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

**ОТЧЕТ**

**По лабораторной работе № 4**

**«Представление данных в памяти»**

ФИО студента Иванов Сергей Евгеньевич

Номер группы А-01-18

Имя преподавателя Мохов Андрей Сергеевич

Василькова Полина Денисовна

**Лабораторная работа №4**

**«Представление данных в памяти»**

**Цель работы:**

1. Изучить представление различных типов и структур данных в памяти ЭВМ.

2. Освоить средства языка и стандартной библиотеки C++ для низкоуровневых манипуляций с битами данных, адресами памяти и строками C.

**Задание на лабораторную работу**

**Общее задание:**

1. Подготовить инструменты для исследований и отладки.

Написать функции для печати отдельных байт и блока данных data размером size

байт в шестнадцатеричном и в двоичном представлении.

void print\_in\_hex(uint8\_t byte);

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

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

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

Указание. Для удобства чтения рекомендуется между байтами добавлять пробелы и делать перевод строки, например, после каждых 16-и байт (в print\_in\_hex()) или каждых 4-х байт (в print\_in\_binary()).

2. Написать программу-калькулятор для побитовых операций.

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

3. Изучить представление и размещение данных в памяти.

3.1. 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;

Пример ввода: 1025 & 127

Соответствующий вывод:

01 04 & 7F 00 = 01 0000000001 00000100 & 01111111 00000000 = 00000001 00000000=2

6) указатель на структуру Student, описывающую старосту группы

(для старосты — нулевой указатель).

Указание. Поле размером в несколько бит (не больше, чем бит в определенным

целочисленном типе) можно объявить так:

целочисленный-тип имя-поля : число-бит;

3.2. Объявить и заполнить массив из трех структур Student, описывающий двух

студентов одной группы и их старосту.

3.3. Напечатать, занести в отчет и письменно пояснить:

1) адрес и размер массива;

2) адреса и размеры всех элементов массива;

3) для всех полей, кроме пола1, одного из элементов массива

(не старосты): адрес, смещение от начала структуры, размер,

шестнадцатеричное и двоичное представление;

4) все элементы массива в шестнадцатеричном виде с указанием

соответствия блоков байт полям структур.

Указание. Смещение поля field структуры типа type от начала любого её

экземпляра можно определить макросом offsetof(type, field).

4. Написать программу для обработки текстового файла, представляя текст только

строками C, размещаемыми в динамической памяти или на стеке.

4.1. Запросить у пользователя имя файла, сохранив его в массиве символов, размещенном на стеке (не в динамической памяти).

4.2. Проверить, используя функции стандартной библиотеки C++ для работы со строками C, что введенное имя файла корректно (в Windows):

1) не содержит запрещенных символов: \*, ", <, >, ? или |;

2) если содержит двоеточие, то только вторым символом, которому

предшествует буква, и за двоеточием следует обратная косая черта (\).

3) если файл имеет расширение, то только \*.txt (в любом регистре).

Указание. Задачи решаются стандартными функциями isalpha(), strchr(),

strrchr(), strncmp(), tolower().

4.3. Если введенное имя файла не имеет расширения, добавить расширение .txt.

4.4. Загрузить содержимое текстового файла в память целиком:

1) использовать ifstream или fopen() для доступа к файлу;

2) использовать методы seekg() и tellg() либо функции fseek()

и ftell() для определения размера файла, переместившись в его

конец и получив текущее положение в файле;

3) выделить в динамической памяти массив достаточного размера;

4) загрузить всё содержимое файла в выделенную область памяти

методом read() или функцией fread().

4.5. Запросить у пользователя строку, поместив её в массив на стеке.

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

4.7. Освободить все выделенные в процессе решения блоки памяти.

**main.cpp**

#include <iostream>

#include <cassert>

using namespace std;

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

{

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

char digits[] = "0123456789abcdef";

return digits[i];

}

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

{

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

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

}

const uint8\_t\* as\_bytes(const void\* data)

{

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

}

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 == 0) {

cout << '\n';

}

else {

cout << ' ';

}

}

}

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 = 7; bit > 0; bit--) {

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

}

cout << bit\_digit(byte, 0);

}

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 == 0) {

cout << '\n';

}

else {

cout << ' ';

}

}

}

struct Student {

char name[17];

uint32\_t year;

float sred\_ball;

int sex:1;

uint8\_t courses;

Student\*starosta;

};

int main(){

assert(nibble\_to\_hex(0x0) == '0');

assert(nibble\_to\_hex(0x1) == '1');

assert(nibble\_to\_hex(0x2) == '2');

assert(nibble\_to\_hex(0x3) == '3');

assert(nibble\_to\_hex(0x4) == '4');

assert(nibble\_to\_hex(0x5) == '5');

assert(nibble\_to\_hex(0x6) == '6');

assert(nibble\_to\_hex(0x7) == '7');

assert(nibble\_to\_hex(0x8) == '8');

assert(nibble\_to\_hex(0x9) == '9');

assert(nibble\_to\_hex(0xa) == 'a');

assert(nibble\_to\_hex(0xb) == 'b');

assert(nibble\_to\_hex(0xc) == 'c');

assert(nibble\_to\_hex(0xd) == 'd');

assert(nibble\_to\_hex(0xe) == 'e');

assert(nibble\_to\_hex(0xf) == 'f');

