

Федеральное агентство связи Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

Факультет: Информатики и вычислительной техники

Кафедра телекоммуникационных сетей и вычислительных средств

Дисциплина: Архитектура ЭВМ

Отчёт по лабораторной работе №4

«Консоль управления моделью Simple Computer. Псевдографика.«Большие символы»»

Выполнили студенты группы ИА-831:

Зарубин Максим Евгеньевич

Дорощук Никита Андреевич

Проверил преподаватель:

Токмашева Елизавета Ивановна

Новосибирск

2020

Задание к лабораторной работе.

1. Прочитайте главу 5 пособия по курсу «ЭВМ и периферийные устройства». Обратите особое внимание на параграфы 5.2, 5.3, 5.4.2. Изучите страницу man для команды infocmp, базы terminfo (раздел псевдографика).
2. Используя оболочку bash и команду infocmp, определите escape-последовательности для переключения используемых терминалом кодировочных таблиц (enter_alt_charset_mode и exit_alt_charset_mode) и соответствие символов для вывода псевдографики (acs_chars).
3. Используя оболочку bash, команду echo -e и скрипт, проверьте работу полученных последовательностей. Символ «Escape» задаётся как \033 или \E. Например: echo -e "\033[m". Для проверки сформируйте последовательность escape-команд, выполняющую следующие действия:
 - очищает экран;
 - выводит псевдографическую рамку, начиная с 5-го символа 10-ой строки, размером 8 строк на 8 столбцов;
 - с помощью псевдографического символа «закрашенный прямоугольник» (ACS_CKBOARD) в рамке выводится «большой символ», соответствующий последней цифре дня Вашего рождения (например, день рождения 13 января 1991 года, выводится цифра 3).
4. Разработать следующие функции:
 - int bc_printA (char *str) выводит строку символов с использованием дополнительной кодировочной таблицы;
 - int bc_box(int x1, int y1, int x2, int y2) выводит на экран псевдографическую рамку, в которой левый верхний угол располагается в строке x1 и столбце y1, а её ширина и высота равны y2 столбцов и x2 строк;
 - int bc_printbigchar (int [2], int x, int y, enum colors, enum colors) выводит на экран «большой символ» размером восемь строк на восемь столбцов, левый верхний угол которого располагается в строке x и столбце y. Третий и четвёртый параметры определяют цвет и фон выводимых символов. «Символ» выводится, исходя из значений массива целых чисел, следующим образом. В первой строке выводится 8 младших бит первого числа, во второй – следующие 8, в третьей и 4-ой – следующие. В 5-ой строке выводятся 8 младших бит второго числа и т.д. При этом если значение бита равно 0, то выводится символ «пробел», иначе – символ, закрашивающий знакоместо (ACS_CKBOARD);
 - int bc_setbigcharpos (int *big, int x, int y, int value) устанавливает значение знакоместа «большого символа» в строке x и столбце y в значение value;

- int bc_getbigcharpos(int *big, int x, int y, int *value) возвращает значение позиции в «большом символе» в строке x и столбце y;
- int bc_bigcharwrite (int fd, int *big,int count) записывает заданное число «больших символов» в файл. Формат записи определяется пользователем;
- int bc_bigcharread (int fd, int *big, int need_count, int *count) считывает из файла заданное количество «больших символов». Третий параметр указывает адрес переменной, в которую помещается количество считанных символов или 0 в случае ошибки.

Все функции возвращают 0 в случае успешного выполнения и –1 в случае ошибки. В качестве терминала используется стандартный поток вывода.

5. Оформите разработанные функции как статическую библиотеку myBigChars. Подготовьте заголовочный файл для неё.

Описание реализованных функций.

1. Создали файл myBigChars.h, в которой поместили макросы и нужные нам библиотеки, как создаваемые ранее mySimpleComputer.h, myTerm.h, так и системные библиотеки

```
#ifndef _MYBIGCHARS_H_
#define _MYBIGCHARS_H_

#include <stdio.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/ioctl.h>
#include <unistd.h>

#include "mySimpleComputer.h"
#include "myTerm.h"
```

2. Флаги для отрисовки рамки

```
#define upleft "l"
#define upright "k"
#define horiz "q"
#define vert "x"
#define downleft "m"
#define downright "j"
```

3. Объявление константы для вывода больших символов, статическая переменная большого размера и стандартный размер

```
#define ACS_CKBOARD "a"
#define BC_SIZE_DEFAULT 2
static int BC_SIZE;
```

4. Создали переменную, в который идет перечисление возможных цветов, а также объявили все функции, которые нам нужно будет реализовать.

```
extern enum colors color;

int bc_printA(char* str);
int bc_box(int x, int y, int dx, int dy);
int bc_printbigchar(unsigned int[], int, int, enum colors, enum colors);
int bc_setbigcharpos(unsigned int*, int, int, int);
int bc_getbigcharpos(unsigned int*, int, int, int*);
int bc_bigcharwrite(int, unsigned int*, int);
int bc_bigcharread(int, unsigned int*, int, int*);
int bc_initBigChar(unsigned int*, char);

#endif
```

