

**МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра информационных систем**

**ОТЧЕТ
по практической работе №1
по дисциплине «Программирование»
Тема: «Типы данных и их внутреннее представление в памяти»**

Студентка гр. 0324

Преподаватель

Колков К.В.

Глущенко А.Г

Санкт-Петербург
2020

Цель работы.

Знакомство с внутренним представлением различных типов данных, используемых компьютером при их обработке.

Основные теоретические положения.

Внутреннее представление величин целого типа – целое число в двоичном коде. При использовании спецификатора signed старший бит числа интерпретируется как знаковый (0 – положительное число, 1 – отрицательное). Для кодирования целых чисел со знаком применяется прямой, обратный и дополнительный коды.

Представление положительных и отрицательных чисел в прямом, обратном и дополнительном кодах отличается. В прямом коде в знаковый разряд помещается цифра 1, а в разряды цифровой части числа – двоичный код его абсолютной величины. Прямой код числа -3 (для 16-разрядного процессора):

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Знак числа

Обратный код получается инвертированием всех цифр двоичного кода абсолютной величины, включая разряд знака: нули заменяются единицами, единицы – нулями. Прямой код можно преобразовать в обратный, инвертировав все значения всех битов (кроме знакового). Обратный код числа -3 :

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0

Знак числа

Дополнительный код получается образованием обратного кода с последующим прибавлением единицы к его младшему разряду. Дополнительный код числа -3 :

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1

Знак числа

Увидеть, каким образом тип данных представляется на компьютере, можно при помощи логических операций: побитового сдвига (<<) и поразрядной конъюнкции (&).

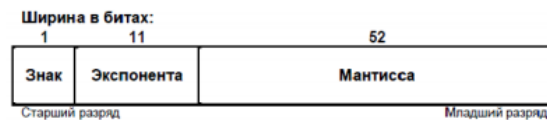
```
putchar(value & mask ? '1' : '0'); // если 1, то возвращается 1, иначе 0
value <<= 1; // побитовый сдвиг влево на 1 бит
```

Putchar возвращает один символ в консоль. Альтернатива - cout. В представленном способе, маска - то, с чем сравнивается значение. И побитовый сдвиг применяется для value. Таким образом 1 бит будет сравниваться с каждым битом числа. Альтернатива - побитовый сдвиг вправо, но при этом нужно проводить данную операцию не над значением(единицей), а над маской (исходным числом, битовое представление которого нужно получить).

При сдвиге вправо для чисел без знака позиции битов, освобожденные при операции сдвига, заполняются нулями. Для чисел со знаком бит знака используется для заполнения освобожденных позиций битов. Другими словами, если число 25 является положительным, используется 0, если число является отрицательным, используется 1. При сдвиге влево позиции битов, освобожденных при операции сдвига, заполняются нулями. Сдвиг влево является логическим сдвигом (биты, сдвигаемые с конца, отбрасываются, включая бит знака).

Вещественные типы данных хранятся в памяти компьютера иначе, чем целочисленные. Внутреннее представление вещественного числа состоит из двух частей – мантиссы и порядка.

Для 32-разрядного процессора для float под мантиссу отводится 23 бита, под экспоненту – 8, под знак – 1. Для double под мантиссу отводится 52 бита, под экспоненту – 11, под знак – 1:



Увидеть, каким образом вещественные типы данных представляются в компьютере немного сложнее. Логические операции, которые использовались с

int, для вещественных типов данных не подходят. Но это ограничение можно легко обойти, используя объединения.

Объединения – это две или более переменных расположенных по одному адресу (они разделяют одну и ту же память). Объединения определяются с использованием ключевого слова union. Объединения не могут хранить одновременно несколько различных значений, они позволяют интерпретировать несколькими различными способами содержимое одной и той же области памяти.

С объединениями нужно быть осторожным. Вся работа с памятью требует грамотного подхода. Более подробно с объединениями можно будет ознакомиться при изучении структур. Пока что объединения будут служить инструментом для работы с float и double.

```
#include <iostream>
using namespace std;
int main()
{
    union {
        int tool;
        float numb_f = 3.14;
    };
    cout << tool << endl; // 1078523331
    cout << numb_f << endl; // 3.14
    tool = tool >> 1; // побитовый сдвиг вправо
    cout << tool << endl; // 5392261665
    cout << numb_f; // 1.3932e-19
    return 0;
}
```

Подобные манипуляции возможны благодаря тому, что int и float занимают 4 байта. Проводя манипуляции над tool, мы изменяем значение numb_f. Таким образом, алгоритм, который использовался для представления в памяти int может использоваться и для float.

Алгоритм представления double немного отличается. Под вещественное число с двойной точностью отводится 8 байт, в то время как под int всего 4 байта. Но и это ограничение можно легко обойти. Так как данные любой

линейной структуры в память записываются последовательно (друг за другом), можно использовать массив из двух `int`, под который будет отведено 8 байт.

