**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**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 IBM PC. ИСПОЛЬЗОВАНИЕ ПРЕРЫВАНИЙ»

Студентка гр. 3376 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дегтярева М.И.

Студент гр. 3376 ­­­ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Константинов Р.И.

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ильин С.Е.

Санкт-Петербург

2024

**КЛАВИАТУРА IBM PC. ИСПОЛЬЗОВАНИЕ ПРЕРЫВАНИЙ**

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

**1. Уровни ввода данных с клавиатуры**

Ввод данных с клавиатуры в программировании может быть организован через различные уровни доступа:

* **MS-DOS уровень**: Ввод через MS-DOS позволяет использовать установленные драйверы и стандартные обработчики, такие как прерывание при нажатии Ctrl-C.
* **BIOS уровень**: BIOS обеспечивает прямой доступ к клавиатуре и позволяет отследить все нажатия, включая нестандартные клавиши, через функцию bioskey().
* **Низкий уровень**: Прямой доступ к буферу клавиатуры позволяет существенно повысить производительность и имитировать нажатия клавиш без физического взаимодействия.

**2. Аппаратные компоненты клавиатуры и скэн-коды**

Клавиатура оснащена микропроцессором, который при нажатии клавиш формирует специальный код (скэн-код) и помещает его в порт 60h. Сигнал от клавиатуры обрабатывается через прерывание 9, инициируя BIOS ISR (обработчик прерывания BIOS), который считывает скэн-коды, определяя, была ли клавиша нажата или отпущена. Скэн-коды делятся на прямые и обратные, в зависимости от статуса клавиши.

**3. Типы клавиш и их обработка**

Клавиши можно классифицировать:

* **Шифт-клавиши**: клавиши для модификации ввода (Shift, Alt, Ctrl).
* **Триггерные клавиши**: отвечают за переключение состояния (NumLock, CapsLock).
* **Специальные клавиши**: команды управления, такие как Ctrl-Break. BIOS отслеживает состояния клавиш с помощью битов памяти и управляет буфером с кодами.

**4. Буфер клавиатуры**

Буфер клавиатуры организован в виде кольцевой очереди и рассчитан на хранение кодов до 15 клавиш (30 байт), а для Enter зарезервированы два байта. Указатели "головы" и "хвоста" помогают следить за текущей позицией чтения и записи в буфере. При переполнении буфера звучит звуковой сигнал, а записи приостанавливаются до освобождения буфера.

**5. Ввод через прерывание 16h (BIOS)**

* **AH = 00h:** ожидание нажатия клавиши и возврат ее ASCII-кода или расширенного кода. Функция блокирует выполнение программы, пока пользователь не нажмет клавишу. В случае нажатия ASCII-клавиши в AL возвращается ASCII-код символа, в противном случае — 0, и следующим вызовом функции возвращается расширенный скэн-код клавиши.
* **AH = 01h:** проверка буфера клавиатуры без ожидания. Если буфер не пуст, в AL возвращается код клавиши, иначе флаг нуля ZF устанавливается в 1.
* **AH = 02h:** возвращает статус нажатия шифт-клавиш. С помощью этой функции можно определить, нажаты ли клавиши Shift, Ctrl, Alt, CapsLock, NumLock и ScrollLock. Статус возвращается в регистр AL, где каждый бит отвечает за определенную клавишу.
* **AH = 03h:** позволяет задать частоту повторения и значение задержки для клавиш. В регистре AL задается задержка, а в регистре AH — частота повторения. Данная функция используется для настройки реакции клавиатуры при длительном нажатии на клавишу.

**6. Ввод через прерывание 21h (MS-DOS)**

Функции прерывания MS-DOS предоставляют различные способы работы с клавиатурой:

* **AH=01h**: Ввод с ожиданием и "эхо" на экран.
* **AH=06h**: Ввод-вывод без ожидания.
* **AH=07h**: Ввод с ожиданием без "эхо".
* **AH=0Bh**: Проверка состояния ввода. Функции MS-DOS обеспечивают совместимость с аппаратными особенностями через слои драйверов и BIOS.

