**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. |  | Тубшинов В.Т. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2022

**Цель работы.**

Изучение типов данных, представления типов данных в памяти компьютера.

**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.

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).

Дополнительный код получается образованием обратного кода с последующим прибавлением единицы к его младшему разряду.

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

Пример кода:

putchar(value & mask ? '1' : '0'); // если 1, то возвращается 1, иначе 0

value <<= 1; // побитовый сдвиг влево на 1 бит

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).

В памяти компьютера под разные типы данных отводится разное количество бит. Для того, чтобы узнать сколько в памяти компьютера занимает тот или иной тип данных надо воспользоваться унарным оператором sizeof(<тип данных>).

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.

#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 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.

#include <iostream>

using namespace std;

int main()

{

int value = -127; // Значение числа

unsigned int order = 32; // Количество разрядов

unsigned int mask = 1 << order – 1; // Маска побитового сравнения

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

{

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

value <<= 1; // Побитовый сдвиг числа

if (i % 8 == 0)

{

putchar(' ');

}

if (i % order – 1 == 0)

{

putchar(' ');

}

}

return 0;

}

В консоль будет выведено: 1 1111111 11111111 11111111 10000001.

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.

**Постановка задачи.**

Необходимо:

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.

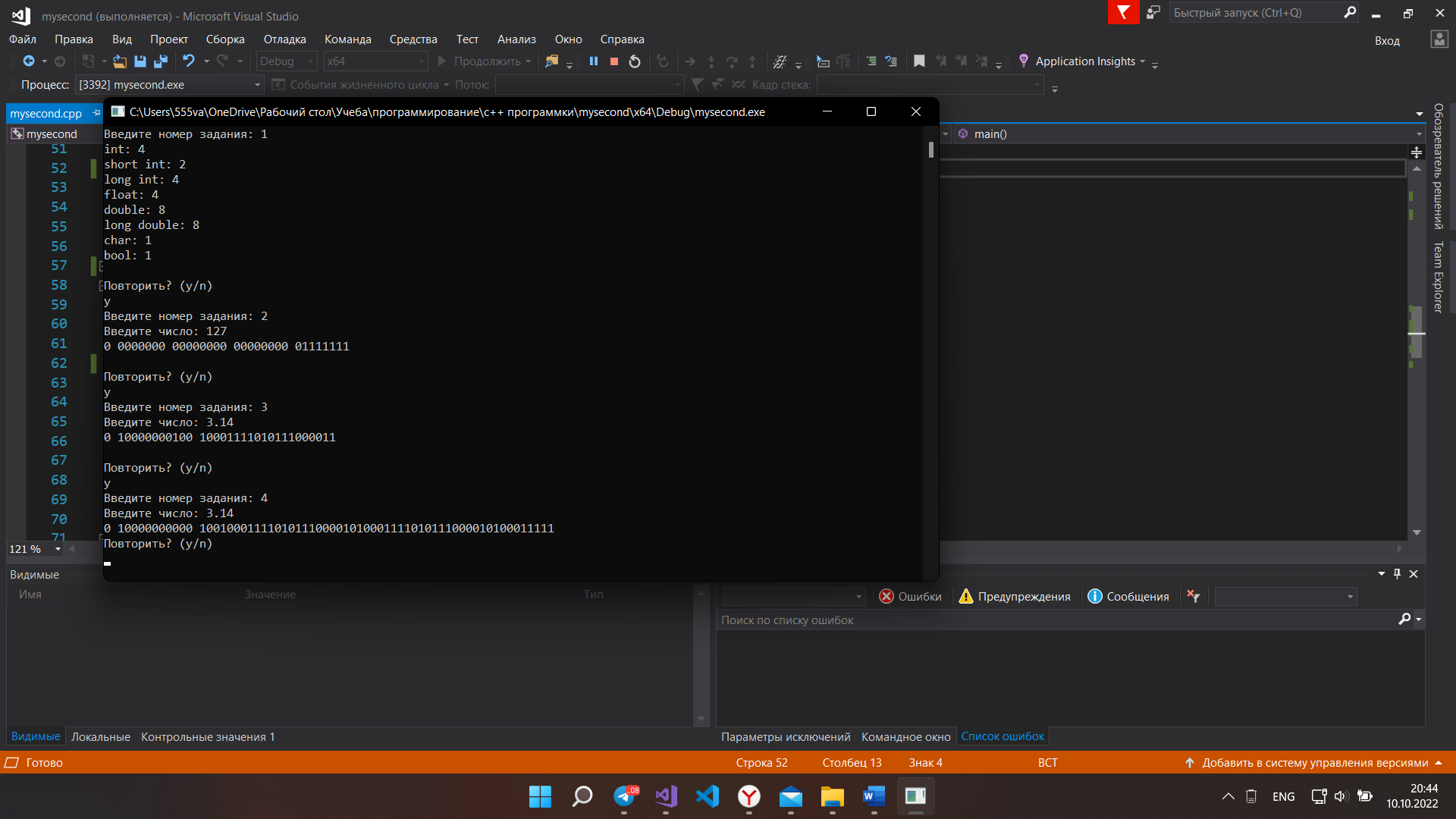
3) Вывести на экран двоичное представление в памяти (все разряды) типа float. При выводе необходимо визуально обозначить знаковый разряд мантиссы, знаковый разряд порядка (если есть), мантиссу и порядок.

