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

ОТЧЕТ

По лабораторной работе № 4

«Представление данных в памяти»

ФИО студента	Уткин Роман
Номер группы	А-01-18
Имя преподавателя	Мохов Андрей Сергеевич

Лабораторная работа №4

«Представление данных в памяти»

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.

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 С, 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]=
{
   {"Roman", 2000, 4.3, 0, 7, &studarray[1]},
   {"Sergey", 1998, 4.0, 1, 7, nullptr },
   {"Ivan", 2000, 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:

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

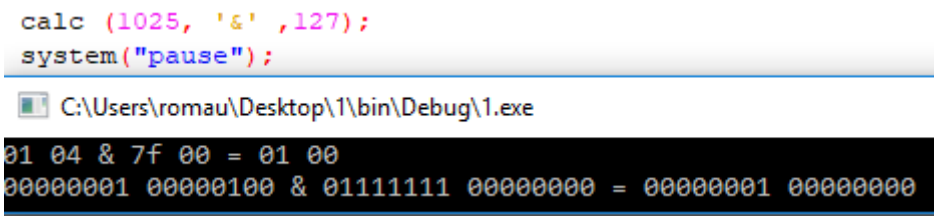
```
#include <iostream>
#include <cassert>
using namespace std;
char nibble_to_hex (uint8_t i)
{
   assert(0x0 <= i && i <= 0xf);
   char digits[] = "0123456789abcdef";
   return digits[i];
}
void print_in_hex (uint8_t byte)
{
   cout << nibble_to_hex(byte >> 4)
        << nibble_to_hex(byte & 0xf);
}
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');
}
```

2. Написали программу калькулятор для побитовых операций:

```
void calc(uint16_t op1,char oper,uint16_t op2)
 {
    print_in_hex(&op1,sizeof op1);
    cout << oper << " ";
    print_in_hex(&op2,sizeof op2);
    cout << "= ";
    uint16_t result;
    if (oper == '&')
    result=op1 & op2;
    if (oper == '|')
    result=op1 | op2;
    if (oper == '^')
    result = op1 ^ op2;
    print_in_hex(&result,sizeof(result));
    cout << '\n';
    print_in_binary(&op1,sizeof op1);
    cout << oper << " ";
    print_in_binary(&op2,sizeof op2);
    cout << "= ";
    print_in_binary(&result,sizeof(result));
    cout << '\n';
 }
```

Пример работы:



Ввод: `1025 & 127`

Вывод на экран: `01 04 & 7f 00 = 01 00`

`00000001 00000100 & 01111111 00000000 = 00000001 00000000`

3. Определили структуру Student, объявили массив из трех структур, описывающий двух студентов одной группы и их старосту.

```
struct Student

{

   char name[17];

   uint32_t year;

   float sred_ball;
```

```
   int sex:1;
   uint8_t courses;
   Student*starosta;
};
Student studarray[3]=
   {
      {"Roman", 2000, 4.3, 0, 7, &studarray[1]},
      {"Sergey", 1998, 4.0, 1, 7, nullptr },
      {"Ivan", 2000, 3.4, 1, 7, &studarray[1]}
   };
```

4. Решим задачу: считать строку С и напечатать по отдельности слов в ней (слова разделены пробелами и знаками препинания).

```
#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);
    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;
}
```