**Задание на лабораторную работу:**

Разработать, написать и отладить программу для пошагового управления перемещением символа в пределах окна (25, 8, 55, 18). Для управления использовать клавиши "стрелка вверх" и "стрелка вниз". Для считывания нажатий с клавиатуры использовать прерывание INT 16h.   
Во второй программе изменить ранее написанную программу заменив стандартные функции своими (getch()).

**Алгоритмы и тексты отлаженных программ.**

**Алгоритм программы с использованием стандартных функций**

1. Объявление переменных
2. Установка размеров окна - window(25, 8, 55, 18) и выделение цветом - textbackground(7), textcolor(4);
3. Скрываем курсор - \_setcursortype(\_NOCURSOR);
4. Определение переменных для начальных координат и для координат границы окна
5. Вывод символа в начальной позиции
6. Основной цикл программы для перемещения символа
   1. Ожидание и считывание нажатой клавиши getch()
   2. Проверка был ли нажат escape для выхода из программы
   3. Проверка были ли нажаты клавиши управления символом (стрелки)
      1. Считывание второго байта стрелки getch();
      2. Проверка на стрелку вверх
         1. Проверка на координаты края окна и сдвиг
      3. Проверка на стрелку вниз
         1. Проверка на координаты края окна и сдвиг
   4. Обновление положения символа с новыми координатами
7. Конец программы

**Алгоритм функции show\_symb()**

1. Принимаются параметры координат x и y
2. Очистка экрана
3. Переход на заданные координаты – gotoxy()
4. Вывод символа

**Код программы №1 со стандартными функциями**

#include <stdio.h>

#include <conio.h>

#include <bios.h>

#include <dos.h>

#include <graphics.h>

#define ESCAPE 27

#define UP 72

#define LEFT 75

#define RIGHT 77

#define DOWN 80

void show\_symb(int x, int y)

{

clrscr();

gotoxy(x, y);

printf("@");

}

int main()

{

int x, y, key;

int true;

true = 1;

//(x1, y1, x2, y2)

window(25, 8, 55, 18); //30 on 10

textbackground(7); //gray

textcolor(4);

\_setcursortype(\_NOCURSOR);

//for center of window

y = 5;

x = 15;

//win border

int minY = 1; //with win(!)

int maxY = 11;

show\_symb(x, y);

while(true)

{

key = getch(); //key of pressed key

if (key == ESCAPE)

break;

if (key == 0 || key == 224)

{

key = getch(); //key of arrow

if (key == UP)

if (y > minY)

y--;

else

y = maxY;

if (key == DOWN)

if (y < maxY)

y++;

else

y = minY;

}

show\_symb(x, y);

}

return 0;

}

**Пример запуска программы №1:**

Отображение символа (рисунок 1) и перемещение символа в крайних координатах окна (рисунки 2, 3)

