Лабораторная работа 1 (0001 = 1)

Ввод-вывод при помощи libc

Цель работы: изучить размеры стандартных типов C/C++ на выбранной платформе; научиться использовать функции ввода-вывода libc.

Все задания Л1 выполняются на чистом C/C++, без использования ассемблера.

Задание Л1.№1

Напечатайте (выведите на стандартный вывод) группу, номер и состав команды при помощи функции $puts()$ библиотеки libc.

```
Группа: ПИН-31Д
Номер: 17
Выполнил: Понкращенков Д.Б.
```

Рис. 1: Результат выполнения puts

```
Листинг:
1. puts("Группа: ПИН-31Д\nНомер: 17\nВыполнил: Понкращенков Д.Б.");
```

Задание Л1.№2

Укажите для платформы, где выполняется работа: — ОС и разрядность ОС; — компилятор (должен относиться к коллекции GCC/MinGW) и его версию; — разрядность сборки; — архитектуру процессора, назначение платформы.

При помощи оператора sizeof языка C/C++ выясните, сколько байтов занимают на выбранной платформе переменные следующих типов: char, char*, bool, wchar_t, short, unsigned short, int, long, long long, float, double, double*, long double, size_t, ptrdiff_t, void*.

```
Задание №2
================================================================================
ОС: Linux
Архитектура процессора: x86_64 (64-бит)
Compiler: GCC
Version: 13.2.1

Информация о размерах стандартных типов:
Размер char = 1 байт
Размер char* = 8 байт
Размер bool = 1 байт
Размер wchar_t = 4 байт
Размер short = 2 байт
Размер unsigned short = 2 байт
Размер int = 4 байт
Размер long = 8 байт
Размер long long = 8 байт
Размер float = 4 байт
Размер double = 8 байт
Размер double* = 8 байт
Размер long double = 16 байт
Размер size_t = 8 байт
Размер ptrdiff_t = 8 байт
Размер void* = 8 байт
```

Рис. 2: Информация о системе и размерах типов

Листинг:

Файл task2.h:

```h
#ifndef task2_H
#define task2_H

#include <stdio.h>

#define PRINT_SIZE(type) printf("Размер %s = %zu байт\n", #type, sizeof(type))

void print_os();
void print_cpu_arch();
void print_compiler_version();

void printSizeOfAll();

void run_task2()
{
    printf("\nЗадание №2\n");
    printf("================================================================================\n");

    print_os();
    print_cpu_arch();
    print_compiler_version();

    printSizeOfAll();
}

#endif
```

Файл task2.c:

```c
#include <stdio.h>    // для printf и других функций ввода-вывода
#include <stddef.h>
#include <stdbool.h>
#include <wchar.h>

#include "task2.h"

    void printSizeOfAll(){

        printf("\nИнформация о размерах стандартных типов:\n");

        PRINT_SIZE(char);
        PRINT_SIZE(char*);
        PRINT_SIZE(bool);
        PRINT_SIZE(wchar_t);
        PRINT_SIZE(short);
        PRINT_SIZE(unsigned short);
        PRINT_SIZE(int);
        PRINT_SIZE(long);
        PRINT_SIZE(long long);
        PRINT_SIZE(float);
        PRINT_SIZE(double);
        PRINT_SIZE(double*);
        PRINT_SIZE(long double);
        PRINT_SIZE(size_t);
        PRINT_SIZE(ptrdiff_t);
        PRINT_SIZE(void*);
    }

    void print_cpu_arch() {
    #if defined(__x86_64__) || defined(_M_X64)
        printf("Архитектура процессора: x86_64 (64-бит)\n");
    #elif defined(__i386) || defined(_M_IX86)
        printf("Архитектура процессора: x86 (32-бит)\n");
    #elif defined(__arm__) || defined(_M_ARM)
        printf("Архитектура процессора: ARM\n");
    #elif defined(__aarch64__)
        printf("Архитектура процессора: ARM64\n");
    #elif defined(__mips__)
        printf("Архитектура процессора: MIPS\n");
    #else
        printf("Архитектура процессора: неизвестна\n");
    #endif
    }

```

```
    void print_compiler_version() {
#if defined(__clang__)
    printf("Compiler: Clang/LLVM\n");
    printf("Version: %d.%d.%d\n", __clang_major__, __clang_minor__, __clang_patchlevel__);
#elif defined(__GNUC__)
    printf("Compiler: GCC\n");
    printf("Version: %d.%d.%d\n", __GNUC__, __GNUC_MINOR__, __GNUC_PATCHLEVEL__);
#elif defined(_MSC_VER)
    printf("Compiler: Microsoft Visual C++\n");
    printf("Version: %d\n", _MSC_VER);
#else
    printf("Unknown compiler\n");
#endif
    }

    void print_os() {
    #if defined(_WIN32) || defined(_WIN64)
        printf("ОС: Windows\n");
    #elif defined(__linux__)
        printf("ОС: Linux\n");
    #elif defined(__APPLE__) && defined(__MACH__)
        printf("ОС: macOS\n");
    #else
        printf("ОС: Неизвестная\n");
    #endif
    }
```

Задание Л1.№3

Создайте и инициализируйте заданными значениями x шесть массивов $M*$, каждый из N = 5 чисел (типы указаны для 32/64-битных режимов x86/amd64, для иных платформ выбирайте по результатам задания Л1.№2):

— $M\ b$ из 8-битных целых чисел $char/unsigned\ char$ (x = 0xED);
— Ms из 16-битных целых чисел $short/unsigned\ short$ (x = 0xFADE);
— Ml из 32-битных целых чисел $int/unsigned\ int$ (x = 0x ADE1 A1DA);
— Mq из 64-битных целых чисел $long\ long/unsigned\ long\ long$ (x = 0x C1A5 51F1 AB1E);
— Mfs из 32-битных чисел с плавающей запятой $float$ (x из таблицы Л1.1);
— Mfl из 64-битных чисел с плавающей запятой $double$ (x из таблицы Л1.1).

Не используйте типы фиксированной разрядности $int*\_t/uint*\_t$, так как для них не существует модификаторов размера форматных полей $printf()/scanf()$.

Напечатайте все массивы $M*$ при помощи функции libc $printf()$ (раздел ??).