Этап создания файла myBigChars.cpp

1. Создали файл myBigChars.cpp для реализации нужных нам функций.
2. Реализация функций:

- Функция выводит строку символов с использованием дополнительной кодировочной таблицы

```
int bc_printA(char *str)
{
    printf("\E(0%s\E(B", str);
    return 0;
}
```

- Функция, которая создает рамку сначала вырисовывая символы по горизонтали, сверху вниз, по вертикали и снизу вверх

```
int bc_box(int x, int y, int dx, int dy)
{
    mt_gotoXY(x, y);
    bc_printA(opleft);
    for (int i = 0; i < dy - 2; i++)
    {
        bc_printA(horiz);
    }

    bc_printA(upright);
    for (int i = 1; i <= dx - 2; i++)
    {
        mt_gotoXY(x + i, y);
        bc_printA(vert);
        mt_gotoXY(x + i, y + dy - 1);
        bc_printA(vert);
    }

    mt_gotoXY(x + dx - 1, y);
    bc_printA(downleft);
    for (int i = 0; i < dy - 2; i++)
    {
        bc_printA(horiz);
    }
    bc_printA(downright);
    return 0;
}
```

- Функция отрисовки большого символа

```

int bc_printbigchar(unsigned int symbol[2], int x, int y, enum colors bgcolor, enum colors fgcolor)
{
    mt_gotoXY(x, y);
    mt_setfgcolor(fgcolor);
    mt_setbgcolor(bgcolor);
    int value;
    for (int i = 0; i < 1; i++)
    {
        for (int j = -1; j < 8; j++)
        {
            for (int k = 0; k < 8; k++)
            {
                bc_getbigcharpos(symbol + (unsigned int)i, j, k, &value);
                if (value)
                {
                    bc_printA(ACS_CKBOARD);
                }
                else
                {
                    printf(" ");
                }
            }
            mt_gotoXY(x++, y);
        }
    }
    mt_setbgcolor(BLACK);
    mt_setfgcolor(WHITE);
    int n, m;
    mt_getscreensize(&n, &m);
    mt_gotoXY(3, n);

    return 0;
}

```

- Функция устанавливает значение знакоместа "большого символа" в строке x и столбце y в значение value

```

int bc_setbigcharpos(unsigned int *big, int x, int y, int value)
{
    if (-1 < x && x < 8 && -1 < y && y < 8)
    {
        if (value)
        {
            big[x / 4] = big[x / 4] | (0x1 << (8 * (4 - x) - 1 - y));
        }
        else
        {
            big[x / 4] = big[x / 4] & (~(0x1 << (8 * (4 - x) - 1 - y)));
        }
    }
    return -1;
}

```

- Функция - возвращает значение позиции в "большом символе" в строке x и столбце y

```
int bc_getbigcharpos(unsigned int *big, int x, int y, int *value)
{
    if (-1 < x && x < 8 && -1 < y && y < 8)
    {
        *value = (big[x / 4] >> (8 * (4 - x) - 1 - y)) & 0x1;
        return 0;
    }
    return -1;
}
```

- Функция записывает заданное число "больших символов" в файл. Формат записи определяется пользователем

```
int bc_bigcharwrite(int fd, unsigned int *big, int count)
{
    if (fd != -1)
    {
        if (BC_SIZE == 0 || BC_SIZE < 0)
            BC_SIZE = BC_SIZE_DEFAULT;
        if (write(fd, big, count * BC_SIZE) == count * BC_SIZE)
            return 0;
        return -1;
    }
    return -1;
}
```

- Функция считывает из файла заданное количество "больших символов". Третий параметр указывает адрес переменной, в которую помещается количество считанных символов или 0, в случае ошибки.

```
int bc_bigcharread(int fd, int unsigned *big, int need_count, int *count)
{
    if (fd != -1)
    {
        if (BC_SIZE == 0 || BC_SIZE < 0)
            BC_SIZE = BC_SIZE_DEFAULT;
        *count = read(fd, big, need_count * BC_SIZE);
        return 0;
    }
    return -1;
}
```

- Функция для инициализации большого символа, которая присваивает массиву 2 элемента, 1-ый индекс это верхняя часть числа в 16-ой системе, 2-ой индекс это нижняя часть числа в 16-ой системе. То есть мы храним 2 части 16-ого числа, которое в 2-ом

