Федеральное государственное бюджетное образовательное учреждение высшего образования "Национальный исследовательский университет "МЭИ"

ОТЧЕТ

По лабораторной работе № 4

# «Представление данных в памяти»

ФИО студента Ключников А.К.

Номер группы А-02-18

Имя преподавателя Мохов Андрей Сергеевич

Козлюк Дмитрий Александрович

Ход работы.

1. Для того, чтобы написать функции для печати отдельных байт и блока данных в шестнадцатеричном и в двоичном представлении, нам потребуются функции печати байтов в двоичном и шестнадцатеричном виде:

char nibble\_to\_hex(uint8\_t i)

{

assert( 0x0 <= i && i <= 0xf );

char digits[] = "0123456789abcdef";

return digits[i];

}

char bit\_digit(uint8\_t byte, uint8\_t bit)

{

if( byte & ( 0x1 << bit ) )

return '1';

return '0';

}

1. На основе этих функций составим требуемые:

void print\_in\_binary(uint8\_t byte)

{

for( int bit = 7; bit >= 0; bit-- )

cout << bit\_digit(byte, bit);

}

void print\_in\_binary(const void\* data, size\_t datasize)

{

const uint8\_t\* bytes = reinterpret\_cast<const uint8\_t\*>(data);

for (size\_t i = 0; i < datasize; i++)

{

print\_in\_binary(bytes[i]);

if ( (i + 1) % 4 == 0 )

cout << '\n';

else

cout << ' ';

}

}

1. Напишем калькулятор побитовых операций. На вход он будет получать численные значения 2 операндов в десятичном виде, и символьное значение оператора. Вывод включает в себя двоичные и шестнадцатеричные формы операндов и результатов операций:

void byte\_calc( uint16\_t first, uint16\_t second, char oper )

{

char operators[] = "&|^";

uint16\_t byteand = first & second;

uint16\_t byteor = first | second;

uint16\_t bytexor = first ^ second;

if( strchr( operators, oper ) == nullptr )

cout << "Wrong operator";

else {

switch( oper )

{

case '&':

print\_in\_hex( &first, sizeof(first) );

cout << "& ";

print\_in\_hex( &second, sizeof(second) );

cout << "= ";

print\_in\_hex( &byteand, sizeof(byteand) );

cout << "\n";

print\_in\_binary( &first, sizeof(first) );

cout << "& ";

print\_in\_binary( &second, sizeof(second));

cout << "= ";

print\_in\_binary( &byteand, sizeof(byteand) );

break;

case '|':

print\_in\_hex( &first, sizeof(first));

cout << "| ";

print\_in\_hex( &second, sizeof(second) );

cout << "= ";

print\_in\_hex( &byteand, sizeof(byteand) );

cout << "\n";

print\_in\_binary( &first, sizeof(first) );

cout << "| ";

print\_in\_binary( &second, sizeof(second) );

cout << "= ";

print\_in\_binary( &byteor, sizeof(byteor) );

break;

case '^':

print\_in\_hex( &first, sizeof(first));

cout << "^ ";

print\_in\_hex( &second, sizeof(second) );

cout << "= ";

print\_in\_hex( &byteand, sizeof(byteand) );

cout << "\n";

print\_in\_binary( &first, sizeof(first) );

cout << "^ ";

print\_in\_binary( &second, sizeof(second) );

cout << "= ";

print\_in\_binary( &bytexor, sizeof(bytexor) );

break;

}

}

}

uint16\_t first, second;

char oper;

cin >> first >> oper >> second;

byte\_calc( first, second, oper );

1. Опишем структуру **Student**, объявим и заполним массив из 2 студентов и их старосты.

Видим, что адрес массива совпадает с адресом первого элемента, а его размер равен сумме размеров каждого элемента, который в свою очередь, предопределен структурой: 17 байт на имя (каждый символ по 2 байта), 2 байта на год (целое беззнаковое), 4 байта на среднюю оценку (с плавающей запятой), 4 байта на адрес переменной старосты. Смещение полей структуры численно равно размеру предыдущего поля. Шестнадцатеричные адреса элементов также отличаются на значение размера элемента. Элемент-структура в байтовом представлении, вообще говоря, содержит в себе байты его полей по порядку.

1. Определим некоторые константы: максимальную длину имени файла, запрещенные символы, ширину расширения:

Const size\_t MAX\_SIZE = 256;

const size\_t MAX\_EXTENSION = 3;

const char\* forbidden = "\*<>?|\"";

Добавим ввод строки – имени файла. Важно отметить, что необходимо последний символ строки – символ переноса заменить на символ нуль-терминатора:

char text[MAX\_SIZE];

cout << "Enter file name: ";

fgets(text, MAX\_SIZE, stdin);

char\* filename = text;

filename[strlen(filename) - 1] = '\0';

1. Остальную часть, включающую в себя проверки на корректность имени файла, загрузку в память содержимого файла, ввод строки, поиск совпадений строки с содержимым файла, оформим в виде цикла **while**. Так будут выглядеть функции проверки имени файла на корректность:

if( strcspn( filename, forbidden ) == strlen( filename ) ) ;

else {

cout << "Invalid format";

break;

}

if( strchr( filename, ':' ) == nullptr ) ;

else {

char\* colon = strchr(filename,':');

if( colon - filename + 1 <= 1 ) {

cout << "Invalid format";

break;

}

while( colon != nullptr ) {

if( isalpha( filename[colon - filename - 1] ) > 0 &&

filename[colon - filename + 1] == '\\')

colon = strchr(colon + 1, ':');

else {

cout << "Invalid format";

break;

}

}

}

if( strrchr( filename, '.' ) == nullptr )

strcat( filename, ".txt" );

else {

char\* ext = strrchr( filename, '.' );

for( int i = 1; ext[i] != '\0'; i ++ )

\*(ext + i) = char(tolower(\*(ext + i)));

if( strncmp( ext, ".txt", MAX\_EXTENSION + 1 ) == 0 ) ;

else {

cout << "Invalid format";

break;

}

}

1. Загрузим в память и сохраним в динамическом массиве-строке содержимое файла, определим размер содержимого в байтах. Добавим запрос строки у пользователя, также не забыв записать в нее нуль-терминатор.

FILE\* file = fopen( filename, "r" );

if( file == nullptr ) {

cout << "Invalid file name";

break;

}

fseek( file, 0, SEEK\_END );

long filesize = ftell( file );

char\* filedata = new char[filesize];

rewind( file );

fread( filedata, 1, filesize, file );

filedata[filesize] = '\0';

char STR[MAX\_SIZE];

cout << "Enter string: ";

fgets(STR, MAX\_SIZE, stdin);

char\* str = STR;

\*( str + strlen(str) - 1 ) = '\0';

1. Подсчитаем количество вхождений введенной строки в файл, выведем его. Также обязательно нужно закрыть файл, откуда копировалось содержимое, и освободить динамически выделенную память. Допишем оператор **break,** совершающий выход из главного цикла, следовательно завершающий работу программы:

int occur = 0;

while( \*filedata != '\0' ) {

if( strncmp( filedata, str, strlen(str) ) != 0 )

filedata += 1;

else {

occur ++;

filedata += 1;

}

}

cout << "Occurance number: " << occur;

delete[] filedata;

fclose( file );

break;