1. Целочисленные Mb, Ms, Ml, Mq четырежды (каждый — четырежды):
   a) шестнадцатеричном представлении (формат X);
   b) в двоичном представлении (формат b);
   c) в десятичном беззнаковом представлении (формат u);
   d) в десятичном знаковом представлении (формат d).

2. С плавающей запятой — Mfs и Mfl— трижды:
   a) в шестнадцатеричном экспоненциальном представлении (формат A);
   b) в десятичном экспоненциальном представлении (формат e);
   c) в представлении с десятичной запятой (формат f).

Напечатайте 8-битный Mb ещё пятый раз: в символьной форме (формат c). Так как тип по умолчанию для формата c — это именно $char/unsigned\ char$, модификатор размера не нужен.

Элементы массива разделяйте пробелами.

Если компилятор устарел и не поддерживает форматов b, A или модификатора размера hh, отметьте это в отчёте и реализуйте то, что поддерживается.

Укажите для какой-либо пары массив+формат ширину поля вывода, добавив некоторое число w между символом % и модификатором размера (если модификатора размера нет — между % и форматом). Пусть исходная ширина выводимого числа $w0$ знакомест: как изменится вывод при w 6 $w0$, как при $w > w0$? Установите $w > w0$. Добавьте между % и w знак «−» (минус). Что изменилось?

Дополните шестнадцатеричное представление всех целочисленных массивов (формат X) ведущими нулями до необходимого количества цифр. Для этого:
a) рассчитайте необходимое количество шестнадцатеричных цифр w — по две на байт, так, для $short/unsigned\ short$ цифр w = 2·sizeof($short$) = 2·2 = 4;
b) укажите ширину поля вывода w и перед ней, но после % — символ 0, так, для $short/unsigned\ short$ форматное поле $\%hX$ замените на $\%04hX$.

Как изменился вывод?

Дополните двоичное представление (формат b) аналогично ведущими нулями до количества бит w = 8 · sizeof($M[i]$).
Дополните знаковое десятичное представление целочисленных массивов (формат d) знаком «+» перед положительными числами.

Для одного из массивов с плавающей запятой и форматов A, e, f и всех форматов его вывода задайте точность в две цифры после запятой, для чего добавьте .2 между шириной поля вывода и модификатором размера.

Что изменилось?

Штраф −1 балл, если вместо именованной константы N здесь и/или позже используется литерал 5.

Штраф −1 балл за пару тип+формат ,где не указан модификатор размера (раздел ??) для типов, отличных от $int/unsigned/float$.

Бонус +1 балл, если вывод массива во всех формах описан как функция и в последующих заданиях используется вызов этой функции, а не копирование и вставка;

+2 балла, если эта функция описана как единый для всех массивов шаблон и принимает тип как параметр шаблона, а адрес начала M, длину N и форматы с модификатором размера как параметры функции;

+3 балла, если вывод описан как единый для всех массивов макрос с соответствующими параметрами.

```
Задание №3
========================================================================
Массив Mb в шестнадцатеричном формате (%02hhX):
ED ED ED ED ED
Массив Mb в двоичном формате (%08b):
11101101 11101101 11101101 11101101 11101101
Массив Mb в десятичном беззнаковом формате (%hhu):
237 237 237 237 237
Массив Mb в десятичном знаковом формате (%+hhd):
-19 -19 -19 -19 -19
========================================================================
Массив Ms в шестнадцатеричном формате (%04hX):
FADE FADE FADE FADE FADE
Массив Ms в двоичном формате (%08b):
11101101 11101101 11101101 11101101 11101101
Массив Ms в десятичном беззнаковом формате (%hu):
64222 64222 64222 64222 64222
Массив Ms в десятичном знаковом формате (%+hd):
-1314 -1314 -1314 -1314 -1314
========================================================================
Массив Ml в шестнадцатеричном формате (%08X):
ADE1A1DA ADE1A1DA ADE1A1DA ADE1A1DA ADE1A1DA
Массив Ml в двоичном формате (%08b):
10101101111100001101000011101101 10101101111100001101000011101101 10101101
Массив Ml в десятичном беззнаковом формате (%u):
2917245402 2917245402 2917245402 2917245402 2917245402
Массив Ml в десятичном знаковом формате (%+d):
-1377721894 -1377721894 -1377721894 -1377721894 -1377721894
========================================================================
Массив Mq в шестнадцатеричном формате (%016llX):
0000C1A551F1AB1E 0000C1A551F1AB1E 0000C1A551F1AB1E 0000C1A551F1AB1E 0000C1
Массив Mq в двоичном формате (самописная ф-я):
0000000000000000110000011010010101010001111100011010101100011110 000000000
Массив Mq в десятичном беззнаковом формате (%llu):
212915788557086 212915788557086 212915788557086 212915788557086 2129157885
Массив Mq в десятичном знаковом формате (%+lld):
+212915788557086 +212915788557086 +212915788557086 +212915788557086 +21291
========================================================================
Массив Mfs в шестнадцатеричном экспоненциальном формате (%A):
0X1.555556P+1 0X1.555556P+1 0X1.555556P+1 0X1.555556P+1 0X1.555556P+1
Массив Mfs в десятичном экспоненциальном формате (%.2e):
2.67e+00 2.67e+00 2.67e+00 2.67e+00 2.67e+00
Массив Mfs в десятичном формате с запятой (%.2f):
2.666667 2.666667 2.666667 2.666667 2.666667
========================================================================
Массив Mfl в шестнадцатеричном экспоненциальном формате (%A):
0X1.5555555555555P+1 0X1.5555555555555P+1 0X1.5555555555555P+1 0X1.5555555
Массив Mfl в десятичном экспоненциальном формате (%.2e):
2.67e+00 2.67e+00 2.67e+00 2.67e+00 2.67e+00
Массив Mfl в десятичном формате с запятой (%.2f):
2.666667 2.666667 2.666667 2.666667 2.666667
========================================================================
```

