**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

|  |  |  |
| --- | --- | --- |
| Студент(ка)гр. |  | Лопаник А. А. 1323 |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2021

**Цель работы.**

* Изучение и демонстрирование представления данных разного типа в памяти компьютера;
* Получение практических навыков по работе с переменными и циклами;
* Практическое использование масок, логических операций и объединений.

**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):



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:



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



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 (исходым 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, использовав 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 острожным. Вся работа с памятью требует грамотного подхода. Более подробно с объединениями можно будет ознакомиться при изучении структур. Пока что объедения 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;

}

**Постановка задачи.**

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.**

**Рабочий код**

#include <iostream>

using namespace std;

int main()

{

setlocale(0, "");

cout << "Объём памяти в байтах, отводящийся под различные типы данных: \n";

cout << "int: " << sizeof(int) << "\n";

cout << "short int: " << sizeof(short int) << "\n";

cout << "long int: " << sizeof(long int) << "\n";

cout << "float: " << sizeof(float) << "\n";

cout << "double: " << sizeof(double) << "\n";

cout << "long double: " << sizeof(long double) << "\n";

cout << "char: " << sizeof(char) << "\n";

cout << "bool: " << sizeof(bool) << "\n";

int value;

cout << "Введите целое число: ";

cin >> value;

unsigned int order = sizeof(value) \* 8;

unsigned int mask = 1U <<(order - 1);

for (int i = 1; i <= order; i++)

{

putchar(value & mask ? '1' : '0');

if (i != 0 && i % 8 == 0)

putchar(' ');

if (i == 1)

putchar(' ');

mask >>= 1;

}

union {

int value\_f;

float numb\_f;

};

cout << "\nВведите вещественное число: \n";

cin >> numb\_f;

unsigned int order\_f = sizeof(value\_f) \* 8;

unsigned int mask\_f = 1U <<(order\_f - 1);

for (int i = 1; i <= order\_f; i++)

{

putchar(value\_f & mask\_f ? '1' : '0');

if (i == 1 || i == 9)

putchar(' ');

mask\_f>>= 1;

}

union {

long long int value\_d;

double numb\_d;

};

cout << "\nВведите число типа double: \n";

cin >> numb\_d;

unsigned int order\_d = sizeof(value\_d) \* 8;

unsigned long long int mask\_d = 1ULL <<(order\_d - 1);

for (int i = 1; i <= order\_d; ++i)

{

putchar(value\_d & mask\_d ? '1' : '0');

if (i == 1 || i == 12)

putchar(' ');

mask\_d >>= 1;

}

return 0;

}

|  |  |
| --- | --- |
| Ввод пользователем и обработка данных | Работа алгоритма и вывод на экран |
| Вывод количества байтов под типы данных | |
| При запуске программы перед пользователем появляется окно, в котором указано кол-во памяти под различные типы данных |  |
| Вывод целого числа | |
| При вводе пользователем целого числа показывается его двоичное представление в памяти |  |
| Вывод float | |
| При вводе пользователем вещественного числа типа float на экран выводится его двоичное представление в памяти |  |
| Вывод double | |
| При вводе пользователем числа типа double на экран выводится его двоичное представление в памяти |  |

**Выводы.**

В ходе работы мы научились определять кол-во байт, занимаемых в памяти определенными типами данных, а также выводить их двоичное представление на экран с визуальным обозначением знакового разряда, значащих разрядов и пр.