**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(ка) гр. |  | Каплунов А. А. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2022

**Цель работы.**

Изучение числовых типов данных и их внутреннего представления в памяти компьютера. Написание программы, показывающей объём памяти, занимаемый разными типами данных и визуализирующей представление чисел в памяти.

**Основные теоретические положения.**

Представление положительных и отрицательных числе 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.

Обратный код получается инвертированием всех цифр двоичного кода абсолютной величины, включая разряд знака. Прямой код можно преобразовать в обратный, инвертировав все значения битов.

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

Увидеть, как тип данных представляется на компьютере, можно при помощи побитового сдвига и поразрядной конъюнкции.

При сдвиге вправо для положительных чисел освобожденные позиции битов, заполняются нулями. Для отрицательных – единицами. Сдвиг влево является логическим сдвигом (биты, сдвигаемые с конца, отбрасываются, включая бит знака).

Внутреннее представление вещественного числа состоит из двух частей – мантиссы и порядка (экспоненты). 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.

**Экспериментальные результаты.**

Код программы представлен в приложении А.

1. При запуске программы пользователю выводится подсказка команды, которая выводит меню доступных команд и ожидается ввод команды с клавиатуры (рис. 1).

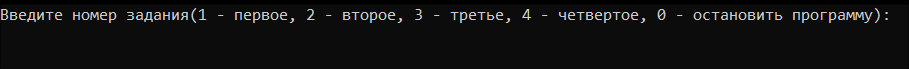


Рисунок 1. Запуск программы

1. Следующий шаг зависит от введенной команды, если пользователь ввёл:
   1. “0”, то выполнение программы завершается.
   2. “1”, то выводятся объёмы памяти, отводимые под различные типы данных со спецификатором (рис. 2)

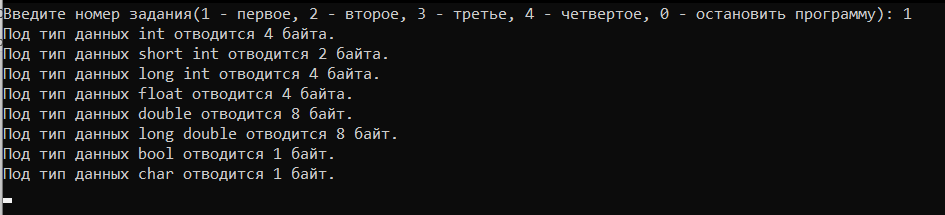


Рисунок 2. Задача 1

* 1. “2”, то ожидается ввод целого числа (int), двоичное представление которого потом будет выведено на экран (рис. 3).

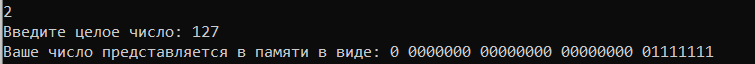


Рисунок 3. Задача 2, представление int в памяти

После вывода числа в двоичном представлении спрашивается: “С какого бита зеркально отобразить каждый следующий бит числа?”. При вводе номера бита выводится число с зеркальным побитовым представлением от заданного бита вправо (рис. 3.1).

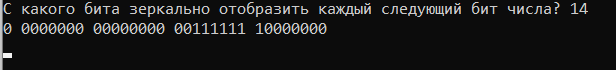


Рисунок 3.1. Задание на защиту

* 1. “3”, то ожидается ввод вещественного числа типа float, двоичное представление которого потом будет выведено на экран (рис. 4).



Рисунок 4. Задача 3, представление float в памяти

После вывода числа в двоичном представлении спрашивается: “С какого бита зеркально отобразить каждый следующий бит числа?”. При вводе номера бита выводится число с зеркальным побитовым представлением от заданного бита вправо (рис. 4.1).



Рисунок 4.1. Задание на защиту

* 1. “4”, то ожидается ввод вещественного числа типа double, двоичное представление которого потом будет выведено на экран (рис. 5).



Рисунок 5. Задача 4, представление double в памяти

После вывода числа в двоичном представлении спрашивается: “С какого бита зеркально отобразить каждый следующий бит числа?”. При вводе номера бита выводится число с зеркальным побитовым представлением от заданного бита вправо (рис. 5.1).



Рисунок 5.1. Задание на защиту

**Выводы.**

В ходе работы было изучено представление числовых типов данных в памяти компьютера, побитовые операции сдвига и поразрядной конъюнкции, методы перевода целых и вещественных чисел в двоичный код.