Рис. 3: Результаты вывода массивов в различных форматах

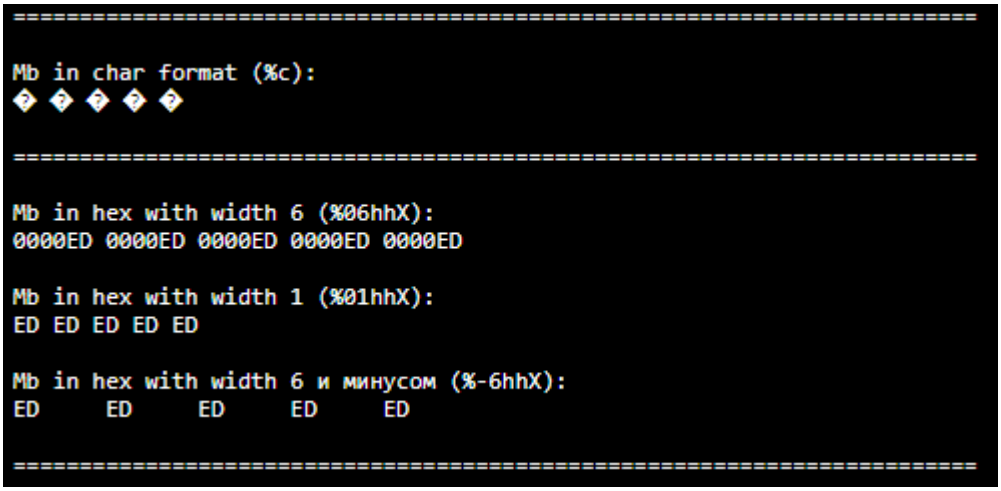

Рис. 4: Печать 8-битного Mb (формат c) и работа с отступами.

Листинг:

Файл task3.h:

```c
#ifndef task3_H
#define task3_H

#include <stdio.h>

#define N 5

// Макрос для вывода массива с параметризацией типа, формата и модификаторов
#define PRINT_ARRAY(arr, type, format_str) \
    do { \
        for (int i = 0; i < N; i++) { \
            printf(format_str "%c", (type)(arr[i]), (i == N-1) ? '\n' : ' '); \
        } \
    } while(0)

void print_binary_array(const void *arr, size_t elem_size, int is_signed);

void run_task3()
{
    printf("\n3адание №3\n");

    // Инициализация массивов
    unsigned char Mb[N] = {0xED, 0xED, 0xED, 0xED, 0xED};
    unsigned short Ms[N] = {0xFADE, 0xFADE, 0xFADE, 0xFADE, 0xFADE};
    unsigned int Ml[N] = {0xADE1A1DA, 0xADE1A1DA, 0xADE1A1DA, 0xADE1A1DA, 0xADE1A1DA};
    unsigned long long Mq[N] = {0xC1A551F1AB1EULL, 0xC1A551F1AB1EULL, 0xC1A551F1AB1EULL,
0xC1A551F1AB1EULL, 0xC1A551F1AB1EULL};

    // Вариант 17: x = 8/3 для float/double
    float Mfs[N];
    double Mfl[N];
    for (size_t i = 0; i < N; i++) {
        Mfs[i] = 8.0f / 3.0f;
        Mfl[i] = 8.0 / 3.0;
    }

    printf("================================================================================\n");

    // 1. Целочисленные массивы: вывод 4 раза (X, b, u, d)
    // Форматы с ведущими нулями для X:
    // Mb: 2*1=2, Ms:4, Ml:8, Mq:16
    // 1) Шестнадцатеричный формат с ведущими нулями (0xED → "ED")
    printf("Массив Mb в шестнадцатеричном формате (%%02hhX):\n");
    PRINT_ARRAY(Mb, unsigned char, "%02hhX");
    // 2) Двоичный формат с ведущими нулями
    printf("Массив Mb в двоичном формате (%%08b):\n");
    PRINT_ARRAY(Mb, unsigned char, "%08b");  //print_binary_array(Mb, sizeof(Mb[0]), 0);
    // 3) Десятичный беззнаковый формат
    printf("Массив Mb в десятичном беззнаковом формате (%%hhu):\n");
    PRINT_ARRAY(Mb, unsigned char, "%hhu");
    // 4) Десятичный знаковый формат с плюсом для положительных
    printf("Массив Mb в десятичном знаковом формате (%%+hhd):\n");
    PRINT_ARRAY(Mb, signed char, "%+hhd");

