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. 0324 Серебрякова А.К.

Преподаватель Глущенко А.Г.

Санкт-Петербург

2020

Цель работы.

Знакомство с внутренним представление различных типов данных, используемых компьютером при их обработке. Научиться работать с побитовыми операциями.

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; }

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.

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.

1.Чтобы узнать, сколько отводиться под тот или иной тип данных или объект памяти, нужно использовать операцию sizeof. Операция sizeof вычисляет размер в байтах.

2.Вывести то, каким образом тип данных представляется на компьютере, можно при помощи логических операций: побитового сдвига (<<) и поразрядной конъюнкции (&), сравнивая маску с двоичным представлением числа.

Размер типа int зависит от компьютера и компилятора. Для 16-разрядного процессора под величины этого типа отводится 2 байта. Для 32-разрядного – 4 байта. Знаковый бит отделяется пробелом.

3.Так как float нельзя использовать с побитовыми операциями, нужно воспользоваться объединением (union). Объединения позволяют интерпретировать несколькими различными способами содержимое одной и той же области памяти. Под int и float в памяти выделено одинаковое количество бит, поэтому float в памяти можно прочитать через Int.

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, все это так же выделяется пробелами.

4. double занимает в 2 раза больше битов в памяти, чем int. Под вещественное число с двойной точностью отводиться 8 байт. Для того, чтобы посмотреть значение double нужно создать int массив из двух элементов и воспользоваться теми же приемами, как и в задании 3.

Код программы.

#include <iostream>

using namespace std;

void taskONE();

void taskTWO(int num, unsigned int bit);

void taskTHREE(int num, unsigned int bit);

void taskFOUR(double ch, unsigned int bit, int mas[]);

int main()

{

    union

    {

        int num;

        float num2;

    };

    union

    {

        double ch;

        int mas[2];

    };

    int task;

    int bit;

    do

    {

        cout << " " << endl;

        cout << "Enter number of task\n " << endl;

        cout << "(1)Display how much memory (in bytes) your computer is allocated for different data types " << endl;

        cout << "(2)Display a binary representation in memory of the integer type " << endl;

        cout << "(3)Display a binary representation in memory of the float type" << endl;

        cout << "(4)Display a binary representation in memory of the double type" << endl;

        cout << "(0)exit" << endl;

        cout << "\n";

        cin >> task;

        cout << " ";

        system("cls");

        switch (task)

        {

        case 1:

            taskONE();

            break;

        case 2:

            cout << "Enter integer number:" << " " ;

            cout << " ";

            cin >> num;

            cout << "\n";

            cout << "Enter the number of bit:" << " ";

            cin >> bit;

            cout << "\n";

            cout << "rezult: ";

            taskTWO(num, bit);

            break;

        case 3:

            cout << "Enter float number:" << " ";

            cout << " ";

            cin >> num2;

            cout << "\n";

            cout << "Enter the number of bit:" << " ";

            cin >> bit;

            cout << "\n";

            cout << "rezult: ";

            taskTHREE(num, bit);

            break;

        case 4:

            cout << "Enter double number:" << " ";

            cout << " ";

            cin >> ch;

            cout << "\n";

            cout << "Enter the number of bit:" << " ";

            cin >> bit;

            cout << "\n";

            cout << "rezult: ";

            taskFOUR(ch, bit, mas);

        case 0:

            break;

        }

    } while (task != 0);

    return 0;

}

void taskONE()

{

    cout << "Int:" << " " << sizeof(int) << " " << "bit" << endl;

    cout << "Short int:" << " " << sizeof(short int) << " " << "bit" << endl;

    cout << "Long int:" << " " << sizeof(long int) << " " << "bit" << endl;

    cout << "Short float:" << " " << sizeof(float) << " " << "bit" << endl;

    cout << "Double:" << " " << sizeof(double) << " " << "bit" << endl;

    cout << "Long double:" << " " << sizeof(long double) << " " << "bit" << endl;

    cout << "Char:" << " " << sizeof(char) << " " << "bit" << endl;

    cout << "Bool:" << " " << sizeof(bool) << " " << "bit" << endl;

    cout << "\n";

};

void taskTWO(int num, unsigned int bit)

{

    unsigned int mask = 1 << bit - 1;

    for (int i = 1; i <= bit; i++)

    {

        cout << (num & mask ? '1' : '0');

        num <<= 1;

        if (i % bit - 1 == 0)

        {

            cout << (' ');

        }

    }

    cout << "\n";

}

void taskTHREE(int num, unsigned int bit)

{

    int k = 1, s = 1;

    unsigned int mask = 1 << bit - 1;

    for (int i = 1; i <= bit; i++)

    {

        cout << (num & mask ? '1' : '0');

        num <<= 1;

        if (k == 1)

        {

            if (i % 9 == 0)

            {

                cout << (' ');

                ++k;

            }

        }

        if (s == 1)

        {

            if (i % 1 == 0)

            {

                cout << (' ');

                ++s;

            }

        }

    }

    cout << "\n";

}

void taskFOUR(double ch, unsigned int bit, int mas[])

{

   for (int i = 1; i >= 0; i--)

    {

        for (int j = 31; j >= 0; j--)

        {

            cout << ((mas[i] >> j) & 1);

            if ((i == 1) && (j == 31))

            {

                cout << " ";

            }

            if ((i == 1) && (j == 20))

            {

                cout << " ";

            }

        }

    }

    cout << "\n";

}

Вывод.

Было изучено представление значений различных типов данных в двоичном коде, а также способ применения побитового сдвига и сравнение двоичных значений двух переменных при помощи поразрядной конъюнкции.

Пример работы.