Student studarray[3]={

{"Oleg", 2018, 4.3, 0, 7, &studarray[1]},

{"Kirill", 2018, 4.0, 1, 7, nullptr },

{"Masha", 2018, 3.4, 1, 7, &studarray[1]}

};

cout<<"Address of array: "<< &studarray<<'\n';

cout<<"Size of array: "<< sizeof(studarray)<<'\n';

cout<<"\t Address of elem:\t Size of elem:\n";

for (int i=0; i<3; i++){

cout<<i<<":\t "<<&studarray[i]<<"\t\t "<< sizeof(studarray[i])<<'\n';

}

cout<<"FIRST ELEMENT\n";

cout<<"\t Address of field: \t Size of field\t Offset:\n";

cout<<"NAME: \t\t"<<&studarray[0].name<<"\t\t"<<sizeof(studarray[0].name)

<<'\t'<<offsetof(struct Student, name)<<'\n';

cout<<"Binary: ";

print\_in\_binary(&studarray[0].name, sizeof(studarray[0].name));

cout<<"Hex: ";

print\_in\_hex(&studarray[0].name, sizeof(studarray[0].name));

cout<<"\nYEAR: \t\t"<<&studarray[0].year <<"\t\t"<<sizeof(studarray[0].year)

<<'\t'<<offsetof(struct Student, year)<<'\n';

cout<<"Binary: ";

print\_in\_binary(&studarray[0].year, sizeof(studarray[0].year));

cout<<"Hex: ";

print\_in\_hex(&studarray[0].year, sizeof(studarray[0].year));

cout<<"\nSRED BALL: \t\t"<<&studarray[0].sred\_ball<<"\t\t"

<<sizeof(studarray[0].sred\_ball)

<<'\t'<<offsetof(struct Student, sred\_ball)<<'\n';

cout<<"Binary: ";

print\_in\_binary(&studarray[0].sred\_ball, sizeof(studarray[0].sred\_ball));

cout<<"Hex: ";

print\_in\_hex(&studarray[0].sred\_ball, sizeof(studarray[0].sred\_ball));

return 0;

}

**Описание решения:**

Написали функции для печати отдельных байт и блока данных data размером size байт в шестнадцатеричном и в двоичном представлении:

#include <iostream>

#include <cassert>

using namespace std;

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

{

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

char digits[] = "0123456789abcdef";

return digits[i];

}

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

{

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

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

}

const uint8\_t\* as\_bytes(const void\* data)

{

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

}

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 == 0) {

cout << '\n';

}

else {

cout << ' ';

}

}

}

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 = 7; bit > 0; bit--) {

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

}

cout << bit\_digit(byte, 0);

}

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 == 0) {

cout << '\n';

}

else {

cout << ' ';

}

}

}

int main(){

assert(nibble\_to\_hex(0x0) == '0');

assert(nibble\_to\_hex(0x1) == '1');

assert(nibble\_to\_hex(0x2) == '2');

assert(nibble\_to\_hex(0x3) == '3');

assert(nibble\_to\_hex(0x4) == '4');

assert(nibble\_to\_hex(0x5) == '5');

assert(nibble\_to\_hex(0x6) == '6');

assert(nibble\_to\_hex(0x7) == '7');

assert(nibble\_to\_hex(0x8) == '8');

assert(nibble\_to\_hex(0x9) == '9');

assert(nibble\_to\_hex(0xa) == 'a');

assert(nibble\_to\_hex(0xb) == 'b');

assert(nibble\_to\_hex(0xc) == 'c');

assert(nibble\_to\_hex(0xd) == 'd');

assert(nibble\_to\_hex(0xe) == 'e');

assert(nibble\_to\_hex(0xf) == 'f');

}

4. Написать программу для обработки текстового файла, представляя текст только строками C, размещаемыми в динамической памяти или на стеке.

#include <iostream>

#include <cstring>

#include <cstdio>

using namespace std;

int main()

{

const size\_t MAX\_SIZE = 256;

char text[MAX\_SIZE];

char\* fgets(char\* str, int count, FILE\* stream);

//stdin expression of type FILE\* associated with the input stream;

fgets(text, MAX\_SIZE, stdin);

const char\* separators = " \r\n,.!?:;()-";

const char\* start = text;

while (true)

{

const size\_t separator\_count = strspn(start, separators);

start += separator\_count;

if (start[0] == '\0')

{

break;

}

const size\_t word\_length = strcspn(start, separators);

cout.write(start, word\_length);

cout << '\n';

start += word\_length;

}

return 0;

}