```

```
54.     printf("=============================================================================\n");
55.
56.     // 1) Шестнадцатеричный формат с ведущими нулями (0xED → "ED")
57.     printf("Массив Ms в шестнадцатеричном формате (%%04hX):\n");
58.     PRINT_ARRAY(Ms, unsigned short, "%04hX");
59.     // 2) Двоичный формат с ведущими нулями
60.     printf("Массив Ms в двоичном формате (%%08b):\n");
61.     PRINT_ARRAY(Mb, unsigned short, "%08b"); //print_binary_array(Ms, sizeof(Ms[0]), 0);
62.     // 3) Десятичный беззнаковый формат
63.     printf("Массив Ms в десятичном беззнаковом формате (%%hu):\n");
64.     PRINT_ARRAY(Ms, unsigned short, "%hu");
65.     // 4) Десятичный знаковый формат с плюсом для положительных
66.     printf("Массив Ms в десятичном знаковом формате (%%+hd):\n");
67.     PRINT_ARRAY(Ms, short, "%+hd");
68.
69.     printf("=============================================================================\n");
70.
71.     // 1) Шестнадцатеричный формат с ведущими нулями (0xED → "ED")
72.     printf("Массив Ml в шестнадцатеричном формате (%%08X):\n");
73.     PRINT_ARRAY(Ml, unsigned int, "%08X");
74.     // 2) Двоичный формат с ведущими нулями
75.     printf("Массив Ml в двоичном формате (%%08b):\n");
76.     print_binary_array(Ml, sizeof(Ml[0]), 0);
77.     // 3) Десятичный беззнаковый формат
78.     printf("Массив Ml в десятичном беззнаковом формате (%%u):\n");
79.     PRINT_ARRAY(Ml, unsigned int, "%u");
80.     // 4) Десятичный знаковый формат с плюсом для положительных
81.     printf("Массив Ml в десятичном знаковом формате (%%+d):\n");
82.     PRINT_ARRAY(Ml, int, "%+d");
83.
84.     printf("=============================================================================\n");
85.
86.     // 1) Шестнадцатеричный формат с ведущими нулями (0xED → "ED")
87.     printf("Массив Mq в шестнадцатеричном формате (%%016llX):\n");
88.     PRINT_ARRAY(Mq, unsigned long long, "%016llX");
89.     // 2) Двоичный формат с ведущими нулями
90.     printf("Массив Mq в двоичном формате (самописная ф-я):\n");
91.     print_binary_array(Mq, sizeof(Mq[0]), 0);
92.     // 3) Десятичный беззнаковый формат
93.     printf("Массив Mq в десятичном беззнаковом формате (%%llu):\n");
94.     PRINT_ARRAY(Mq, unsigned long long, "%llu");
95.     // 4) Десятичный знаковый формат с плюсом для положительных
96.     printf("Массив Mq в десятичном знаковом формате (%%+lld):\n");
97.     PRINT_ARRAY(Mq, long long, "%+lld");
98.
99.     printf("=============================================================================\n");
100.
101.    // 2. Плавающая точка: 3 раза (A, e, f)
102.    printf("Массив Mfs в шестнадцатеричном экспоненциальном формате (%%A):\n");
103.    PRINT_ARRAY(Mfs, float, "%A");
104.    printf("Массив Mfs в десятичном экспоненциальном формате (%%.2e):\n");
105.    PRINT_ARRAY(Mfs, float, "%.2e");
106.    printf("Массив Mfs в десятичном формате с запятой (%%.2f):\n");
107.    PRINT_ARRAY(Mfs, float, "%f");
108.
109.    printf("=============================================================================\n");
110.
111.    printf("Массив Mfl в шестнадцатеричном экспоненциальном формате (%%A):\n");
112.    PRINT_ARRAY(Mfl, double, "%A");
113.    printf("Массив Mfl в десятичном экспоненциальном формате (%%.2e):\n");
114.    PRINT_ARRAY(Mfl, double, "%.2e");
115.    printf("Массив Mfl в десятичном формате с запятой (%%.2f):\n");
116.    PRINT_ARRAY(Mfl, double, "%f");
117.
118.    printf("=============================================================================\n");
119.
120.    // 5-й раз Mb в символьном формате
121.    printf("\nMb in char format (%%c):\n");
122.    PRINT_ARRAY(Mb, unsigned char, "%c");
123.
124.    printf("\n=============================================================================\n");
125.
126.    // Демонстрация ширины поля вывода
127.    printf("\nMb in hex with width 6 (%%06hhX):\n");
128.    PRINT_ARRAY(Mb, unsigned char, "%06hhX"); // ширина больше w0=2, добавятся нули слева
129.
130.    printf("\nMb in hex with width 1 (%%01hhX):\n");
131.    PRINT_ARRAY(Mb, unsigned char, "%01hhX"); // ширина меньше w0=2, выводится без усечения
```

```
132.
133.     printf("\nMb in hex with width 6 и минусом (%%-6hhX):\n");
134.     PRINT_ARRAY(Mb, unsigned char, "%-6hhX"); // выравнивание по левому краю, справа пробелы
135.
136.     printf("\n================================================================================\n");
137.
138.     printf("\n");
139. }
140.
141. #endif
```

Файл task3.c:

```
 1. #include <stdint.h>   // для uint64_t
 2. #include <stddef.h>   // для size_t, wchar_t, ptrdiff_t
 3. #include <stdbool.h>  // для bool
 4. #include <stdio.h>    // для printf и других функций ввода-вывода
 5.
 6. #include "task3.h"
 7.
 8. // Функция для вывода двоичного представления одного значения
 9. void print_binary_val(unsigned long long val, int bits) {
10.     for (int i = bits - 1; i >= 0; i--) {
11.         putchar((val & (1ULL << i)) ? '1' : '0');
12.     }
13. }
14.
15. // Вспомогательная функция для вывода двоичного формата с ведущими нулями
16. void print_binary_array(const void *arr, size_t elem_size, int is_signed) {
17.     unsigned char *bytes = (unsigned char *)arr;
18.     for (int i = 0; i < N; ++i) {
19.         unsigned long long val = 0;
20.         // Скопируем байты элемента в val (учитывая порядок байт)
21.         for (size_t b = 0; b < elem_size; ++b) {
22.             val |= ((unsigned long long)bytes[i * elem_size + b]) << (8 * b);
23.         }
24.         // Выводим биты с ведущими нулями
25.         print_binary_val(val, 8 * elem_size);
26.         putchar(i == N - 1 ? '\n' : ' ');
27.     }
28. }
```

Задание Л1.№4.

Для каждого массива M из всех созданных введите с клавиатуры новое значение элемента $M[i]$, i = 2 при помощи функции libc $scanf()$.

Проанализировав возвращённое $scanf()$ значение, определите корректность ввода; при необходимости отобразите сообщение об ошибке при помощи функции libc $puts()$.

Выведите массивы на экран до и после ввода, каждый раз — во всех форматах, описанных в Л1.№3; убедитесь, что элемент $M[i]$ приобрёл ожидаемое значение, а другие элементы массива не изменились (если изменились — проверьте, верно ли вы указали модификатор размера).

В данном задании необходимо передать функции $scanf()$ адрес $M[i]$, а не промежуточной переменной — иначе нет смысла контролировать значение соседних элементов массива.

Штраф −1 балл, если массив после ввода нового значения $M[i]$ выводится только в одном формате (а если для целочисленных это ещё и не X, то −2 балла).

