МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра ВТ

Лабораторная работа №5
ИСПОЛЬЗОВАНИЕ АППАРАТНЫХ ПРЕРЫВАНИЙ

Студентка гр. 6306 _____ Пустовойтова А. А.

Преподаватель _____ Бекенева Я. А.

Санкт-Петербург

2017

Краткие сведения

Сведения о подсистеме ввода информации с клавиатуры

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 AH = 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.

Используемые 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. 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 C 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 AH=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 AH=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 AH=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.

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 BIOSом.

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. Если выполняется чтение буфера с разрушением информации (функция AH = 0 прерывания 16h, ISR читает два байта памяти по адресу, смещение которого задает указатель "головы", а сегмент равен 40h. Затем ISR прерывания 16h увеличивает (продвигает) указатель "головы" на 2. Если значения указателей "головы" и "хвоста" равны между собой, буфер пуст. В этом случае ISR прерывания 16 выполняет бесконечный цикл ожидания, условием выхода из которого будет неравенство указателей. При нажатии клавиши, генерирующей двухбайтовый код, этот цикл будет прерван BIOS-обработчиком прерывания 9. В результате указатель "головы" продвинется на 2. После завершения ISR BIOS прерывания 9 возобновится выполнение ISR BIOS прерывания 16h. Так как буфер уже не пуст, произойдет выход из цикла и передача в точку вызова прочитанного в

буфере двухбайтового кода.Если выполняется чтение буфера без разрушения информации (функция AH = 1 прерывания 16h), продвижение указателя "головы" не происходит, но прочитанный по этому указателю код передается в точку вызова, если только буфер не пуст. Если буфер пуст, ISR прерывания 16h не выполняет цикл ожидания. Вместо этого флаг переноса CF устанавливается в 1, и обработчик завершает свою работу.

На рис. 4.1. приведены примеры пустого буфера клавиатуры и буфера после ввода с клавиатуры строки "TEST", нажатия клавиш Left и F1 при условии, что текущая программа не выполняла в этот момент ввод с клавиатуры.

Рис. 4.1. Организация буфера клавиатуры: а - пустой буфер (значения указателей "готовы" и "хвоста" равны); б- тот же буфер после набора на клавиатуре строки TEST, нажатия специальных клавиш Left и F1 (текущая программа не выполняет ввод информации и указатель "головы" не продвигается)

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 AH= 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. Так как производитель постоянно анализирует наличие переполнения буфера, не происходит переопределения не прочитанных еще потребителем кодов клавиатуры. Другими словами, при переполнении буфера производитель блокируется до тех пор, пока потребитель не прочитает одно или несколько слов из буфера. Если же буфер пуст и выполняется попытка чтения информации, функция AH = 00h прерывания 16h BIOS переходит к бесконечному циклу, условием которого является неравенство между собой указателей "головы" и "хвоста". Фактически текущая программа, выполняющая ввод с клавиатуры, блокируется, не давая "потребить" несуществующую еще информацию.

Функции обслуживания

int getch (void)

Выполняет ввод с клавиатуры через функцию MS-DOS AH=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.

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.

char *getpass(char * prompt)

Выводит на экран ASCII-строку, на начало которой указывает prompt, а 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.

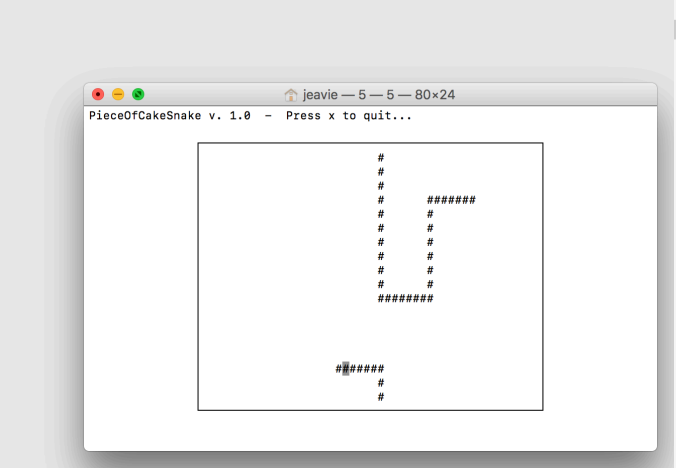
int kbhit (void)

