**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. 3373 |  | Шаткова В. А. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2023

**Цель работы.**

Изучение и организация структур; получение практических навыков работы со структурами; определение преимуществ и недостатков использования структур.

**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;

}

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:

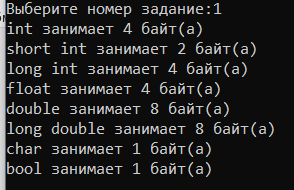
1) Вывести, сколько памяти (в байтах) на вашем компьютере отводится под различные типы данных со спецификаторами и без: int, short int, long int, float, double, long double, char и bool.

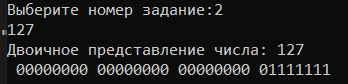
2) Вывести на экран двоичное представление в памяти (все разряды) целого числа. При выводе необходимо визуально обозначить знаковый разряд и значащие разряды отступами или цветом.

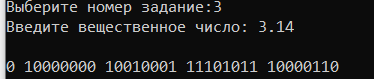
3) Вывести на экран двоичное представление в памяти (все разряды) типа float. При выводе необходимо визуально обозначить знаковый разряд мантиссы, знаковый разряд порядка (если есть), мантиссу и порядок.

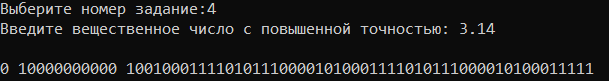
4) Вывести на экран двоичное представление в памяти (все разряды) типа double. При выводе необходимо визуально обозначить знаковый разряд мантиссы, знаковый разряд порядка (если есть), мантиссу и порядок. (\*)

**Выполнение работы.**









Приложение А

рабочий код

#include <iostream>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

cout << "Выберите номер задание:";

int choose, num = -1;

cin >> choose;

switch (choose) {

case 0: {

cout << "Работа программы завершена" << endl;

choose = -1;

}

break;

case 1: {

cout << "int занимает " << sizeof(int) << " байт(а)" << endl;

cout << "short int занимает " << sizeof(short int) << " байт(а)" << endl;

cout << "long int занимает " << sizeof(long int) << " байт(а)" << endl;

cout << "float занимает " << sizeof(float) << " байт(а)" << endl;

cout << "double занимает " << sizeof(double) << " байт(а)" << endl;

cout << "long double занимает " << sizeof(long double) << " байт(а)" << endl;

cout << "char занимает " << sizeof(char) << " байт(а)" << endl;

cout << "bool занимает " << sizeof(bool) << " байт(а)" << endl;

}

break;

case 2: {

cin >> num;

int mask = 1 << ((sizeof(int) \* 8) - 1); // Создаем маску для проверки каждого бита

cout << "Двоичное представление числа: " << num << endl;

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

if (i % 8 == 0) {

cout << " "; // Разделение каждых 4 разряда для удобочитаемости

}

cout << ((num & mask) ? '1' : '0'); // Используем битовую операцию И для получения значений битов

num <<= 1;// Сдвигаем число на 1 бит влево

}

cout << endl;

}

break;

case 3: {

cout << "Введите вещественное число: ";

float float\_num;

unsigned int float\_mask;

cin >> float\_num;

cout << "\n";

float\_mask = 1 << ((sizeof(float) \* 8) - 1);

union {

int intMemory;

float floatMemory;

};

floatMemory = float\_num;

for (int i = 0; i <= sizeof(float) \* 8; i++) {

cout << ((float\_mask & intMemory) ? 1 : 0);

float\_mask >>= 1;

if (i % 8 == 0) {

cout << " ";

}

}

}

break;

case 4: {

double double\_num;

int double\_order;

unsigned int double\_mask;

cout << "Введите вещественное число с повышенной точностью: ";

cin >> double\_num;

cout << "\n";

double\_order = sizeof(double\_num) \* 8 - 1;

double\_mask = 1 << double\_order;

union {

int intMemory[2];

double doubleMemory;

};

doubleMemory = double\_num;

for (int i = 0; i <= (double\_order / 2); i++) {

cout << ((double\_mask & intMemory[1]) ? 1 : 0);

double\_mask >>= 1;

if (!i or i == 11)

cout << " ";

}

double\_mask = 1 << double\_order;

for (int i = 0; i <= (double\_order / 2); i++) {

cout << ((double\_mask & intMemory[0]) ? 1 : 0);

double\_mask >>= 1;

}

}

break;

return 0;

}