Постановка задачи.

Нужно разработать алгоритм и написать программу, которая позволяет:

- 1) Вывести, сколько памяти (в байтах) на вашем компьютере отводится под различные типы данных со спецификаторами и без: `int`, `short int`, `long int`, `float`, `double`, `long double`, `char` и `bool`.
- 2) Вывести на экран двоичное представление в памяти (все разряды) целого числа. При выводе необходимо визуально обозначить знаковый разряд и значащие разряды отступами или цветом.
- 3) Вывести на экран двоичное представление в памяти (все разряды) типа `float`. При выводе необходимо визуально обозначить знаковый разряд мантиссы, знаковый разряд порядка (если есть), мантиссу и порядок.
- 4) Вывести на экран двоичное представление в памяти (все разряды) типа `double`. При выводе необходимо визуально обозначить знаковый разряд мантиссы, знаковый разряд порядка (если есть), мантиссу и порядок.

Выполнение работы.

Для решения поставленных задач я написал программу на языке программирования C++, код которой будет представлен ниже.

В первой задаче был использован оператор `sizeof`, который помогает вычислить размер типа данных в байтах и выводит результат в консоль.

Во второй задаче для двоичного представления целого числа я создал маску и воспользовался побитовым сдвигом и побитовой конъюнкцией, в результате чего получил введенное число в двоичной системе счисления.

В третьей задаче я использовал `union`, для того, чтобы смотреть на значение `float` в памяти и прочитать его через `int`. Распечатка в консоль

осуществлялась через введение новой переменной (маски), побитового перемножения значения на маску со сдвигом этого значения.

В четвертой задаче double занимает в 2 раза больше битов в памяти, чем int. Чтобы «подсмотреть» в ячейку через int, сделан массив из 2-х int, который стал занимать 8 байт. Появилась возможность прочитать значение, и обработать его, как в пункте 3.

```
//
// main.cpp
// StepikLab
//
// Created by Кирилл Колков on 30.09.2020.
//

#include <iostream>
using namespace std;

int main() {

    // Вывести, сколько памяти (в байтах) на вашем компьютере
    // отводится под различные типы данных со спецификаторами и без: int,
    // short int, long int, float, double, long double, char и bool.

    cout << "Под тип Integer в памяти отводится " << sizeof(int)
    << " байт\n";
    cout << "Под тип Short Integer в памяти отводится " <<
    sizeof(short int) << " байт\n";
    cout << "Под тип Long Integer в памяти отводится " <<
    sizeof(long int) << " байт\n";
    cout << "Под тип Float в памяти отводится " << sizeof(float)
    << " байт\n";
    cout << "Под тип Double в памяти отводится " <<
    sizeof(double) << " байт\n";
    cout << "Под тип Long Double в памяти отводится " <<
    sizeof(long double) << " байт\n";
    cout << "Под тип Char в памяти отводится " << sizeof(char) <<
    " байт\n";
    cout << "Под тип Bool в памяти отводится " << sizeof(bool) <<
    " байт\n" << "\n";

    // Вывести на экран двоичное представление в памяти (все
    // разряды) целого числа. При выводе необходимо визуально обозначить
    // знаковый разряд и значащие разряды отступами или цветом.

    int a;
    unsigned int mask = 1 << 31;
```

```

cout << "Для вывода на экран двоичного представления целого
числа в памяти, введите это число: ";
cin >> a;

```

```

cout << "Вот результат: ";
for (int i = 0; i <= 31; i++) {
    if (i == 1) {
        cout << " ";
    }
    putchar(a & mask ? '1' : '0');
    mask >>= 1;
}
cout << "\n";

```

// Вывести на экран двоичное представление в памяти (все разряды) типа float. При выводе необходимо визуально обозначить знаковый разряд мантиссы, знаковый разряд порядка (если есть), мантиссу и порядок.

```

union {
    int forFloat;
    float b;
};

unsigned int floatMask = 1 << 31;

cout << "Для вывода на экран двоичного представления типа
float в памяти, введите это число: ";
cin >> b;

for (int j = 0; j <= 31; j++) {
    if (j == 1 || j == 9) {
        cout << " ";
    }
    putchar(forFloat & floatMask ? '1' : '0');
    forFloat <<= 1;
}
cout << "\n";

```

// Вывести на экран двоичное представление в памяти (все разряды) типа double. При выводе необходимо визуально обозначить знаковый разряд мантиссы, знаковый разряд порядка (если есть), мантиссу и порядок. (*)

// Ход мысли: Под Дабл в памяти отводится 8 байт, то есть 64 бита. Возникает проблема – мы не можем через Интеджер подглядеть в ячейку с Даблом, так как под Инт выделяется ровно в 2 раза меньше (4 байта).

// Вариантов сходу два: 1. Создать массив и положить в него 2 интджера, придется работать с левой и правой половиной массива. 2. Заюзать Long Integer – тут и Int и 8 байт, через него и глянем в Дабл. Второй вариант нравится больше, но смущает то, то long int не int, так что выбираем 1 вариант.