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.**

Код программы представлен в приложении А.

|  |  |
| --- | --- |
| Ввод пользователем и обработка данных | Работа алгоритма и вывод на экран |
| При запуске программы перед пользователем появляется окно, где он должен ввести номер задания, которое будет выполнять программа. | Окно: |
| При вводе других чисел, вылезает уведомление об ошибке и предложение повторить работу программу. |  |
| При вводе корректных значений программа продолжает работу. |  |
| Чтобы завершить работу программы надо ввести с клавиатуры букву “n”. |  |

Блок скриншотов работы программы

Блок таблицы с тестовыми данными

Тестовые данные для задания 2.



Тестовые данные для задания 3.



Тестовые данные для задания 4.

**Выводы.**

Закрепились знания, полученные ранее в курсе, а именно: работа с циклами, работа со спецификаторами, теория о типах данных, операциях.

Приложение А

рабочий код

#include <iostream>

using namespace std;

int main() {

setlocale(0, "");

char answer = 'y';

do {

// объявление переменных для второго задания

int celoe;

unsigned int mask = 1 << (sizeof(celoe) \* 8 - 1);

// объявление переменных для третьего задания

static union {

int cel;

float vesh;

};

unsigned int mask3 = 1 << sizeof(vesh) \* 8 - 1;

// объявление переменных для четвертого задания

static union {

int arr[2];

double vesh2;

};

int zn4 = 1 << sizeof(int)\*8 - 1;

int msk4 = 1 << sizeof(int) \* 8 - 2;

int mask4, znak4;

int zamena;

int n;

cout << "Введите номер задания: ";

while (!(cin >> n) || cin.get() != '\n')

{

cin.clear();

cin.sync();

}

switch (n) {

case 1:

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

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

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

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

break;

case 2:

mask = 1 << 31;

cout << "Ввeдите число: ";

cin >> celoe;

for (int i = 0; i <= sizeof(celoe) \* 8 - 1; i++) {

if (i == 1 || i % 8 == 0 && i != 0) {

putchar(' ');

}

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

mask = mask >> 1;

}

cout << "\n";

break;

case 3:

cout << "Введите число: ";

cin >> vesh;

for (int i = 0; i <= sizeof(vesh) \* 8 - 1; i++) {

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

putchar(' ');

}

putchar(cel & mask3 ? '1' : '0');

mask3 >>= 1;

}

cout << "\n";

break;

case 4:

cout << "Введите число: "; cin >> vesh2;

zamena = arr[0];

arr[0] = arr[1];

arr[1] = zamena;

for (int i = 0; i < 2; i++) {

znak4 = zn4;

mask4 = msk4;

for (int j = 0; j <= (sizeof(int) \* 8 - 1); j++) {

if (i == 0 && (j == 1 || j == 12)) {

putchar(' ');

}

if ((i == 0 && j == 0) || (i == 1 && j == 0)) {

putchar(znak4&arr[i] ? '1' : '0');

}

else {

putchar(mask4&arr[i] ? '1' : '0');

mask4 = mask4 >> 1;

}

}

}

break;

default: cout << "Такого задания нет:(" << "\n";

}

cout << "\n";

cout << "Повторить? (y/n)" << "\n";

cin >> answer;

}

while (answer != 'n');

}