Штраф −2 балла, если используется промежуточная переменная для ввода-вывода.

```
Задание №4
================================================================================
Mb in hex (X):
ED ED ED ED ED
Mb in binary (b):
11101101 11101101 11101101 11101101 11101101
Mb unsigned decimal (u):
237 237 237 237 237
Mb signed decimal with plus (d):
-19 -19 -19 -19 -19
Введите новое значение Mb[2] (целое число 0..255): 100
Mb in hex (X):
ED ED 64 ED ED
Mb in binary (b):
11101101 11101101 01100100 11101101 11101101
Mb unsigned decimal (u):
237 237 100 237 237
Mb signed decimal with plus (d):
-19 -19 +100 -19 -19
Ms in hex (X):
FADE FADE FADE FADE FADE
Ms in binary (b):
11101101 11101101 01100100 11101101 11101101
Ms unsigned decimal (u):
64222 64222 64222 64222 64222
Ms signed decimal with plus (d):
-1314 -1314 -1314 -1314 -1314
Введите новое значение Ms[2] (целое число 0..65535): 100
Ms in hex (X):
FADE FADE 0064 FADE FADE
Ms in binary (b):
11101101 11101101 01100100 11101101 11101101
Ms unsigned decimal (u):
64222 64222 100 64222 64222
Ms signed decimal with plus (d):
-1314 -1314 +100 -1314 -1314
Ml in hex (X):
ADE1A1DA ADE1A1DA ADE1A1DA ADE1A1DA ADE1A1DA
Ml in binary (b):
11101101 11101101 01100100 11101101 11101101
Ml unsigned decimal (u):
2917245402 2917245402 2917245402 2917245402 2917245402
Ml signed decimal with plus (d):
-1377721894 -1377721894 -1377721894 -1377721894 -1377721894
Введите новое значение Ml[2] (целое число 0..4294967295): 100
Ml in hex (X):
ADE1A1DA ADE1A1DA 00000064 ADE1A1DA ADE1A1DA
Ml in binary (b):
11101101 11101101 01100100 11101101 11101101
Ml unsigned decimal (u):
2917245402 2917245402 100 2917245402 2917245402
Ml signed decimal with plus (d):
-1377721894 -1377721894 +100 -1377721894 -1377721894
```

Рис. 5: Ввод для каждого целочисленного массива M[2] и вывод изменений

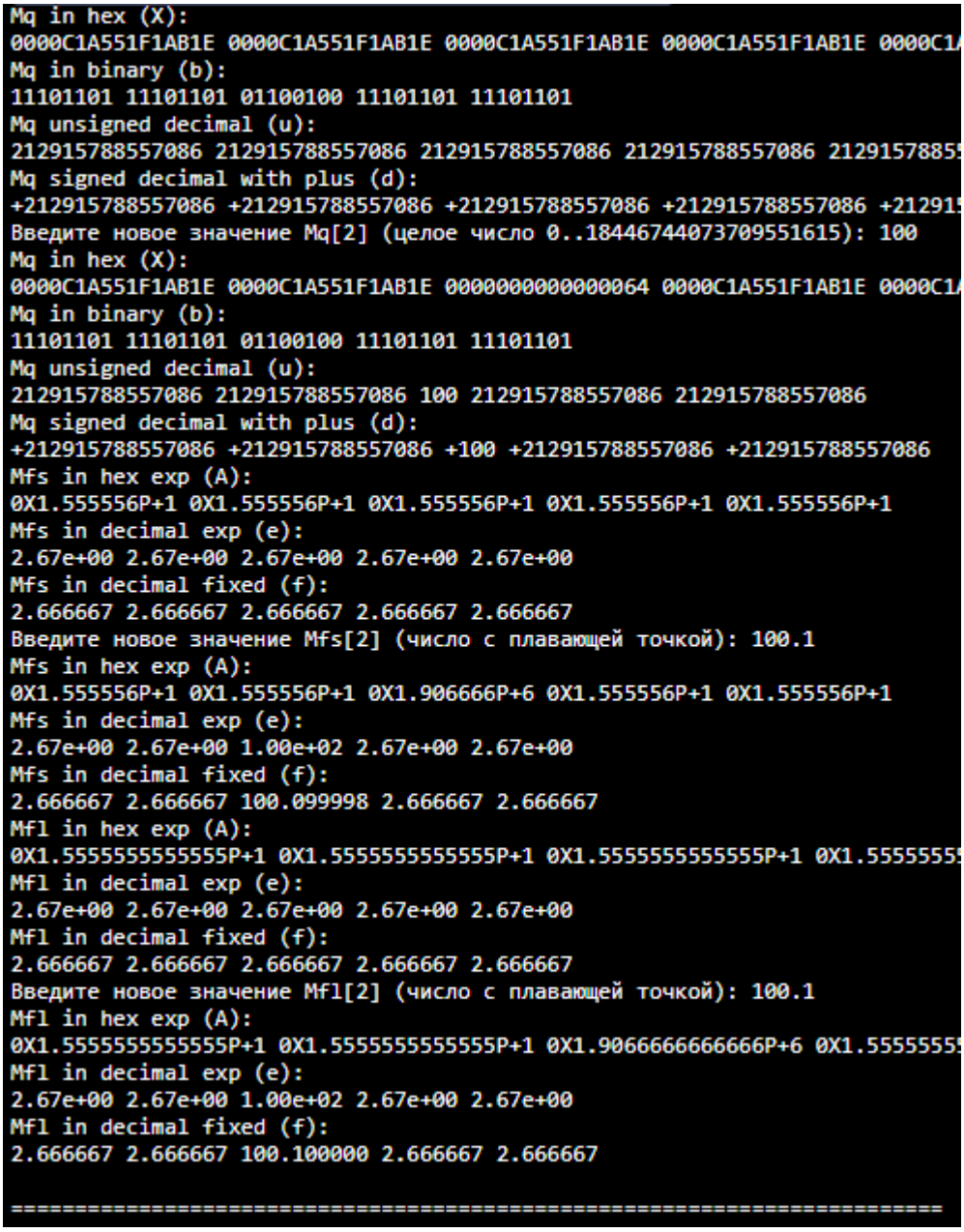

Рис. 6: Ввод для каждого массива чисел с плавающей запятой M[2] и вывод изменений

Листинг:

Файл Task4.h:

```
#ifndef task4_H
#define task4_H

#include <stdio.h>

#define N 5

// Макрос для вывода массива с параметризацией типа, формата и модификаторов
#define PRINT_ARRAY(arr, type, format_str) \
    do { \
        for (int i = 0; i < N; i++) { \
            printf(format_str "%c", (type)(arr[i]), (i == N-1) ? '\n' : ' '); \
        } \
    } while(0)

// Макрос для вывода целочисленного массива во всех форматах
#define PRINT_INT_ARRAY_ALL(arr, type, signed_type, fmt_hex, fmt_unsigned, fmt_signed, elem_size)
do { \
    printf(#arr " in hex (X):\n"); \
    PRINT_ARRAY(arr, type, fmt_hex); \
    printf(#arr " in binary (b):\n"); \
    PRINT_ARRAY(Mb, unsigned char, "%08b"); \
    printf(#arr " unsigned decimal (u):\n"); \
    PRINT_ARRAY(arr, type, fmt_unsigned); \
    printf(#arr " signed decimal with plus (d):\n"); \
    PRINT_ARRAY(arr, signed_type, fmt_signed); \
} while(0)

// Макрос для вывода массива с плавающей точкой во всех форматах
#define PRINT_FLOAT_ARRAY_ALL(arr, type, fmt_A, fmt_e, fmt_f) do { \
    printf(#arr " in hex exp (A):\n"); \
    PRINT_ARRAY(arr, type, fmt_A); \
    printf(#arr " in decimal exp (e):\n"); \
    PRINT_ARRAY(arr, type, fmt_e); \
    printf(#arr " in decimal fixed (f):\n"); \
    PRINT_ARRAY(arr, type, fmt_f); \
} while(0)

int input_element_int(void *element, const char *fmt);
int input_element_float(void *element, const char *fmt);

void run_task4()
{
    printf("\nЗадание №4\n");

    // Инициализация массивов
    unsigned char Mb[N] = {0xED, 0xED, 0xED, 0xED, 0xED};
    unsigned short Ms[N] = {0xFADE, 0xFADE, 0xFADE, 0xFADE, 0xFADE};
    unsigned int Ml[N] = {0xADE1A1DA, 0xADE1A1DA, 0xADE1A1DA, 0xADE1A1DA, 0xADE1A1DA};
    unsigned long long Mq[N] = {0xC1A551F1AB1EULL, 0xC1A551F1AB1EULL, 0xC1A551F1AB1EULL,
0xC1A551F1AB1EULL, 0xC1A551F1AB1EULL};

    // Вариант 17: x = 8/3 для float/double
    float Mfs[N];
    double Mfl[N];
    for (size_t i = 0; i < N; i++) {
        Mfs[i] = 8.0f / 3.0f;
        Mfl[i] = 8.0 / 3.0;
    }

    printf("================================================================================\n");

    // --- Mb ---
    PRINT_INT_ARRAY_ALL(Mb, unsigned char, signed char, "%02hhX", "%hhu", "%+hhd", sizeof(Mb[0]));
    printf("Введите новое значение Mb[2] (целое число 0..255): ");
    while (!input_element_int(&Mb[2], "%hhu")) {
        printf("Повторите ввод Mb[2]: ");
    }
    PRINT_INT_ARRAY_ALL(Mb, unsigned char, signed char, "%02hhX", "%hhu", "%+hhd", sizeof(Mb[0]));

    // --- Ms ---
    PRINT_INT_ARRAY_ALL(Ms, unsigned short, short, "%04hX", "%hu", "%+hd", sizeof(Ms[0]));
    printf("Введите новое значение Ms[2] (целое число 0..65535): ");
    while (!input_element_int(&Ms[2], "%hu")) {
```

```
 73.         printf("Повторите ввод Ms[2]: ");
 74.     }
 75.     PRINT_INT_ARRAY_ALL(Ms, unsigned short, short, "%04hX", "%hu", "%+hd", sizeof(Ms[0]));
 76.
 77.     // --- Ml ---
 78.     PRINT_INT_ARRAY_ALL(Ml, unsigned int, int, "%08X", "%u", "%+d", sizeof(Ml[0]));
 79.     printf("Введите новое значение Ml[2] (целое число 0..4294967295): ");
 80.     while (!input_element_int(&Ml[2], "%u")) {
 81.         printf("Повторите ввод Ml[2]: ");
 82.     }
 83.     PRINT_INT_ARRAY_ALL(Ml, unsigned int, int, "%08X", "%u", "%+d", sizeof(Ml[0]));
 84.
 85.     // --- Mq ---
 86.     PRINT_INT_ARRAY_ALL(Mq, unsigned long long, long long, "%016llX", "%llu", "%+lld",
sizeof(Mq[0]));
 87.     printf("Введите новое значение Mq[2] (целое число 0..18446744073709551615): ");
 88.     while (!input_element_int(&Mq[2], "%llu")) {
 89.         printf("Повторите ввод Mq[2]: ");
 90.     }
 91.     PRINT_INT_ARRAY_ALL(Mq, unsigned long long, long long, "%016llX", "%llu", "%+lld",
sizeof(Mq[0]));
 92.
 93.     // --- Mfs ---
 94.     PRINT_FLOAT_ARRAY_ALL(Mfs, float, "%A", "%.2e", "%f");
 95.     printf("Введите новое значение Mfs[2] (число с плавающей точкой): ");
 96.     while (!input_element_float(&Mfs[2], "%f")) {
 97.         printf("Повторите ввод Mfs[2]: ");
 98.     }
 99.     PRINT_FLOAT_ARRAY_ALL(Mfs, float, "%A", "%.2e", "%f");
100.
101.     // --- Mfl ---
102.     PRINT_FLOAT_ARRAY_ALL(Mfl, double, "%A", "%.2e", "%f");
103.     printf("Введите новое значение Mfl[2] (число с плавающей точкой): ");
104.     while (!input_element_float(&Mfl[2], "%lf")) {
105.         printf("Повторите ввод Mfl[2]: ");
106.     }
107.     PRINT_FLOAT_ARRAY_ALL(Mfl, double, "%A", "%.2e", "%f");
108.
109.     printf("\n==========================================================================\n");
110.
111.     printf("\n");
112. }
113.
114. #endif
```

Файл task4.c:

```
 1. #include <stdint.h>   // для uint64_t
 2. #include <stddef.h>   // для size_t, wchar_t, ptrdiff_t
 3. #include <stdbool.h>  // для bool
 4. #include <stdio.h>    // для printf и других функций ввода-вывода
 5.
 6. #include "task4.h"
 7.
 8. // Функция ввода элемента M[i] с проверкой для целочисленных типов
 9. int input_element_int(void *element, const char *fmt) {
10.     int ret = scanf(fmt, element);
11.     if (ret != 1) {
12.         puts("Ошибка ввода! Попробуйте ещё раз.");
13.         // Очищаем буфер ввода
14.         int ch;
15.         while ((ch = getchar()) != '\n' && ch != EOF);
16.         return 0;
17.     }
18.     return 1;
19. }
20.
21. // Функция ввода элемента M[i] с проверкой для плавающей точки
22. int input_element_float(void *element, const char *fmt) {
23.     int ret = scanf(fmt, element);
24.     if (ret != 1) {
25.         puts("Ошибка ввода! Попробуйте ещё раз.");
26.         int ch;
27.         while ((ch = getchar()) != '\n' && ch != EOF);
28.         return 0;
29.     }
30.     return 1;
31. }
```

Задание Л1.№5.