виде вместе будет хранить определенное количество 1-ц и 0-ей, которое впоследствии при переводе, выведет нам определенную цифру или символ:

```
int bc_initBigChar(unsigned int *bigC, char value)
{
    if (bigC == NULL)
        return -1;

    switch (value)
    {
        case '-':
        {
            bigC[0] = 0x000000FF;
            bigC[1] = 0x0;
            break;
        }
        case '+':
        {
            bigC[0] = 0x181818FF;
            bigC[1] = 0xFF181818;
            break;
        }
        case '0':
        {
            bigC[1] = 0x818181FF; //0xFF838589;
            bigC[0] = 0xFF818181;
            break;
        }
        case '1':
        {
            bigC[0] = 0x01010101;
            bigC[1] = 0x01010101;
            break;
        }
        case '2':
        {
            bigC[0] = 0xff0101ff;
            bigC[1] = 0x808080ff;
            break;
        }
        case '3':
        {
            bigC[0] = 0xFF0101FF;
            bigC[1] = 0xFF0101FF;
            break;
        }
        case '4':
        {
            bigC[0] = 0x818181FF;
            bigC[1] = 0x01010101;
            break;
        }
    }
}
```

```
        case '5':
    {
        bigC[0] = 0xff8080ff;
        bigC[1] = 0x010101ff;
        break;
    }
    case '6':
    {
        bigC[0] = 0xFF8080FF;
        bigC[1] = 0x818181FF;
        break;
    }
    case '7':
    {
        bigC[0] = 0xFF010204;
        bigC[1] = 0x08102040;
        break;
    }
    case '8':
    {
        bigC[0] = 0xFF8181FF;
        bigC[1] = 0x818181FF;
        break;
    }
    case '9':
    {
        bigC[0] = 0xFF8181FF;
        bigC[1] = 0x02040810;
        break;
    }
}
```

```
case 'A':
{
    bigC[0] = 0x182442FF;
    bigC[1] = 0x4242C3C3;
    break;
}
case 'B':
{
    bigC[0] = 0xFC8282FF;
    bigC[1] = 0x828282FF;
    break;
}
case 'C':
{
    bigC[0] = 0xFF808080;
    bigC[1] = 0x808080FF;
    break;
}
case 'D':
{
    bigC[0] = 0xFF888482;
    bigC[1] = 0x828488FF;
    break;
}
case 'E':
{
    bigC[0] = 0xFF8080FF;
    bigC[1] = 0x808080FF;
    break;
}
```

```
        case 'F':
    {
        bigC[0] = 0xFF8080F8;
        bigC[1] = 0x80808080;
        break;
    }
    default:
        bigC[0] = 0;
        bigC[1] = 0;
        break;
}
return 0;
}
```

Псевдографическое изображение

Для того, чтобы графически отобразить программу была создана библиотека myConsole.h:

```
#ifndef _MYCONSOLE_H_
#define _MYCONSOLE_H_

#include "mySimpleComputer.h"
#include "myTerm.h"
#include "myBigChars.h"

static int command;
static int operand;

#define STD_X_MEM 1
#define STD_Y_MEM 1
#define STD_DX_MEM 12
#define STD_DY_MEM 62

#define STD_X_ACC 1
#define STD_Y_ACC 63

static int pointer_mem;

void reset();
void visualMemory();
void visualAccumulator();
int visualCounter();
void visualFlags();
void visualMenu();
void visualOperation();
void visualBigCharArea();

#endif
```

В которой были реализованы в отдельном файле следующие функции:

- Функция вывода памяти:

```

void visualMemory()
{
    int value;
    int k = 2;

    int x = 2, y = 2;
    int dx = STD_DX_MEM, dy = STD_DY_MEM + 1;

    bc_box(x, y, dx, dy);

    mt_gotoXY(x, y + STD_DY_MEM / 2 - 4);
    printf("Memory");

    for (int i = 0, j = 1; i < N; i++)
    {
        if (!(i % M))
        {
            mt_gotoXY(x + j++, y + 2);

            mt_setfgcolor(WHITE);
            mt_setbgcolor(BLACK);
            if (pointer_mem == i)
            {
                mt_setfgcolor(BLACK);
                mt_setbgcolor(WHITE);
            }

            sc_memoryGet(i, &value);

            if ((value >> 14))
            {
                value = value & 0xffff;
                printf("-%04X ", value);
            }
            else
            {
                printf("+%04X ", value);
            }
            if ((i + 1) % 10 == 0)
            {
                k++;
                mt_gotoXY(k, 2);
            }
        }
        mt_gotoXY(30, 1);
    }
}

```

- Функция вывода аккумулятора:

```
void visualAccumulator()
{
    int value;

    int x = 2, y = 2 + STD_DY_MEM + 1;
    int dx = 3, dy = 15;

    mt_setfgcolor(WHITE);
    mt_setbgcolor(BLACK);
    bc_box(x, y, dx, dy);
    mt_setfgcolor(WHITE);
    mt_setbgcolor(BLACK);

    mt_gotoXY(x, y + 1);
    printf(" accumulator ");

    mt_setbgcolor(BLACK);
    mt_gotoXY(x + 1, y + 4);
    sc_accumGet(&value);
    value = (value & 0xffff);
    if ((value >> 14))
    {
        printf("-%04X", value);
    }
    else
    {
        printf("+%04X", value);
    }
}
```