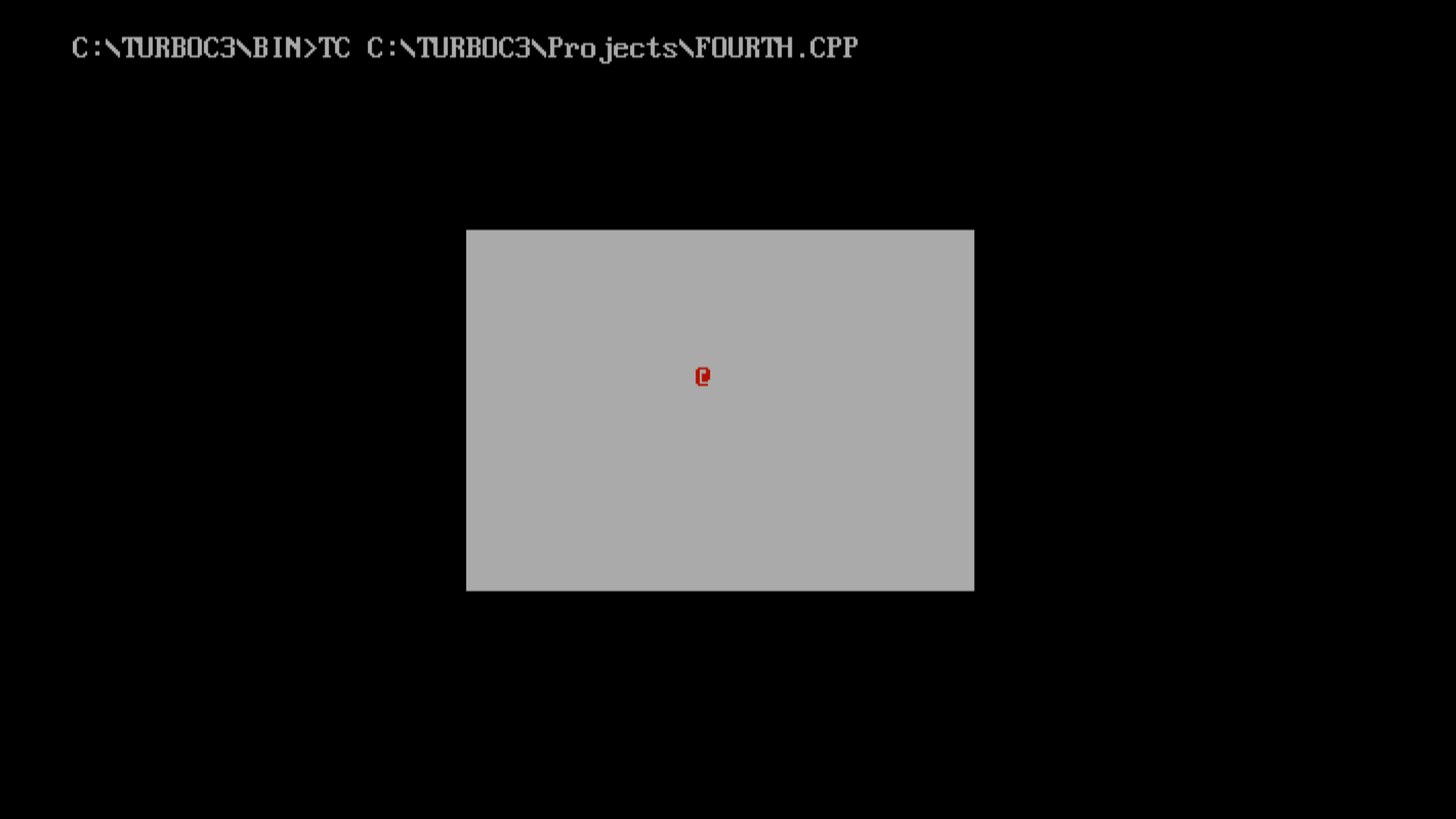


Рисунок 1 – начальный запуск

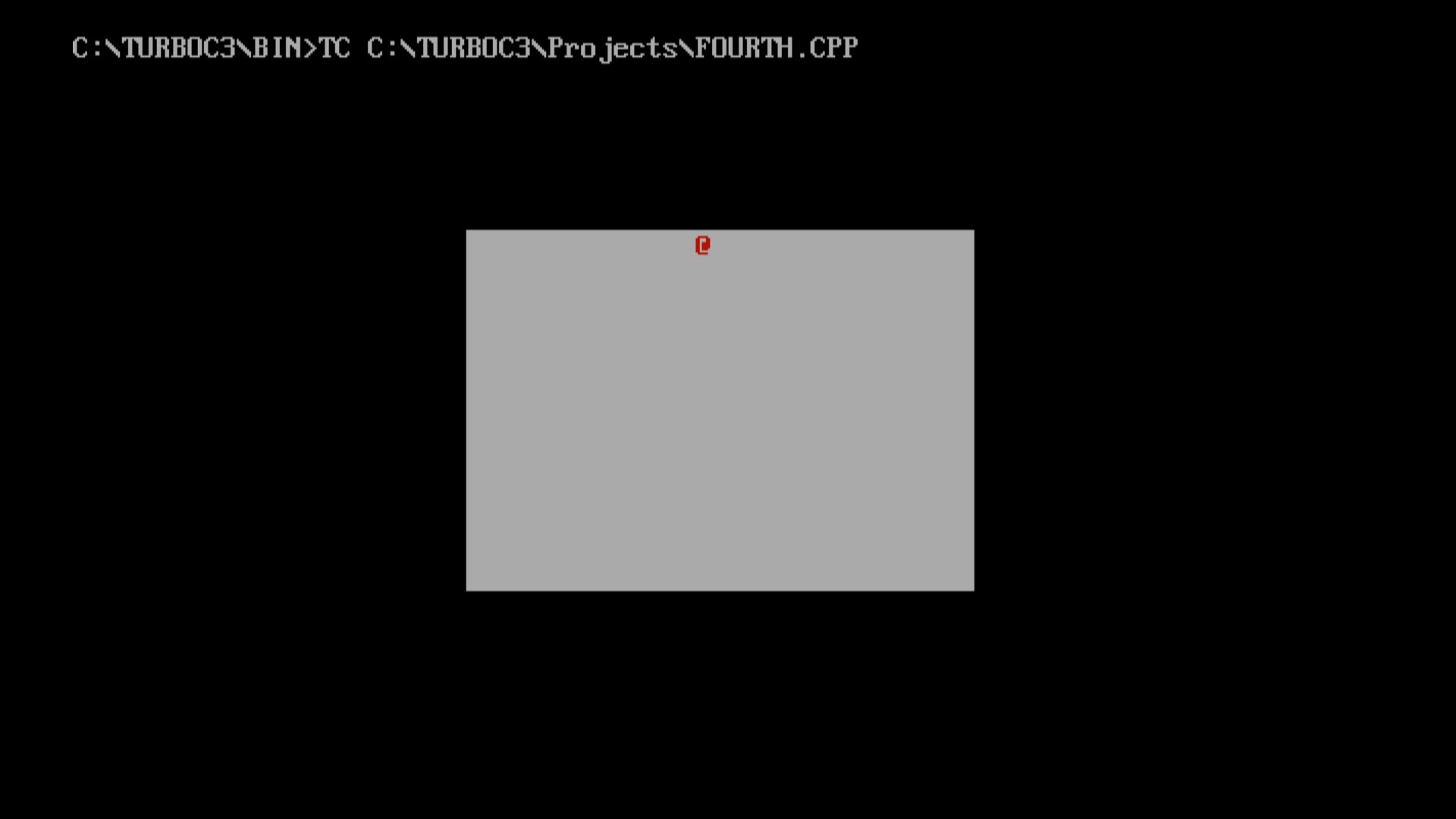


Рисунок 2 – Точка у минимального значения координаты y

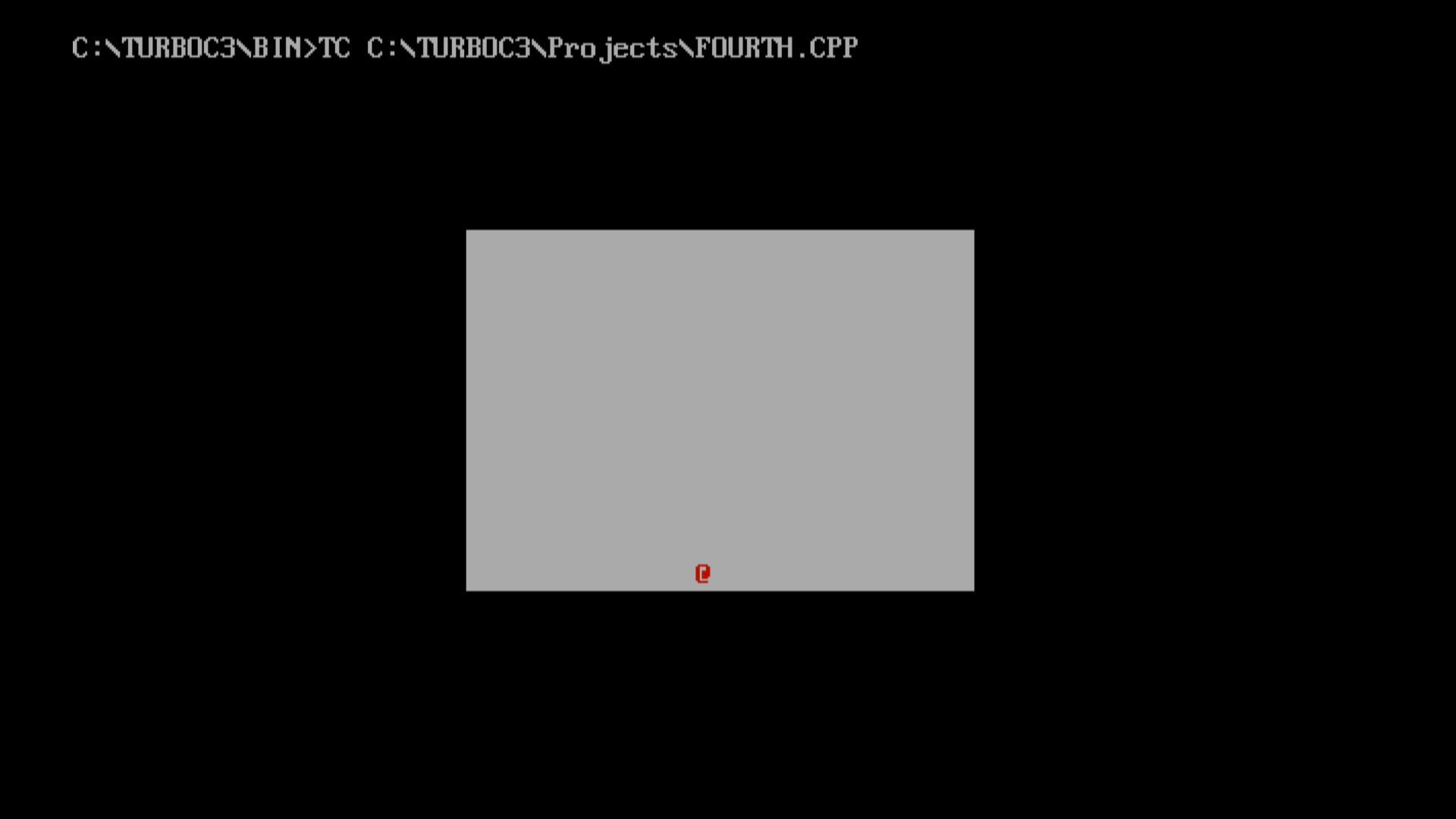


Рисунок 3 – изменение координаты y на максимальную

**Алгоритм программы с использованием собственных функций**

1. Объявление переменных
2. Установка размеров окна - window(25, 8, 55, 18) и выделение цветом - textbackground(7), textcolor(4);
3. Скрываем курсор - \_setcursortype(\_NOCURSOR);
4. Определение переменных для начальных координат и для координат границы окна
5. Вывод символа в начальной позиции
6. Основной цикл программы для перемещения символа
   1. Ожидание и считывание нажатой клавиши \_getch()
   2. Проверка был ли нажат escape для выхода из программы
   3. Проверка на стрелку вверх
      1. Проверка на координаты края окна и сдвиг
   4. Проверка на стрелку вниз
      1. Проверка на координаты края окна и сдвиг
   5. Обновление положения символа с новыми координатами
7. Конец программы

**Алгоритм функции \_getch():**

1. Объявление структуры union REGS r для работы с регистрами процессора
2. Устанавливается regs.h.ah = 0x00, что означает команду ожидания нажатия клавиши через прерывание BIOS.
3. Вызывается функция int86(0x16, &regs, &regs), которая обрабатывает прерывание 0x16, ожидая нажатие клавиши.
4. Проверка кода клавиши
   1. Если в регистре regs.h.al содержится 0 или 224, это означает, что была нажата специальная клавиша (стрелка)
   2. В этом случае возвращается код специальной клавиши из regs.h.ah
5. Если клавиша обычная, возвращается значение regs.h.al, содержащее её ASCII-код.