Для одного из массивов M (по варианту согласно таблице Л1.2) напечатайте адреса (формат p — указатели): — начала массива — M; — M[0], начального (нулевого) элемента массива — &(M[0]); — M[1], следующего за M[0] элемента массива — &(M[1]).

Сравните полученные значения между собой и с размером элемента массива M. Как расположены в памяти элементы массива?

Создайте статическую матрицу MM из R строк и N столбцов; элементы MM того же типа, что и элементы M.

Напечатайте адреса элементов MM[0][0], MM[0][1], MM[1][0], MM[1][1]. Как расположены в памяти элементы матрицы?

Как можно воспроизвести эту структуру на динамическом массиве M? Сколько памяти необходимо выделить? Как рассчитать индекс idx в M по номерам строки i и столбца j в MM? Требуется именно воспроизвести структуру матрицы в памяти, а не сымитировать синтаксис обращения $MM[i][j]$.

```
Задание №5
========================================================================
Адрес начала массива Ml (Ml): 0x7ffd95fe0c80
Адрес Ml[0]: 0x7ffd95fe0c80
Адрес Ml[1]: 0x7ffd95fe0c84
Размер элемента массива Ml: 4 байт

Адреса элементов матрицы MM:
MM[0][0]: 0x7ffd95fe0ca0
MM[0][1]: 0x7ffd95fe0ca4
MM[1][0]: 0x7ffd95fe0cb4
MM[1][1]: 0x7ffd95fe0cb8

Для динамического массива:
Адрес dynamic_MM: 0x5a8a874092a0
Адрес элемента dynamic_MM[0]: 0x5a8a874092a0
Адрес элемента dynamic_MM[1]: 0x5a8a874092a4
Адрес элемента dynamic_MM[N]: 0x5a8a874092b4 (начало второй строки)

========================================================================
```

Рис. 7: Вывод адресов элементов статического и динамического массива Ml

Листинг:

Файл task5.c:

```c
#ifndef task5_H
#define task5_H

#include <stdio.h>
#include <stdlib.h>

#define N 5
#define R 3

void run_task5()
{
    printf("\nЗадание №5\n");

    // Инициализация массива
    unsigned int Ml[N] = {0xADE1A1DA, 0xADE1A1DA, 0xADE1A1DA, 0xADE1A1DA, 0xADE1A1DA};

    printf("================================================================================\n");


    printf("Адрес начала массива Ml (Ml): %p\n", (void*)Ml);
    printf("Адрес Ml[0]: %p\n", (void*)&Ml[0]);
    printf("Адрес Ml[1]: %p\n", (void*)&Ml[1]);

    printf("Размер элемента массива Ml: %zu байт\n", sizeof(Ml[0]));

    // Создаём статическую матрицу MM типа unsigned int
    unsigned int MM[R][N] = {0};

    printf("\nАдреса элементов матрицы MM:\n");
    printf("MM[0][0]: %p\n", (void*)&MM[0][0]);
    printf("MM[0][1]: %p\n", (void*)&MM[0][1]);
    printf("MM[1][0]: %p\n", (void*)&MM[1][0]);
    printf("MM[1][1]: %p\n", (void*)&MM[1][1]);

    /*
    Анализ:
    - Адрес Ml и &Ml[0] совпадают, это начало массива.
    - Адрес Ml[1] должен быть на sizeof(unsigned int) байт дальше.
    - В памяти элементы массива расположены подряд (contiguous).
    - Аналогично, в матрице MM элементы каждой строки расположены подряд.
    - Адрес MM[0][1] на sizeof(unsigned int) байт дальше MM[0][0].
    - Адрес MM[1][0] находится на sizeof(unsigned int)*N байт дальше MM[0][0].
    */

    // Как воспроизвести структуру матрицы на динамическом массиве:
    // Нужно выделить память под R*N элементов типа unsigned int:
    unsigned int *dynamic_MM = (unsigned int*)malloc(R * N * sizeof(unsigned int));
    if (!dynamic_MM) {
        puts("Ошибка выделения памяти");
        return;
    }

    // Индекс в одномерном массиве для элемента [i][j]:
    // idx = i * N + j;

    printf("\nДля динамического массива:\n");
    printf("Адрес dynamic_MM: %p\n", (void*)dynamic_MM);
    printf("Адрес элемента dynamic_MM[0]: %p\n", (void*)&dynamic_MM[0]);
    printf("Адрес элемента dynamic_MM[1]: %p\n", (void*)&dynamic_MM[1]);
    printf("Адрес элемента dynamic_MM[N]: %p (начало второй строки)\n", (void*)&dynamic_MM[N]);

    // Освобождаем память
    free(dynamic_MM);

    printf("\n================================================================================\n");

    printf("\n");
}

#endif
```

Задание Л1.№6.

Для одного из массивов M (по варианту согласно таблице Л1.2) введите с клавиатуры новое значение всех пяти элементов при помощи одного вызова функции libc $scanf()$.

Проанализировав возвращённое $scanf()$ значение, определите корректность ввода; при необходимости отобразите сообщение о количестве введённых и не введённых элементов.

Выведите массив на экран до и после ввода; убедитесь, что количество изменившихся элементов соответствует ожиданиям.