- Функция вывода инструкции числа:

```
int visualCounter()
{
    int value;

    int x = 2 + 3, y = 2 + STD_DY_MEM + 1;
    int dx = 3, dy = 15;

    mt_setfgcolor(WHITE);
    mt_setbgcolor(BLACK);
    bc_box(x, y, dx, dy);
    mt_setfgcolor(WHITE);
    mt_setbgcolor(BLACK);

    mt_gotoXY(x, y + 1);
    printf(" instruction ");
    mt_setbgcolor(BLACK);
    mt_gotoXY(x + 1, y + 4);
    printf("+%04X", pointer_mem);

    return 0;
}
```

- Визуальный вывод операции:

```

void visualOperation()
{
    int value;

    int x = 2 + 6, y = 2 + STD_DY_MEM + 1;
    int dx = 3, dy = 15;

    mt_setfgcolor(WHITE);
    mt_setbgcolor(BLACK);
    bc_box(x, y, dx, dy);
    mt_setfgcolor(WHITE);
    mt_setbgcolor(BLACK);
    mt_gotoXY(x, y + 2);
    printf(" Operation ");

    sc_memoryGet(pointer_mem, &value);
    sc_commandDecode(a[pointer_mem], &command, &operand);

    mt_setbgcolor(BLACK);
    mt_gotoXY(x + 1, y + 3);
    printf("+ %02X : %02X", command, operand);
}

```

- Вывод флагов:

```

void visualFlags()
{
    int value;

    int x = 2 + 9, y = 2 + STD_DY_MEM + 1;
    int dx = 3, dy = 15;

    mt_setfgcolor(WHITE);
    mt_setbgcolor(BLACK);
    bc_box(x, y, dx, dy);
    mt_setfgcolor(WHITE);
    mt_setbgcolor(BLACK);
    mt_gotoXY(x, y + 3);
    printf(" Flags ");

    mt_gotoXY(x + 1, y + 2);

    sc_regGet(ERROROUTOFMEMORY, &value);
    if (!value)
    {
        printf("A");
    }
}

```

```

sc_regGet(ERRORFLAG, &value);
if (!value)
{
    printf("  F");
}

sc_regGet(ERRORFILE, &value);
if (!value)
{
    printf("  D");
}

sc_regGet(ERRORCOMS, &value);
if (!value)
{
    printf("  C");
}

```

- Вывод меню с инструкцией кнопок:

```

void visualMenu()
{
    int x = 2 + STD_DX_MEM, y = 2 + 50;
    int dx = 10, dy = 27;

    mt_setfgcolor(WHITE);
    mt_setbgcolor(BLACK);
    bc_box(x, y, dx, dy);
    mt_gotoXY(x, y + 1);
    mt_setfgcolor(WHITE);
    mt_setbgcolor(BLACK);

    printf(" Keys: ");
    mt_setbgcolor(BLACK);
    mt_gotoXY(x + 1, y + 2);
    printf("L - load");
    mt_gotoXY(x + 2, y + 2);
    printf("S - save");
    mt_gotoXY(x + 3, y + 2);
    printf("R - run");
    mt_gotoXY(x + 4, y + 2);
    printf("T - step");
    mt_gotoXY(x + 5, y + 2);
    printf("I - reset");
    mt_gotoXY(x + 6, y + 2);
    printf("A - accumulator");
    mt_gotoXY(x + 7, y + 2);
    printf("C - instructionCounter");
    mt_gotoXY(x + 8, y + 2);
    printf("Press \"Q\" to exit.");
}

```

- Вывод больших символов в консоль:

```
void visualBigCharArea()
{
    int value;

    int x = 2 + STD_DX_MEM, y = 2;
    int dx = 10, dy = 49;

    mt_setfgcolor(WHITE);
    mt_setbgcolor(BLACK);
    bc_box(x, y, dx, dy);

    unsigned int big[2] = {0, 0};
    char ss[8];

    sc_memoryGet(pointer_mem, &value);
    if ((value >> 14))
    {
        value = value & 0xffff;
        sprintf(ss, "-%04X", value);
    }
    else
    {
        sprintf(ss, "+%04X", value);
    }

    for (int i = 0; i < 5; i++)
    {
        bc_initBigChar(big, ss[i]);
        bc_printbigchar(big, x + 1, y + 2 + i * 9, BLACK, WHITE);
    }
}
```

Вывод

Сделали вывод нужных компонентов памяти и нужных ячеек с описанием. Конечный вид программы:



