**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 гр. 3373 | Журухин Н.А. |  |
| Преподаватель | Глущенко А. Г. |  |

Санкт-Петербург

2023

**Цель работы.**

Научиться работать с различными типами данных, выводить их двоичное представление с помощью побитового сдвига и поразрядной конъюнкции, узнать, сколько памяти под них выделяется, а также что такое спецификаторы, объединения, прямой, обратный и дополнительный коды. Разработать алгоритм и написать программу с помощью полученных знаний.

**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 |

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 |

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 |

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

 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.

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.

 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.

Алгоритма представления double немного отличается. Под вещественное число с двойной точностью отводиться 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.**

Разработать алгоритм и написать программу, которая позволяет:

* Вывести, сколько памяти (в байтах) на компьютере отводится под различные типы данных со спецификаторами и без: int, short int, long int, float, double, long double, char и bool.
* Вывести на экран двоичное представление в памяти (все разряды) целого числа. При выводе визуально обозначить знаковый разряд и значащие разряды отступами или цветом.
* Вывести на экран двоичное представление в памяти (все разряды) типа float. При выводе визуально обозначить знаковый разряд числа, мантиссу и порядок.
* Вывести на экран двоичное представление в памяти (все разряды) типа double. При выводе визуально обозначить знаковый разряд числа, мантиссу и порядок.

**Выполнение работы.**

Код программы представлен в приложении А.

Блок описания кода и использованных алгоритмов

Программа выполняет 4 задачи. Сначала она выводит количество памяти, выделяющиеся под типы данных int, short int, long int, float, double, long double, char и bool.

 Далее программа просит пользователя ввести число типа integer и с помощью побитового сдвига заданной маски вправо производит поразрядную конъюнкцию маски с числом и выводит двоичное представление числа с обозначенными отступами значащими разрядами и отделенным знаком числа.



Следом с помощью объединения по одному адресу располагаются две переменные типов float и integer. Программа просит пользователя ввести число типа float. С помощью побитового сдвига заданной маски вправо производится поразрядная конъюнкция маски с числом integer и выводится двоичное представление числа float, где отступами обозначены знак числа, мантисса и порядок.

 Далее с помощью объединения по одному адресу располагаются две переменные: double и массив из двух элементов integer. Программа просит пользователя ввести число типа double. С помощью побитового сдвига дважды заданной маски вправо производится поразрядная конъюнкция маски с каждым элементом массива и выводится двоичное представление числа double, где отступами обозначены знак числа, мантисса и порядок.

 Выполнение программы заканчивается.

Блок скриншотов работы программы

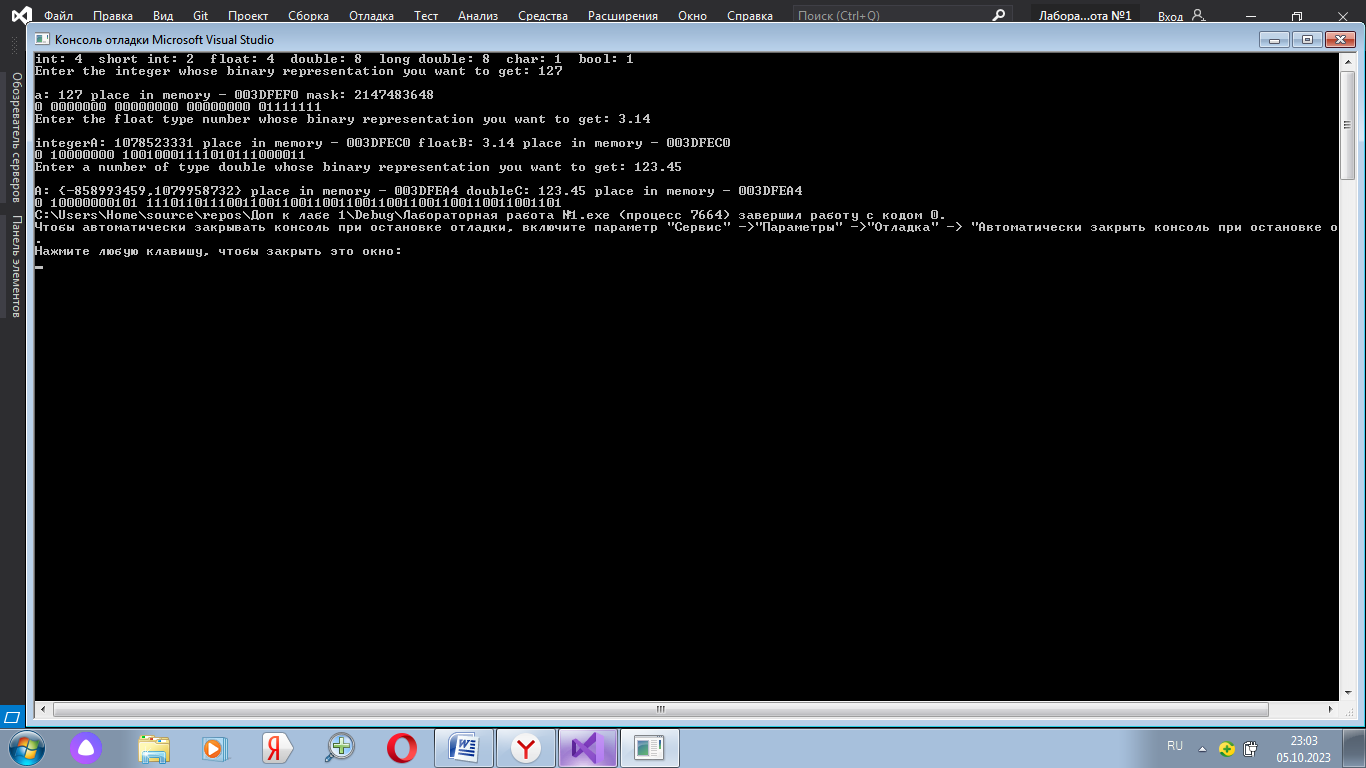


Рис.1 Выполнение программы с введенными положительными числами

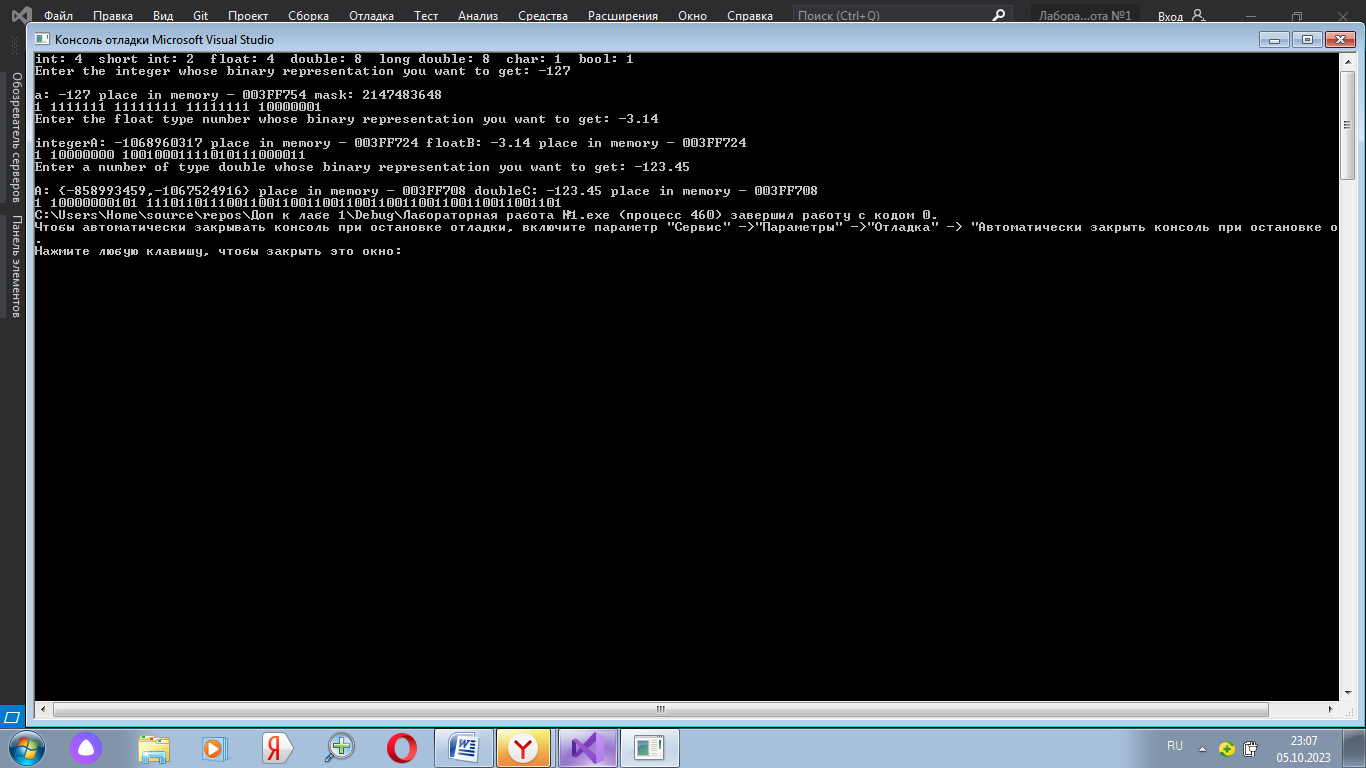


Рис.2 Выполнение программы с введенными отрицательными числами

Блок таблицы с тестовыми данными

|  |  |  |
| --- | --- | --- |
| тип | число | двоичное представление |
| Integer | 127 | 0 0000000 00000000 00000000 01111111 |
| Integer | -127 | 1 1111111 11111111 11111111 10000001 |
| Float | 3.14 | 0 10000000 10010001111010111000011 |
| Float | -3.14 | 1 10000000 10010001111010111000011 |
| Double | 123.45 | 0 10000000101 1110110111001100110011001100110011001100110011001101 |
| Double | -123.45 | 1 10000000101 1110110111001100110011001100110011001100110011001101 |

**Выводы.**

Выполнив данную лабораторную работу, я узнал, сколько памяти в моей среде программирования занимают различные типы данных, научился выводить двоичное представление типов данных integer, float, double, работать с объединениями.

Приложение А

рабочий код

#include <iostream>

using namespace std;

int main() {

setlocale (0, "");

cout << "int: " << sizeof(int) << " short int: " << sizeof(short int) << " float: " << sizeof(float) << " double: " << sizeof(double) << " long double: " << sizeof(long double) << " char: " << sizeof(char) << " bool: " << sizeof(bool);

cout << "\n";

cout << "Enter the integer whose binary representation you want to get: ";

int a;

cin >> a;

cout << "\n";

int order = sizeof(int) \* 8;

unsigned int mask = 1 << order - 1;

