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

ОТЧЕТ

По лабораторной работе № 4

# «Представление данных в памяти»

ФИО студента Борисов Роман Витальевич

Номер группы А-01-18

Имя преподавателя Мохов Андрей Сергеевич

Василькова Полина Денисовна

Москва, «НИУ «МЭИ», 201

*Цель работы*

1. Знать представление различных типов и структур данных в памяти.
2. Уметь использовать средства языка и стандартной библиотеки C++ для манипуляций с битами данных, адресами памяти и строками C.

*Ход работы*

**1. Подготовить инструменты для исследований и отладки**

Написать функции для печати отдельных байт и блока данных data размером size байтов в шестнадцатеричном и в двоичном представлении.

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

{

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

char digits[] = "0123456789abcdef";

return digits[i];

}

const uint8\_t\*

as\_bytes(const void\* data)

{

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

}

void

print\_in\_hex(uint8\_t byte)

{

cout << nibble\_to\_hex(byte >> 4)

<< nibble\_to\_hex(byte & 0xf);

}

void

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

{

const uint8\_t\* bytes = as\_bytes(data);

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

{

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

if ((i + 1) % 16)

{

cout << ' ';

}

else

{

cout << '\n';

}

}

}

char

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

{

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

{

return '1';

}

return '0';

}

void

print\_in\_binary(uint8\_t byte)

{

for (uint8\_t bit = 8; bit > 0; bit--)

{

cout << bit\_digit(byte, bit-1);

}

}

void

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

{

const uint8\_t\* bytes = as\_bytes(data);

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

{

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

if ((i + 1) % 4 )

{

cout << ' ';

}

else

{

cout << '\n';

}

}

}

**2. Написать программу-калькулятор для побитовых операций**

Пользователь вводит первый операнд, затем оператор (символ &, | или ^), затем второй операнд. Программа выполняет указанное действие над операндами и печатает результат в шестнадцатеричном и двоичном виде. Операнды — двухбайтовые беззнаковые целые числа (uint16\_t).

void bitcalc()

{

uint16\_t operand1,operand2,result;

char op;

cin>> operand1>>op>>operand2;

assert(op == '&' || op=='|' || op=='^');

switch (op)

{

case '&':

result = operand1 & operand2;

break;

case '|':

result = operand1 | operand2;

break;

case '^':

result = operand1 ^ operand2;

break;

// default: cout<<"incorrect operator";

}

print\_in\_hex(&operand1, sizeof(operand1));

cout<<op<<" ";

print\_in\_hex(&operand2, sizeof(operand2));

cout<<"= ";

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

cout<<'\n';

print\_in\_binary(&operand1, sizeof(operand1));

cout<<op<<" ";

print\_in\_binary(&operand2, sizeof(operand2));

cout<<"= ";

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

}

**3. Изучить представление и размещение данных в памяти**

1. Определить структуру Student, описывающую студента атрибутами:
   1. Имя (массив из 17 символов, включая завершающий '\0').
   2. Год поступления (беззнаковое целое, 2 байта).
   3. Средний балл (с плавающей запятой).
   4. Пол, представленный одним битом (0 — женский, 1 — мужской).
   5. Количество пройденных курсов.
   6. Указатель на структуру Student, описывающую старосту группы (для старосты — нулевой указатель).
2. Объявить и заполнить массив из трех структур Student, описывающий двух студентов одной группы и их старосту.
3. Напечатать, занести в отчет и письменно пояснить:
   1. адрес и размер массива;
   2. адреса и размеры всех элементов массива;
   3. для всех полей одного из элементов массива (не старосты), кроме пола (у битовых полей нет адреса):
      * адрес,
      * смещение от начала структуры,
      * размер,
      * шестнадцатеричное и двоичное представление;
   4. все элементы массива в шестнадцатеричном виде с указанием соответствия блоков байт полям структур.

struct Student

{

char name[17];

uint16\_t year;

double mark;

uint8\_t gender : 1;

Student \*starosta;

uint8\_t cours\_compleate;

};

void

hex\_razdelenie\_na\_bloki(const void\* data, size\_t size,size\_t separat[6]);

int main()

