**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 IBM PC. ИСПОЛЬЗОВАНИЕ ПРЕРЫВАНИЙ**

|  |  |  |
| --- | --- | --- |
| Студенты гр. 3353 |  | Карпенко А.Ю. |
| Преподаватель |  | Гречухин М.Н. |

Санкт-Петербург

2024

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

**Задание:**

1. Разработать, написать и отладить программу управления перемещением символа (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 стандартные функции языка C++.
2. Изменить программу, заменив стандартные функции библиотеки C++ своими. Для написания функций используйте заданное прерывание (см. таблицу), если его возможностей достаточно. Если его возможностей недостаточно, то замените его по своему усмотрению. Сохраните отлаженную программу.

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

**Подсистема ввода информации с клавиатуры**

Большинство программ, работающих в текстовом режиме, используют клавиатуру как основной источник ввода данных. Ввод информации с клавиатуры может быть выполнен на трех уровнях:

1. **Через функции MS-DOS** – обеспечивает обработку вводимых символов через драйверы и отслеживание комбинаций клавиш, таких как **Ctrl-C** (прерывание программы).
2. **Через функции BIOS** – позволяет получать доступ к расширенным возможностям клавиатуры и отслеживать не только символьные, но и управляющие клавиши (стрелки, функциональные клавиши и т.д.).
3. **Прямой доступ к аппаратным средствам** – работа с буфером клавиатуры напрямую, что существенно ускоряет обработку ввода.

**Ввод на уровне MS-DOS**

Ввод с клавиатуры на уровне **MS-DOS** проходит через драйверы, что позволяет отслеживать специальные комбинации клавиш и обеспечивает обработку ошибок ввода. Например, функции **getch()** и **getche()** используют прерывание **07h** для ввода данных с клавиатуры:

* **getch()** – выполняет ввод символа с клавиатуры без вывода его на экран (без "эхо").
* **getche()** – выполняет ввод символа с выводом на экран (с "эхо").

Функция \**getpass(char prompt)* позволяет вводить пароли, скрывая введенные символы, что важно для безопасности.

**Ввод на уровне BIOS**

Доступ к клавиатуре через BIOS позволяет работать с буфером клавиатуры и расширенными клавишами, такими как функциональные клавиши, стрелки и другие управляющие клавиши. В языке **C++** это реализуется через функцию **bioskey()**, которая взаимодействует с BIOS для получения нажатий всех типов клавиш.

BIOS работает через прерывание **16h**, предоставляя такие функции:

* **AH=00h** – получить символ с клавиатуры, удалив его из буфера.
* **AH=01h** – проверить, есть ли символ в буфере, не удаляя его.
* **AH=02h** – получить статус клавиш-модификаторов (Shift, Ctrl, Alt).
* **AH=03h** – получить код расширенных клавиш (стрелки, функциональные клавиши и т.д.).

**Буфер клавиатуры**

**Буфер клавиатуры** – это кольцевой буфер, используемый для хранения скан-кодов нажатых клавиш. Когда клавиша нажимается, ее **скан-код** передается через порт **60h** в ППИ (программируемый периферийный интерфейс), который генерирует аппаратное прерывание **09h**. После этого BIOS обрабатывает скан-код и помещает его в буфер клавиатуры.

**Скан-код** включает информацию о нажатии (или отпускании) клавиши. Например:

* Прямой скан-код (восьмой бит равен 0) означает, что клавиша нажата.
* Обратный скан-код (восьмой бит равен 1) означает, что клавиша отпущена.

Буфер используется для асинхронного взаимодействия: процессор (производитель) записывает данные о нажатых клавишах, а программа (потребитель) читает их, когда готова к обработке. Это позволяет избежать потери данных при быстром вводе.

**Имитация нажатий клавиш**

Некоторые программы имитируют нажатия клавиш, записывая скан-коды напрямую в буфер клавиатуры. Это используется в демонстрационных программах, автоматизирующих работу пользователя, или в резидентных программах, таких как **THELP.COM**, которые могут вставлять текст в другие приложения.

**Функции для работы с вводом**

1. **getch()** – ввод символа без его отображения на экране.
2. **getche()** – ввод символа с его отображением.
3. **getpass()** – ввод строки без отображения символов (используется для ввода паролей).
4. **kbhit()** – проверяет, есть ли символы в буфере клавиатуры, что полезно для предотвращения зацикливания программы при ожидании ввода.

**Преимущества и недостатки уровней ввода**

* **MS-DOS** ввод проще в использовании, но медленнее и имеет ограниченные возможности.
* **BIOS** ввод предоставляет больше возможностей для работы с клавиатурой, включая доступ к аппаратному уровню, но медленнее по сравнению с прямым доступом к буферу.
* **Прямой доступ к аппаратным средствам** – самый быстрый способ, позволяющий более эффективно управлять вводом и имитировать нажатия клавиш.