cout << "a:" << " " << a << " " << "place in memory" << " " << '-' << " " << &a << " " << "mask:" << " " << mask << " " << "\n";

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

{

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

mask >>= 1;

if (i % 8 == 0)

{

putchar(' ');

}

if (i % order - 1 == 0)

{

putchar(' ');

}

}

union {

int integerA;

float floatB;

};

cout << "\n";

cout << "Enter the float type number whose binary representation you want to get: ";

cin >> floatB;

mask = 1 << order - 1;

cout << "\n";

cout << "integerA:" << " " << integerA << " " << "place in memory" << " " << '-' << " " << &integerA << " " << "floatB:" << " " << floatB << " " << "place in memory" << " " << '-' << " " << &floatB << "\n";

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

{

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

mask >>= 1;

if ((i % order - 1 == 0) || (i % order - 9 == 0))

{

putchar(' ');

}

}

union {

int A[2];

double doubleC;

};

cout << "\n";

cout << "Enter a number of type double whose binary representation you want to get: ";

cin >> doubleC;

cout << "\n";

cout << "A:" << " " << "{" << A[0] << "," << A[1] << "}" << " " << "place in memory" << " " << "-" << " " << &A << " " << "doubleC:" << " " << doubleC << " " << "place in memory" << " " << "-" << " " << &doubleC << "\n";

int order2 = sizeof(double) \* 8;

for (int j = 1; j >= 0; j--)

{

mask = 1 << order - 1;

for (int i = 1; i <= order2/2; i++) {

putchar(A[j] & mask ? '1' : '0');

mask >>= 1;

if (((j == 1) && (i % order - 1 == 0)) || ((j == 1) && (i % order - 12 == 0)))

{

putchar(' ');

}

}

}

}