{

Student students[3]=

{

{"Roma",2000,4.55,1,&students[2],1},

{"Yuri",2000,4.31,1,&students[2],1},

{"Misha",2001,4.80,0,nullptr,10} //starosta

};

cout<< "Adress of struct: "<<&students<<"; Size of struct: "<<sizeof(students)<<'\n';

cout<< "Elements of struct: \n";

cout<< "#\tAdress\t\tSize"<<'\n';

for (size\_t i = 0; i<3; i++)

{

cout<<i<<'\t'<<&students[i]<<'\t'<<sizeof(students[i])<<'\n';

}

size\_t s = 0;

cout<<"\nElement #"<<s<<":\n";

cout<<"\nField\tAdress\t\tOffset\tSize\n"

<<"Name\t"<<&students[s].name<<'\t'<<offsetof(Student,name)<<'\t'<<sizeof(students[s].name);

cout<<"\nHex:\n";

print\_in\_hex(students[s].name,sizeof(students[s].name));

cout<<"\nBin:\n";

print\_in\_binary(students[s].name,sizeof(students[s].name));

cout<<"\n\nField\tAdress\t\tOffset\tSize\n"

<<"Year\t"<<&students[s].year<<'\t'<<offsetof(Student,year)<<'\t'<<sizeof(students[s].year);

cout<<"\nHex:\t";

print\_in\_hex(&students[s].year,sizeof(students[s].year));

cout<<'\t';

cout<<"\nBin:\t";

print\_in\_binary(&students[s].year,sizeof(students[s].year));

cout<<"\n\nField\tAdress\t\tOffset\tSize\n"

<<"Mark\t"<<&students[s].mark<<'\t'<<offsetof(Student,mark)<<'\t'<<sizeof(students[s].mark);

cout<<"\nHex:\t";

print\_in\_hex(&students[s].mark,sizeof(students[s].mark));

cout<<'\t';

cout<<"\nBin:\t";

print\_in\_binary(&students[s].mark,sizeof(students[s].mark));

cout<<"\n\nField\tAdress\tOffset\tSize\n"

<<"Courses\t"<<&students[s].cours\_compleate<<'\t'<<offsetof(Student,cours\_compleate)<<'\t'<<sizeof(students[s].cours\_compleate);

cout<<"\nHex:\t";

print\_in\_hex(&students[s].cours\_compleate,sizeof(students[s].cours\_compleate));

cout<<'\t';

cout<<"\nBin:\t";

print\_in\_binary(&students[s].cours\_compleate,sizeof(students[s].cours\_compleate));

cout<<"\n\nField\t\tAdress\t\tOffset\tSize\n"

<<"starosta\t"<<&students[s].starosta<<'\t'<<offsetof(Student,starosta)<<'\t'<<sizeof(students[s].starosta);

cout<<"\nHex:\t";

print\_in\_hex(&students[s].starosta,sizeof(students[s].starosta));

cout<<'\t';

cout<<"\nBin:\t";

print\_in\_binary(&students[s].starosta,sizeof(students[s].starosta));

size\_t offsetarrey[6] =

{

offsetof(Student,name),offsetof(Student,year),offsetof(Student,mark),

24,offsetof(Student,cours\_compleate),offsetof(Student,starosta),

};

cout<<"\n#|Name|Year|Mark|Gender|Courses|Starosta\n";

for( size\_t i = 0; i<3; i++)

{

cout<<'\n'<<"#"<<i<<" ";

hex\_razdelenie\_na\_bloki(&students[i],sizeof(students[i]),offsetarrey);

}

return 0;

}

void

hex\_razdelenie\_na\_bloki(const void\* data, size\_t size,size\_t separat[6])

{

const uint8\_t\* bytes = as\_bytes(data);

size\_t j = 0;

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

{

if (i == separat[j])

{

cout<<"|";

j++;

}

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

if ((i + 1) % 16)

{

cout << ' ';

}

else

{

cout << '\n';

}

}

}

В результате выполнения этой части программы в консоль выводится следующее сообщение:

Adress of struct: 0x6dfe50; Size of struct: 144

Elements of struct:

# Adress Size

0 0x6dfe50 48

1 0x6dfe80 48

2 0x6dfeb0 48

Element #0:

Field Adress Offset Size

Name 0x6dfe50 0 17

Hex:

4c 65 73 68 61 00 00 00 00 00 00 00 00 00 00 00

00

Bin:

01001100 01100101 01110011 01101000

01100001 00000000 00000000 00000000

00000000 00000000 00000000 00000000

00000000 00000000 00000000 00000000

00000000

Field Adress Offset Size

Year 0x6dfe62 18 2

Hex: d0 07

Bin: 11010000 00000111

Field Adress Offset Size

Mark 0x6dfe68 24 8

Hex: 33 33 33 33 33 33 12 40

Bin: 00110011 00110011 00110011 00110011

00110011 00110011 00010010 01000000

Field Adress Offset Size

Courses 40 1

Hex: 01

Bin: 00000001

Field Adress Offset Size

starosta 0x6dfe74 36 4

Hex: b0 fe 6d 00

Bin: 10110000 11111110 01101101 00000000

#|Name|Year|Mark|Gender|Courses|Starosta

#0 |4c 65 73 68 61 00 00 00 00 00 00 00 00 00 00 00

00 00 |d0 07 00 00 00 00 |33 33 33 33 33 33 12 40

01 00 00 00 b0 fe 6d 00 01 00 00 00 00 00 00 00

#1 |56 6f 76 61 00 00 00 00 00 00 00 00 00 00 00 00

00 00 |d0 07 00 00 00 00 |3d 0a d7 a3 70 3d 11 40

01 00 00 00 b0 fe 6d 00 01 00 00 00 00 00 00 00

#2 |4d 61 73 68 61 00 00 00 00 00 00 00 00 00 00 00

00 00 |d1 07 00 00 00 00 |33 33 33 33 33 33 13 40

00 00 00 00 00 00 00 00 0a 00 00 00 00 00 00 00

1. **Написать программу для обработки текстового файла**
2. Запросить у пользователя имя файла, сохранив его в массиве символов, размещенном на стеке (не в динамической памяти).
3. Проверить, используя функции стандартной библиотеки C++ для работы со строками C, что введенное имя файла корректно (в Windows):
   * не содержит запрещенных символов: \*, ", <, >, ? или |;
   * если содержит двоеточие, то только вторым символом, которому предшествует буква, и за двоеточием следует обратная косая черта (\).
   * если файл имеет расширение, то только \*.txt (в любом регистре).
4. Если введенное имя файла не имеет расширения, добавить расширение .txt.

void enter(char filename[128]){

for (bool valid = false; !valid;)

{valid = true;

cout << "Enter filename" << endl;

cin>> filename;

//проверка на особые символы

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

for (int i = 0; i<6; i++){

if (strchr(filename,bansymbols[i])){

valid = false;

cout<<"incorrect filename\n";

}

}

//проверка на двоеточие

for (int i = 0; i<128; i++){

if (filename[i] == ':') {

if ((i!=1) || !isalpha(filename[0]) || (filename[2]!='\\')) {

valid = false;

cout<<"incorrect filename\n";

}

}

}

//проверка расширения

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

const char\* txt = ".txt";

if (!strrchr(filename,'.') || strcmp(format,txt)){

strcat(filename,txt);

} }

}

1. Загрузить содержимое текстового файла в память целиком:
2. Запросить у пользователя строку, поместив её в массив на стеке.

char filename[128];

enter(filename);

cout<<filename;

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

fseek(file, SEEK\_SET,SEEK\_END-SEEK\_SET);

size\_t length = ftell(file);

length/=sizeof(char);

char \*data = new char[length];

fseek(file,0,0);

length = fread(data, sizeof(char),length,file);

data[length]='\0';

fclose(file);

char usstring[128];

cout << endl<< "Enter string" << endl;

cin>> usstring;

size\_t occurcount = findtext(usstring, data);

cout <<"Occurance number "<< occurcount;

delete[] data;

return 0;

1. Подсчитать и вывести число вхождений введенной строки в текст файла.

size\_t findtext(char str[128], char\* data)

{

size\_t strlength = strlen(str);

size\_t occur = 0;

for ( size\_t i=0; data[i]!='\0';i++ )

{

for (size\_t j =0; j<strlength;j++){

if (data[i+j]==str[j]){

if (j==strlength-1){occur++;};

}else break;

}

}

return occur;

}