**Текст программы**

1. С использованием стандартных функций:

#include <stdio.h>

#include <conio.h>

#include <bios.h>

#include <dos.h>

#define ESCAPE 27

#define UP 72

#define DOWN 80

#define LEFT 75

#define RIGHT 77

#define X1 15

#define X2 65

#define Y1 10

#define Y2 20

int main()

{

int x, y, ch;

x = y = 1;

window(X1, Y1, X2, Y2);

textbackground(7);

textcolor(4);

\_setcursortype(\_NOCURSOR);

gotoxy(x, y);

clrscr();

printf("\*");

gotoxy(x, y);

do {

ch = getch();

if (ch == 0 || ch == 224) {

switch (getch()) {

case DOWN:

clrscr();

if (y <= (Y2 - Y1)) {

y++;

gotoxy(x, y);

printf("\*");

gotoxy(x, y);

}

else {

y = 1;

gotoxy(x, y);

printf("\*");

gotoxy(x, y);

}

break;

case UP:

clrscr();

if (y - 1 >= 1) {

y--;

gotoxy(x, y);

printf("\*");

gotoxy(x, y);

}

else {

y = Y2 - Y1 + 1;

gotoxy(x, y);

printf("\*");

gotoxy(x, y);

}

break;

case LEFT:

clrscr();

if (x - 1 >= 1) {

x--;

gotoxy(x, y);

printf("\*");

gotoxy(x, y);

}

else {

x = X2 - X1 + 1;

gotoxy(x, y);

printf("\*");

gotoxy(x, y);

}

break;

case RIGHT:

clrscr();

if (x + 1 <= (X2 - X1) + 1) {

x++;

gotoxy(x, y);

printf("\*");

gotoxy(x, y);

}

else {

x = 1;

gotoxy(x, y);

printf("\*");

gotoxy(x, y);

}

break;

}

}

} while (ch != ESCAPE);

return 0;

}

1. С использованием своих функций:

#include <stdio.h>

#include <conio.h>

#include <bios.h>

#include <dos.h>

#define ESCAPE 27 //объявление константы ESCAPE, которая соответствует ASCII-коду клавиши "Esc" (27).

#define UP 72

#define DOWN 80

#define LEFT 75

#define RIGHT 77

#define X1 15

#define X2 65

#define Y1 10

#define Y2 20

int keypressed(){ // проверяет, была ли нажата клавиша.

union REGS r;

r.h.ah = 0x0B; // запись значения 0x0B (функция DOS, которая проверяет ввод с клавиатуры) в старший байт регистра ah.

int86(0x21, &r, &r); //вызов DOS-прерывания 0x21, которое обрабатывает системные функции.

return r.h.al; // возврат значения регистра al (младшего байта), которое указывает, была ли нажата клавиша.

//Если клавиша была нажата, это значение будет ненулевым.

}

int getKey(){ //возвращает целое значение (код нажатой клавиши).

union REGS r;

r.h.ah = 0x7;

/\*установка значения регистра ah на 0x7.

Это код функции BIOS для "ввода символа с клавиатуры" (без эха на экран),

которая ожидает нажатие клавиши и возвращает её код.

Эхо — это автоматическое отображение нажатой клавиши в консоль.\*/

int86(0x16, &r, &r); //прерывание 0x16 для работы с клавиатурой.

return r.h.al; //возврат значения из регистра al, который содержит код нажатой клавиши.

}

int main(){

int x, y, ch; //ch — переменная для хранения кода нажатой клавиши.

x = y = 1;

window(X1, Y1, X2, Y2); //размеры окна

textbackground(7); // установка фона текста

textcolor(4); //установка цвета текста

\_setcursortype(\_NOCURSOR); //отключение видимости курсора

gotoxy(x, y); //перемещение курсора в координаты (x, y).

clrscr(); //очистка экрана.

printf("\*");

gotoxy(x, y);

do {

ch = getKey(); //Программа ожидает нажатие клавиши, считывая её код через функцию getKey()

if (ch == 0 || ch == 224) { //проверка на то, была ли нажата функциональная клавиша (клавиши-стрелки возвращают такие коды).

switch (getKey()) {

case DOWN:

clrscr();

if (y <= (Y2 - Y1)) {

y++;

gotoxy(x, y);

printf("\*");

gotoxy(x, y);

}

else {

y = 1;

gotoxy(x, y);

printf("\*");

gotoxy(x, y);

}

break;

case UP:

clrscr();

if (y - 1 >= 1) {

y--;

gotoxy(x, y);

printf("\*");

gotoxy(x, y);

}

else {

y = Y2 - Y1 + 1;

gotoxy(x, y);

printf("\*");

gotoxy(x, y);

}

break;

case LEFT:

clrscr();

if (x - 1 >= 1) {

x--;

gotoxy(x, y);

printf("\*");

gotoxy(x, y);

}

else {

x = X2 - X1 + 1;

gotoxy(x, y);

printf("\*");

gotoxy(x, y);

}

break;

case RIGHT:

clrscr();

if (x + 1 <= (X2 - X1) + 1) {

x++;

gotoxy(x, y);

printf("\*");

gotoxy(x, y);

}

else {

x = 1;

gotoxy(x, y);

printf("\*");

gotoxy(x, y);

}

break;

}

}

} while (ch != ESCAPE); // программа продолжает работать, пока не будет нажата клавиша Esc

return 0;

}}

**Выводы**

В ходе лабораторной работы мы разработали программу управления перемещением символа «\*» в пределах заданного на экране окна. Для управления использовали клавиши: «стрелка влево», «стрелка вверх», «стрелка вправо», «стрелка вниз». Далее изменили программу, заменив стандартные функции библиотеки С++ своими.