Приложение А

рабочий код

#include <iostream>

#include <cmath>

using namespace std;

int main() {

setlocale(0, "Rus");

int task;

cout << "Введите номер задания(1 - первое, 2 - второе, 3 - третье, 4 - четвертое, 0 - остановить программу): ";

while (true) {

cin >> task;

if (task == 0)

break;

switch (task) {

case 1:

cout << "Под тип данных int отводится " << sizeof(int) << " байта." << "\n";

cout << "Под тип данных short int отводится " << sizeof(short int) << " байта." << "\n";

cout << "Под тип данных long int отводится " << sizeof(long int) << " байта." << "\n";

cout << "Под тип данных float отводится " << sizeof(float) << " байта." << "\n";

cout << "Под тип данных double отводится " << sizeof(double) << " байт." << "\n";

cout << "Под тип данных long double отводится " << sizeof(long double) << " байт." << "\n";

cout << "Под тип данных bool отводится " << sizeof(bool) << " байт." << "\n";

cout << "Под тип данных char отводится " << sizeof(char) << " байт." << "\n";

break;

case 2:

{

int value;

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

cout << "Введите целое число: ";

cin >> value;

cout << "Ваше число представляется в памяти в виде: ";

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

{

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

mask >>= 1;

if (i % 8 == 0)

{

putchar(' ');

}

if (i % order - 1 == 0)

{

putchar(' ');

}

}

cout << endl;

cout << "С какого бита зеркально отобразить каждый следующий бит числа? ";

mask = 1 << (order - 1);

int bitNumber;

cin >> bitNumber;

for (int i = 1; i <= (order - bitNumber); i++) {

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

mask >>= 1;

if (i == 1 || i % 8 == 0) putchar(' ');

}

for (int i = order - bitNumber + 1; i <= order; i++) {

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

mask >>= 1;

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

}

cout << endl;

break;

}

case 3:

{

union {

float numb\_f;

int value;

};

unsigned order = sizeof(float) \* 8, mask = 1 << order - 1;

cout << "Введите вещественное число: ";

cin >> numb\_f;

cout << "Ваше число представляется в памяти в виде: ";

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

{

if (i == 2 || i == 10)

{

putchar(' ');

}

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

mask >>= 1;

}

cout << endl;

cout << "С какого бита зеркально отобразить каждый следующий бит числа? ";

mask = 1 << (order - 1);

int bitNumber;

cin >> bitNumber;

for (int i = 1; i <= (order - bitNumber); i++) {

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

mask >>= 1;

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

}

for (int i = order - bitNumber + 1; i <= order; i++) {

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

mask >>= 1;

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

}

cout << endl;

break;

}

case 4:

{

union {

double numb\_d;

int values[2];

};

cout << "Введите вещественное число с плавающей точкой: ";

cin >> numb\_d;

cout << "Ваше число представляется в памяти в виде: ";

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

unsigned order = sizeof(double) \* 8, mask = 1 << (order - 1);

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

{

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

mask >>= 1;

if ((i == 1 || i == 12) && j == 1) { // ((i % 8 == 0 || i == 1) && j == 1 || i % 8 == 0)

putchar(' ');

}

}

}

cout << endl;

cout << "С какого бита зеркально отобразить каждый следующий бит числа? ";

unsigned order = sizeof(double) \* 8, mask = 1 << (order - 1);

int bitNumber;

cin >> bitNumber;

if (bitNumber <= 32) {

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

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

mask >>= 1;

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

putchar(' ');

}

}

mask = 1 << (order - 1);

for (int i = 1; i <= (order / 2 - bitNumber); i++)

{

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

values[0] <<= 1;

}

for (int i = order / 2 - bitNumber + 1; i <= order / 2; i++)

{

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

values[0] <<= 1;

}

cout << endl;

break;

}

if (bitNumber <= 63 || bitNumber > 32) {

for (int i = 1; i <= order - bitNumber; i++) {

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

mask >>= 1;

if (i == 1 || i == 11) {

putchar(' ');

}

}

for (int i = order - bitNumber; i <= order / 2; i++) {

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

mask >>= 1;

if (i == 1 || i == 11) {

putchar(' ');

}

}

mask = 1 << (order - 1);

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

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

mask >>= 1;

}

}

cout << endl;

break;

}

}

}

}