 1;

draw(x, y, SYMBOL);

break;

case 'd':

draw(x, y, ' ');

if (checkIn(x, y + 1) == 1) y++;

else y = Y1 + 1;

draw(x, y, SYMBOL);

break;

case 'r':

draw(x, y, ' ');

if (checkIn(x + 1, y) == 1) x++;

else x = X1 + 1;

draw(x, y, SYMBOL);

break;

case '0': break;

case 'n': break;

}

if (key == '0') break;

}

}

void drawBorder() {

for (int y = Y1; y <= Y2; y++) {

for (int x = X1; x <= X2; x++) {

gotoxy(x, y);

if ((y == Y1 || y == Y2) && x != X1 && x != X2) {

cout << "-";

}

else if ((x == X1 || x == X2) && y != Y1 && y != Y2) {

cout << "|";

}

else if (y == Y1 || y == Y2 || x == X1 || x == X2) {

cout << "+";

}

else {

cout << " ";

}

}

}

}

int main() {

clrscr();

window(X1, Y2, X2, Y1);

drawBorder();

move();

return 0;

}

Текст программы (версия с реализацией функций getch и kbhit)

**ВАДИМ ТУТА ТЫ СДЕЛАЙ**

# 4. Пример запуска программы

На данных изображениях (см. рис. 6-8) показан процесс выполнения программы. При её запуске символ изначально находится по центру, после нажатия клавиши F9 начинается движение влево, после нажатия F10 начинается движение вверх, F11 – вниз, а F12 – вправо. Для программы с собственной реализацией функций getch() и kbhit() примеры работы никак не будут отличаться.