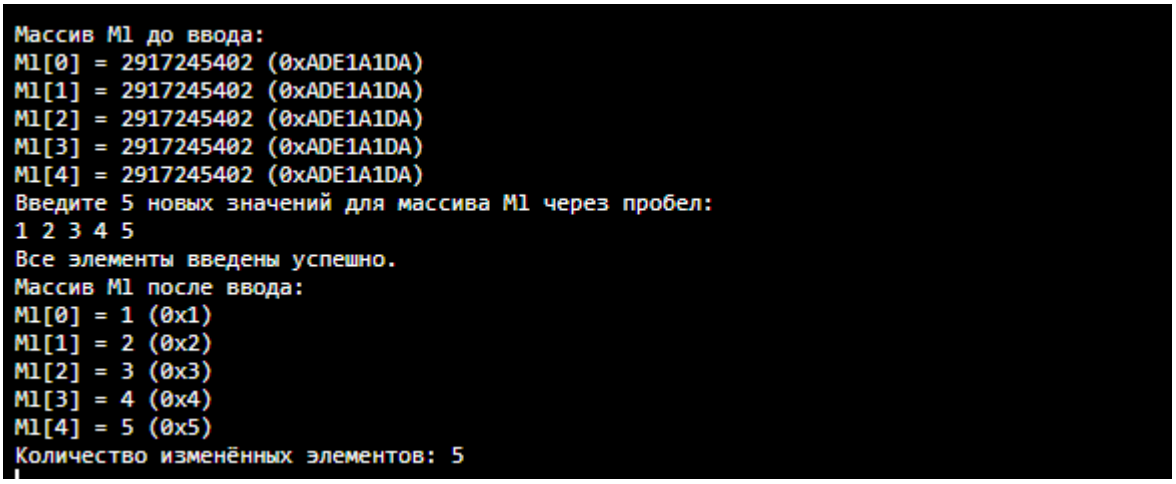

Рис. 8: Результаты изменения массива путём ввода значений с клавиатуры

Листинг:

Файл task6.c:

```c
#include <stdio.h>

#define N 5

void run_task6(){
    unsigned int Ml[N] = {0xADE1A1DA, 0xADE1A1DA, 0xADE1A1DA, 0xADE1A1DA, 0xADE1A1DA};
    unsigned int Ml_before[N];

    // Копируем исходный массив
    for (int i = 0; i < N; ++i) {
        Ml_before[i] = Ml[i];
    }

    printf("Массив Ml до ввода:\n");
    for (int i = 0; i < N; ++i) {
        printf("Ml[%d] = %u (0x%X)\n", i, Ml[i], Ml[i]);
    }

    printf("Введите %d новых значений для массива Ml через пробел:\n", N);

    int count = scanf("%u %u %u %u %u", &Ml[0], &Ml[1], &Ml[2], &Ml[3], &Ml[4]);

    if (count == EOF) {
        puts("Ошибка ввода: достигнут конец файла или ошибка.");
        return;
    }

    if (count < N) {
        printf("Введено %d элементов, не введено %d элементов.\n", count, N - count);
    } else {
        puts("Все элементы введены успешно.");
    }

    printf("Массив Ml после ввода:\n");
    for (int i = 0; i < N; ++i) {
        printf("Ml[%d] = %u (0x%X)\n", i, Ml[i], Ml[i]);
    }

    // Подсчёт изменённых элементов
    int changed = 0;
    for (int i = 0; i < N; ++i) {
```

```
42.         if (Ml[i] != Ml_before[i]) {
43.             changed++;
44.         }
45.     }
46.
47.     printf("Количество изменённых элементов: %d\n", changed);
48.
49.     if (changed != count) {
50.         printf("Внимание: количество изменённых элементов (%d) не совпадает с количеством введённых
(%d).\n", changed, count);
51.     }
52. }
53.
```

Задание Л1.№7.

Бонус +2 балла для пар, обязательное для троек.

Введите с клавиатуры (каждую строку — одним вызовом $scanf()$):

a) слово (строку без пробелов) $s1$ (формат s без модификаторов);

b) слово $s2$ таким образом, чтобы принимающий его буфер гарантированно не переполнился: если буфер длины k — вводить не более $k - 1$ символов (ширина поля ввода задаётся аналогично ширине поля вывода);

c) строку, возможно, содержащую пробелы $s3$ (формат [] — регулярное выражение Perl).

Выведите на экран при помощи функций libc строки «$***s1***$», «$***s2***$», «$***s3***$» (между звёздочками должна быть введённые строки, а не литералы s1-s3) и убедитесь, что ввод корректен.

```
asddgffg
Введите слово s1 (без пробелов):

Введите слово s2 (не более 19 символов):
qwetrrytuyoiop[op[]p]asfdsdgffhghjjlkl'zxcvxvbnvbmj.,/
Введите строку s3 (с пробелами):
sadsad fghgfhfgfdoi qwewqe
***asddgffg***
***qwetrrytuyoiop[op[]***
***sadsad fghgfhfgfdoi qwewqe***
```

Рис. 9: Ввод и вывод строк с различными ограничениями

Листинг:

Файл task7.c:

```c
#include <stdio.h>

#define K 20  // длина буфера для s2

void run_task7() {
    char s1[100];   // для слова без ограничений по длине (предположим 100)
    char s2[K];     // буфер длиной K
    char s3[200];   // для строки с пробелами

    // Ввод s1 (слово без пробелов)
    printf("Введите слово s1 (без пробелов):\n");
    if (scanf("%99s", s1) != 1) {
        printf("Ошибка ввода s1\n");
        return;
    }

    // Ввод s2 (слово с ограничением длины: не более K-1 символов)
    printf("Введите слово s2 (не более %d символов):\n", K-1);
    // Формат %<N>s ограничит ввод до N символов
    char format_s2[10];
    snprintf(format_s2, sizeof(format_s2), "%%%ds", K-1);
    if (scanf(format_s2, s2) != 1) {
        printf("Ошибка ввода s2\n");
        return;
    }

    // Очистка остатка строки после ввода s2 (если есть)
    int c;
    while ((c = getchar()) != '\n' && c != EOF);

    // Ввод s3 (строка с пробелами)
    printf("Введите строку s3 (с пробелами):\n");
    if (scanf("%199[^\n]", s3) != 1) {
        printf("Ошибка ввода s3\n");
        return;
    }

    // Вывод строк с обрамлением ***
    printf("***%s***\n", s1);
    printf("***%s***\n", s2);
    printf("***%s***\n", s3);
}
```