**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, определяемые пользователем. Структуры

|  |  |  |
| --- | --- | --- |
| Студент гр. 5376 |  | Сафаров Е.В. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2025

**Цель работы.**

Изучение работы с побитовыми операциями. Изучение строения типов данных на языке с++.

**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.

**Постановка задачи.**

Разработать алгоритм и написать программу, которая позволяет:

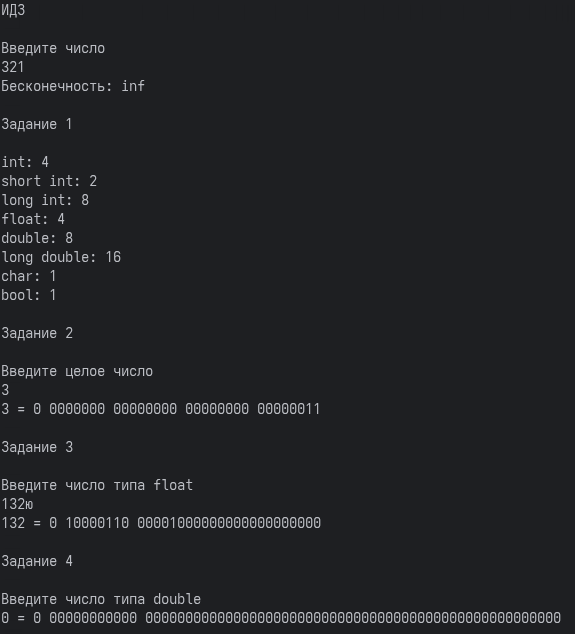
1) Вывести, сколько памяти (в байтах) на вашем компьютере отводится под различные типы данных со спецификаторами и без: int, short int, long int, float, double, long double, char и bool.

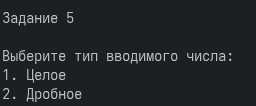
2) Вывести на экран двоичное представление в памяти (все разряды) целого числа. При выводе необходимо визуально обозначить знаковый разряд и значащие разряды отступами или цветом.

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. (\*)

5) Реализовать возможность произвольного изменения любого бита в введенном числе (для всех типов данных) с использованием логических операций.

**Выполнение работы.**



**Выводы.**

Написали программу, которая выполняет все поставленные задачи.

Приложение А

рабочий код

#include <clocale>

#include <iostream>

#include <ostream>

int main()

{

setlocale(0, "");

{

//17 Создание бесконечности. Установите все биты порядка в 1, а все биты мантиссы — в 0. Знаковый бит оставьте без изменения. Вы получите +inf или -inf.

union FloatUnion

{

float f;

int u;

};

std::cout << "ИДЗ\n\nВведите число\n";

unsigned int inf = 0b1111111100000000000000000000000;

unsigned int mask = 0b1111111111111111111111111111111;

FloatUnion value;

std::cin >> value.f;

value.u = (value.u & ~mask) | (inf & mask);

std::cout << "Бесконечность: "<< value.f;

}

std::cout << "\n\nЗадание 1\n\n";

{

std::cout

<< "int: " << sizeof(int)

<< "\nshort int: " << sizeof(short int)

<< "\nlong int: " << sizeof(long int)

<< "\nfloat: " << sizeof(float)

<< "\ndouble: " << sizeof(double)

<< "\nlong double: " << sizeof(long double)

<< "\nchar: " << sizeof(char)

<< "\nbool: " << sizeof(bool);

}

std::cout << "\n\nЗадание 2\n\n";

{

std::cout << "Введите целое число\n";

int value;

std::cin >> value;

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

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

std::cout << value << " = ";

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

{

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

value <<= 1;

if (i % 8 == 0 || i % order - 1 == 0)

{

putchar(' ');

}

}

}

std::cout << "\n\nЗадание 3\n\n";

{

union FloatUnion

{

float f;

int u;

};

std::cout << "Введите число типа float\n";

FloatUnion fu;

std::cin >> fu.f;

int order = (sizeof(float) \* 8);

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

std::cout << fu.f << " = ";

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

{

putchar(fu.u & mask ? '1' : '0');

fu.u <<= 1;

if (i == 1 || i == 9) putchar(' ');

}

}

std::cout << "\n\nЗадание 4\n\n";

{

union DoubleUnion

{

double d;

unsigned long int u;

};

std::cout << "Введите число типа double\n";

DoubleUnion du;

std::cin >> du.d;

int order = (sizeof(double) \* 8);

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

std::cout << du.d << " = ";

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

{

putchar(du.u & mask ? '1' : '0');

mask >>= 1;

if (i == 1 || i == 12) putchar(' ');

}

}

std::cout << "\n\nЗадание 5\n\n";

{

union DoubleIntUnion

{

double d;

long int u;

};

std::cout << "Выберите тип вводимого числа:\n1. Целое\n2. Дробное\n";

int TypeChoice;

std::cin >> TypeChoice;

DoubleIntUnion diu;

if (TypeChoice == 1)

{

std::cout << "Введите целое число\n";

std::cin >> diu.u;

int order = (sizeof(long int) \* 8);

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

std::cout << diu.u << " = ";

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

{

putchar(diu.u & mask ? '1' : '0');

mask >>= 1;

if (i % 8 == 0 || i % order - 1 == 0)

{

putchar(' ');

}

}

std::cout << "\n\nКакой бит изменить? (считая справа)\n";

int BitChoice;

std::cin >> BitChoice;

mask = 1;

diu.u = diu.u ^ (mask << BitChoice);

std::cout << diu.u << " = ";

mask = 1ULL << (order - 1);

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

{

putchar(diu.u & mask ? '1' : '0');

mask >>= 1;

if (i % 8 == 0 || i % order - 1 == 0)

{

putchar(' ');

}

}

}

if (TypeChoice == 2)

{

std::cout << "Введите дробное число\n";

std::cin >> diu.d;

int order = (sizeof(double) \* 8);

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

std::cout << diu.d << " = ";

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

{

putchar(diu.u & mask ? '1' : '0');

mask >>= 1;

if (i == 1 || i == 12) putchar(' ');

}

std::cout << "\nКакой бит изменить? (считая справа)\n";

int BitChoice;

std::cin >> BitChoice;

mask = 1;

diu.u = diu.u ^ (mask << BitChoice);

std::cout << diu.d << " = ";

mask = 1ULL << (order - 1);

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

{

putchar(diu.u & mask ? '1' : '0');

mask >>= 1;

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

{

putchar(' ');

}

}

}

std::cout << "\n";

return 0;

}

}