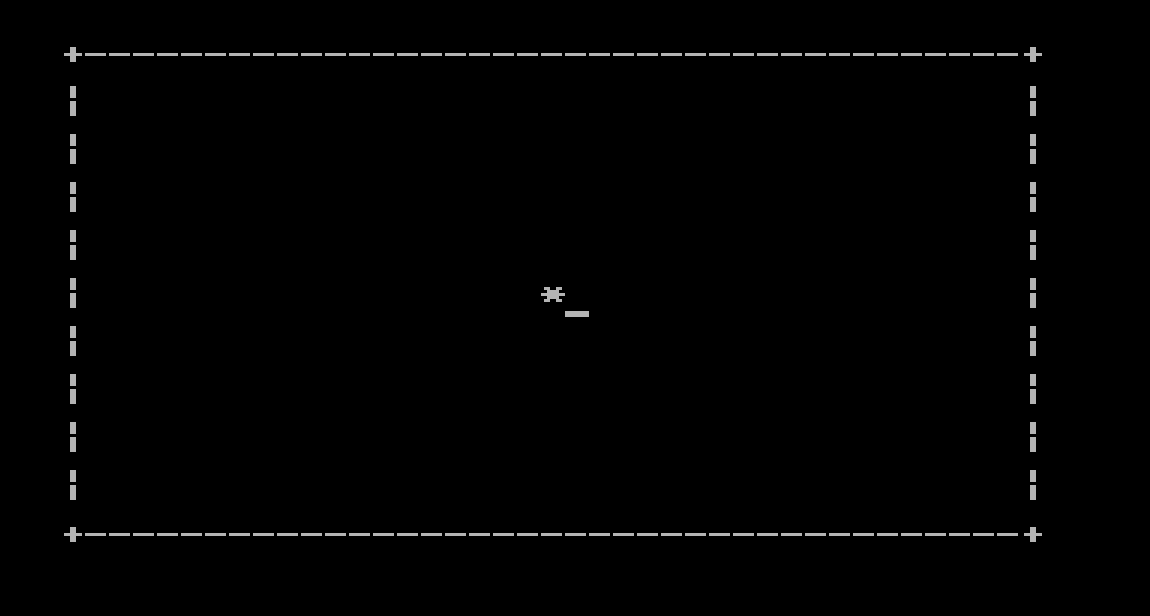


Рисунок 6 – Начальный фон

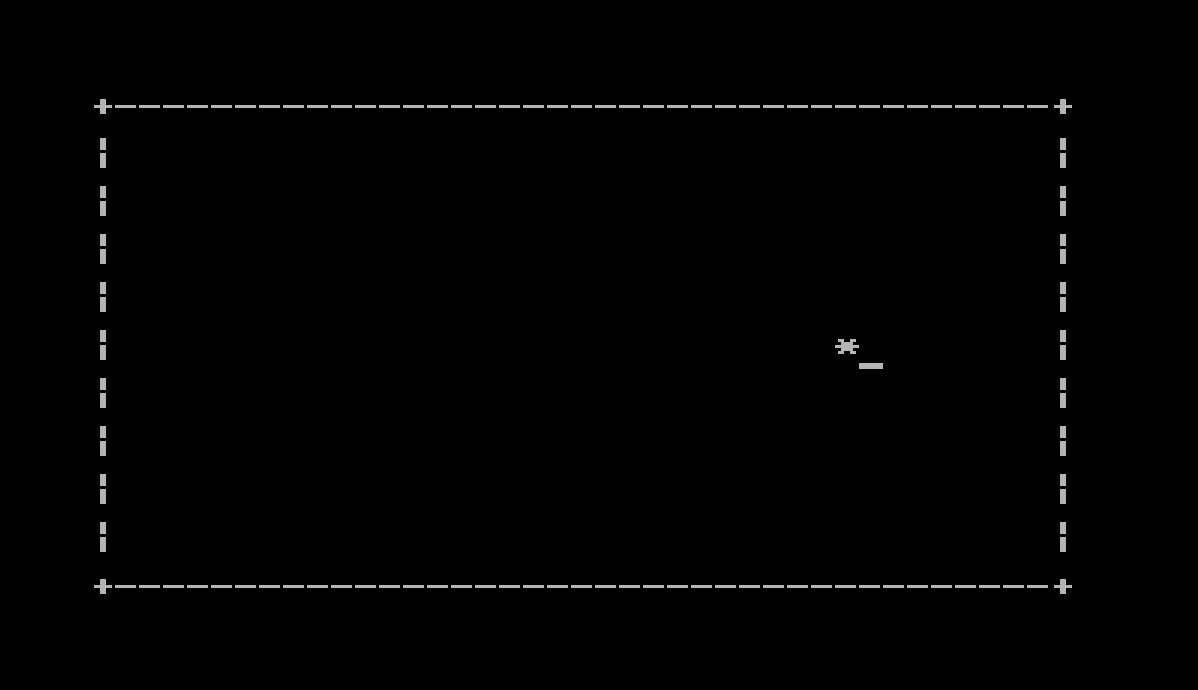


Рисунок 7 – Движение по горизонтали

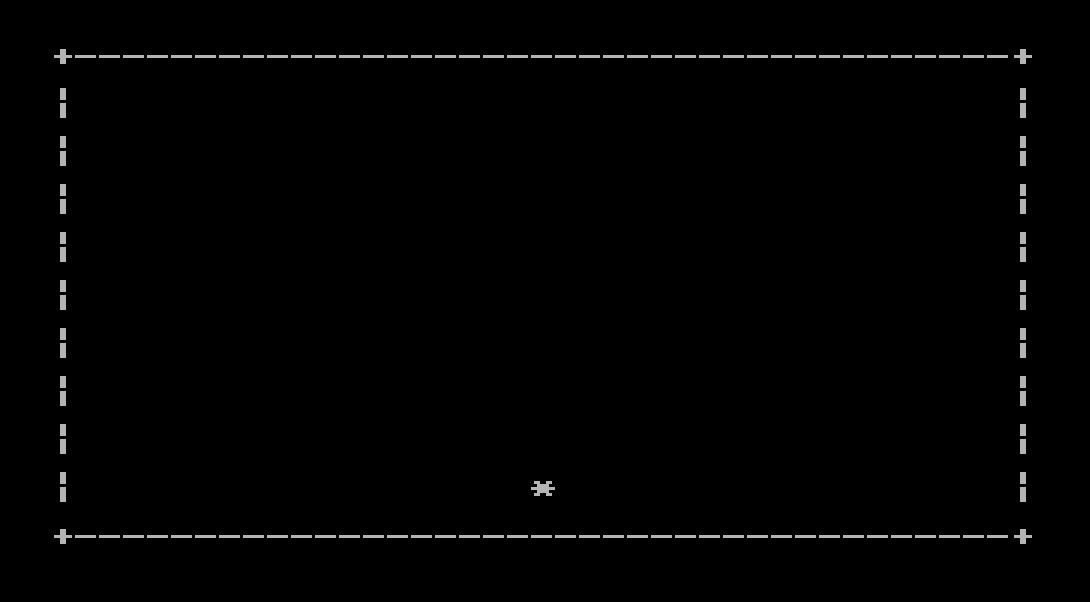


Рисунок 8 – Движение по вертикали

# 5. Выводы

Во время выполнения данной лабораторной работы мы научились использовать библиотеку dos.h, а в частности, во время написания кода, мы подробнее изучили функции getch() и kbhit(). Также ознакомились с видами записи информации с клавиатуры.

# 6. Среда разработки

Turbo C++