**Код программы №2 с собственными функциями:**

#include <stdio.h>

#include <conio.h>

#include <dos.h>

#define ESCAPE 27

#define UP 72

#define DOWN 80

int \_getch() // get key

{

union REGS regs;

regs.h.ah = 0x00; // BIOS waiting

int86(0x16, &regs, &regs);

if(regs.h.al == 0 || regs.h.al == 224)

return regs.h.ah;

return regs.h.al; // ASCII-code of key

}

void show\_symb(int x, int y)

{

clrscr();

gotoxy(x, y);

printf("@");

}

int main()

{

int x, y, key;

int true = 1;

//(x1, y1, x2, y2)

window(25, 8, 55, 18);

textbackground(7); //gray

textcolor(4); //red

\_setcursortype(\_NOCURSOR); //hide cursor

//center of window

y = 5;

x = 15;

//win border

int minY = 1;

int maxY = 11;

show\_symb(x, y);

while (true)

{

key = \_getch(); //key of pressed key

if (key == ESCAPE) //if ESC

break;

if (key == UP)

if (y > minY)

y--;

else

y = maxY;

if (key == DOWN)

if (y < maxY)

y++;

else

y = minY;

show\_symb(x, y);

}

return 0;

}

**Пример запуска программы №2:**

Отображение символа (рисунок 4) и перемещение символа в крайних координатах окна (рисунки 5, 6)

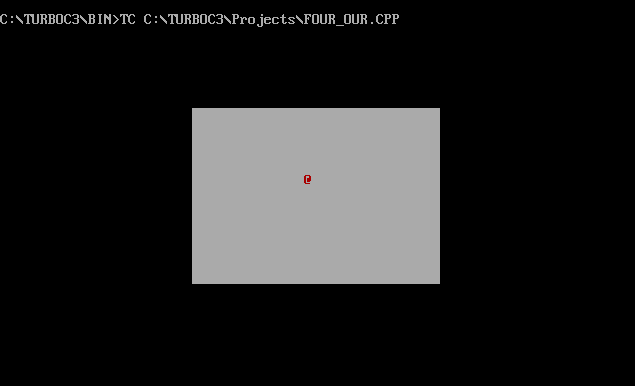


Рисунок 4 – начальный запуск

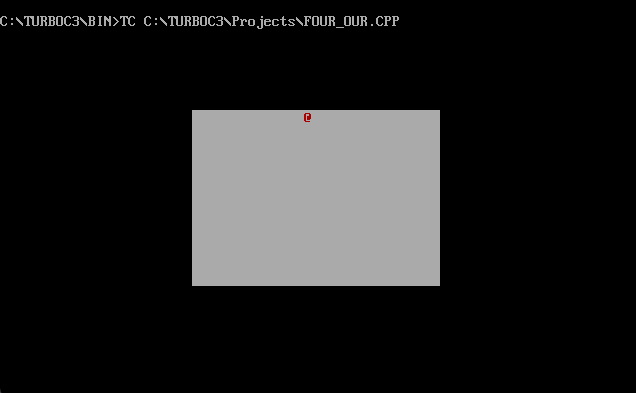


Рисунок 5 – Точка у минимального значения координаты y



Рисунок 6 – изменение координаты y на максимальную

**Вывод.**

Изучили работу с прерываниями, изучили работу прерывания INT 16h, ознакомились со стандартными средствами библиотеки C++ и средствами системы прерываний DOS и BIOS, обслуживающими клавиатуру.

Реализовали собственную функцию аналог getch()- \_getch().