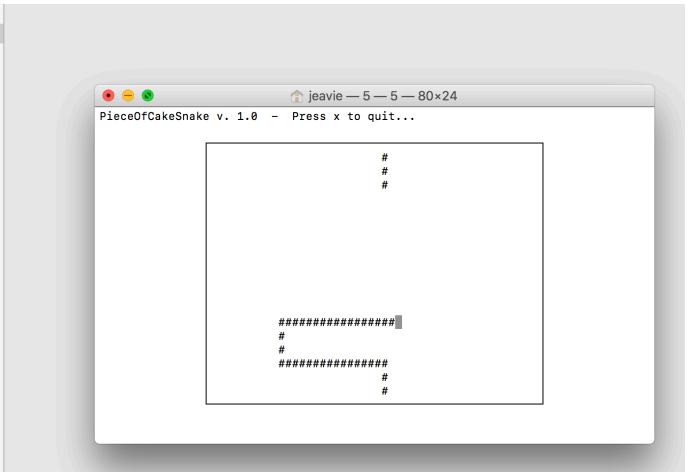
Проверяет, пуст ли буфер клавиатуры. Если в буфере есть символы, функция возвращает ненулевое значение, в противном случае она возвращает 0. Использует функцию 0Bh MS-DOS. Является удобным средством предотвращения "зацикливания" при ожидании невозможного в данный момент события. Кроме того, при выполнении функции 0Bh осуществляется проверка нажатия комбинации клавиш Ctrl-Break, что позволяет выполнить аварийное завершение программы.

Задание

Разработать, написать и отладить программу управления перемещением последовательностью символ наподобие змейки (например, "*") в пределах заданного на экране окна. Для управления использовать клавиши из набора: "стрелка вверх" (СтВВ), "стрелка вниз" (СтВН), "стрелка вправо" (СтВП), "стрелка влево" (СтВЛ) или функциональные клавиши Fl - F12 (варианты см. в таблице 4.2). Для ввода использовать свои функции языка С++. Сохранить отлаженную программу.

Примеры запуска





Текст программы

Main.cpp

```
#include <ncurses.h>

#define TICKRATE 100

#define WORLD_WIDTH 50
#define WORLD_HEIGHT 20

#define SNAKEY_LENGTH 40

enum direction { UP, DOWN, RIGHT, LEFT };

typedef struct spart {
    int x;
    int y;
} snakeypart;

int move_snakey(WINDOW *win, int direction,
                snakeypart snakey[]);

int main(int argc, char *argv[]) {

    WINDOW *snakeys_world;
    int offsetx, offsety, i, ch;

    initscr();
    noecho();
    cbreak();
    timeout(TICKRATE);
    keypad(stdscr, TRUE);

    printw("PieceOfCakeSnake v. 1.0  -  Press x to quit...");
```

```
    refresh();

    offsetx = (COLS - WORLD_WIDTH) / 2;
    offsety = (LINES - WORLD_HEIGHT) / 2;

    snakeys_world = newwin(WORLD_HEIGHT,
                           WORLD_WIDTH,
                           offsety,
                           offsetx);

    snakeypart snakey[SNAKEY_LENGTH];

    int sbegx = (WORLD_WIDTH - SNAKEY_LENGTH) / 2;
    int sbegy = (WORLD_HEIGHT - 1) / 2;

    for (i = 0; i < SNAKEY_LENGTH; i++) {
        snakey[i].x = sbegx + i;
        snakey[i].y = sbegy;
    }

    int cur_dir = RIGHT;

    while ((ch = getch()) != 'x') {
        move_snakey(snakeys_world, cur_dir, snakey);
        if(ch != ERR) {
            switch(ch) {
                case KEY_UP:
                    cur_dir = UP;
                    break;
                case KEY_DOWN:
                    cur_dir = DOWN;
                    break;
                case KEY_RIGHT:
                    cur_dir = RIGHT;
                    break;
                case KEY_LEFT:
                    cur_dir = LEFT;
                    break;
                default:
                    break;
            }

        }
    }

    delwin(snakeys_world);

    endwin();

    return 0;

}

int move_snakey(WINDOW *win, int direction,
                snakeypart snakey[]) {

    wclear(win);

    for (int i = 0; i < SNAKEY_LENGTH - 1; i++) {
```

```
        snakey[i] = snakey[i + 1];
        mvwaddch(win, snakey[i].y, snakey[i].x, '#');
    }

    int x = snakey[SNAKEY_LENGTH - 1].x;
    int y = snakey[SNAKEY_LENGTH - 1].y;
    switch (direction) {
        case UP:
            y - 1 == 0 ? y = WORLD_HEIGHT - 2 : y--;
            break;
        case DOWN:
            y + 1 == WORLD_HEIGHT - 1 ? y = 1 : y++;
            break;
        case RIGHT:
            x + 1 == WORLD_WIDTH - 1 ? x = 1 : x++;
            break;
        case LEFT:
            x - 1 == 0 ? x = WORLD_WIDTH - 2 : x--;
            break;
        default:
            break;
    }

    snakey[SNAKEY_LENGTH - 1].x = x;
    snakey[SNAKEY_LENGTH - 1].y = y;

    mvwaddch(win, y, x, '#');

    box(win, 0 , 0);

    wrefresh(win);

    return 0;
}
```