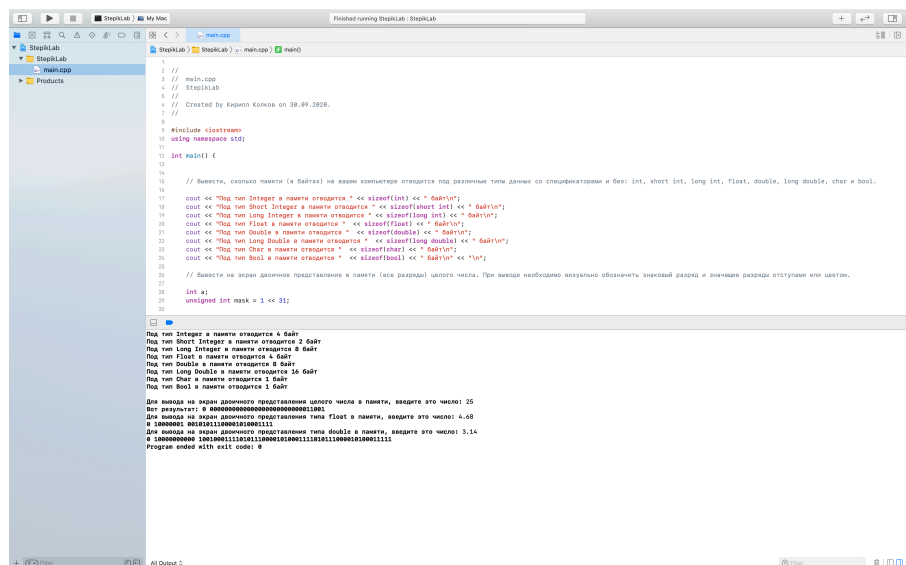
```
union {
    int arr[2];
    double c;
};

unsigned int doubleMask = 1 << 31;

cout << "Для вывода на экран двоичного представления типа
double в памяти, введите это число: ";
cin >> c;

for (int j = 0; j <= 31; j++) {
    if (j == 1 || j == 12) {
        cout << " ";
    }
    putchar(arr[1] & doubleMask ? '1' : '0');
    arr[1] <= 1;
}
for (int j = 0; j <= 31; j++) {
    putchar(arr[0] & doubleMask ? '1' : '0');
    arr[0] <= 1;
}
cout << "\n";
}
```

Результат работы программы.



```
1 //
2 //
3 // main.cpp
4 // StepLab
5 // Created by Kirill Konovalov on 28-09-2020.
6 //
7 //
8 #include <iostream>
9 using namespace std;
10
11 int main() {
12
13     // Вывести, сколько памяти (в байтах) на каждом компьютере отводится под различные типы данных со спецификаторами и без: int, short int, long int, float, double, long double, char и bool.
14
15     cout << "Под тип Integer в памяти отводится " << sizeof(int) << " байт\n";
16     cout << "Под тип Short Integer в памяти отводится " << sizeof(short int) << " байт\n";
17     cout << "Под тип Long Integer в памяти отводится " << sizeof(long int) << " байт\n";
18     cout << "Под тип Float в памяти отводится " << sizeof(float) << " байт\n";
19     cout << "Под тип Double в памяти отводится " << sizeof(double) << " байт\n";
20     cout << "Под тип Long Double в памяти отводится " << sizeof(long double) << " байт\n";
21     cout << "Под тип Char в памяти отводится " << sizeof(char) << " байт\n";
22     cout << "Под тип Bool в памяти отводится " << sizeof(bool) << " байт\n";
23
24     // Вывести на экран двоичное представление в памяти (все разряды) целого числа. При выводе необходимо явно указывать знаковый разряд и значащие разряды отступами или цветом.
25
26     int a;
27     unsigned int mask = 1 << 31;
28
29     Под тип Integer в памяти отводится 4 байт
30     Под тип Short Integer в памяти отводится 2 байт
31     Под тип Long Integer в памяти отводится 8 байт
32     Под тип Float в памяти отводится 4 байт
33     Под тип Double в памяти отводится 8 байт
34     Под тип Long Double в памяти отводится 16 байт
35     Под тип Char в памяти отводится 1 байт
36     Под тип Bool в памяти отводится 1 байт
37
38     Для вывода на экран двоичного представления целого числа в памяти, введите это число: 25
39     Введите int: 25
40     Для вывода на экран двоичного представления типа float в памяти, введите это число: 4.68
41     Введите float: 4.68
42     Для вывода на экран двоичного представления типа double в памяти, введите это число: 3.14
43     Введите double: 3.14
44     Program ended with exit code: 0
```

Вывод.

Полученные результаты программы соответствуют с реальным представлением в памяти целого числа, типа float, а также типа double, что и требовалось от написанной программы.