**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.

|  |  |  |
| --- | --- | --- |
| Студент гр. 2372 |  | Братко В.В. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2022

**Цель работы.**

Изучение и организация типов данных; получение практических навыков работы с типами данных; определение преимуществ и недостатков использования различных типов данных.

**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.

#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.

**Постановка задачи.**

1. Необходимо 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.
   1. Добавить возможность изменить произвольные биты на 1 или 0 по указанию пользователя.
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.**

|  |  |
| --- | --- |
| Ввод пользователем и обработка данных | Работа алгоритма и вывод на экран |
| 1.Вывод размера типов данных | |
| При запуске программы перед пользователем появляется окно с размером типов данных в байтах |  |

Продолжение Таблицы

|  |  |
| --- | --- |
| 2.Вывод целого числа в двоичной системе записи | |
| Пользователь вводит десятичное число, а программа выводит результат в двоичном виде. | Обработка исключений: |

|  |  |
| --- | --- |
| 2.1 изменение произвольных битов на 1 или 0 | |
| Пользователь вводит бит и номер бита и программа меняет указанный бит |  |

|  |  |
| --- | --- |
| 3 Вывод дробного числа(float) в двоичной системе записи | |
| Пользователь вводит десятичное число, а программа выводит результат в двоичном виде. |  |

Продолжение Таблицы

|  |  |
| --- | --- |
| 4 Вывод дробного числа(double) в двоичной системе записи | |
| Пользователь вводит десятичное число, а программа выводит результат в двоичном виде. |  |

|  |  |
| --- | --- |
| Цикличность программы | |
| Eсли пользователь желает, то он может запустить выполнение программы заново, нажав цифру 1 |  |

Продолжение Таблицы

|  |  |
| --- | --- |
| Полный вывод программы | |
| Здесь представлен весь вывод консоли. | Вывод в консоли: |

**Полный код программы:**

#include <iostream>  
#include <algorithm>  
  
using namespace std;  
// Добавить возможность изменить произвольные биты на 1 или 0 по указанию пользователя.  
void change(int input){  
 const int count = 8 \* sizeof(int) - 1;  
 cout << "Enter the bit and bit number you want to replace (separated by a space): ";  
 int index,value;  
 int mask;  
 cin >> value >> index;  
 if (value==1){  
 mask = 1 << (index-1);  
 input |= mask;  
 }else if (value==0){  
 mask = 1 << (index-1);  
 mask = ~mask;  
 input &= mask;  
 }else {  
 cout << "ERROR!!!";  
 }  
 for (int i = 0; i <= count; i++) {  
 if (i == 1 || (i % 8 == 0 and i != 0)) {  
 putchar(' ');  
 }  
 mask = 1 << (count - i);  
 putchar(input & mask ? '1' : '0');  
 }  
}  
  
int main(){  
 const int count = 8 \* sizeof(int) - 1;  
 int contin = 1;  
 while (contin == 1) {  
 //1. Вывести, сколько памяти (в байтах) на вашем компьютере отводится под различные типы данных со спецификаторами и без.  
 cout << "|int:" << sizeof(int) << "\n|int short: " << sizeof(int short) << "\n|long int: " << sizeof(long int);  
 cout << "\n|float: " << sizeof(float) << "\n|double: " << sizeof(double) << "\n|long double: "  
 << sizeof(long double);  
 cout << "\n|char: " << sizeof(char) << "\n|bool: " << sizeof(bool);  
 //2. Вывести на экран двоичное представление в памяти (все разряды) целого числа.  
 int inputInt;  
 cout << "\n" << "Enter an integer number(integer): ";  
 try {  
 cin >> inputInt;  
 if (cin.fail()) {  
 throw 1;  
 }  
 } catch (int exeption) {  
 cout << "ERROR!!!";  
 exit(0);  
 }  
 cin.sync();  
 int mask;  
 for (int i = 0; i <= count; i++) {  
 if (i == 1 || (i % 8 == 0 and i != 0)) {  
 putchar(' ');  
 }  
 mask = 1 << (count - i);  
 putchar(inputInt & mask ? '1' : '0');  
 }  
 cout << "\n";  
 change(inputInt);  
  
 // 3 Вывести на экран двоичное представление в памяти (все разряды) типа float.  
  
 union {  
 float inputFloat;  
 int equalFloat;  
 };  
 cout << "\n" << "Enter a fractional number(float): ";  
 try {  
 cin >> inputFloat;  
 if (cin.fail()) {  
 throw 1;  
 }  
 } catch (int exeption) {  
 cout << "ERROR!!!";  
 exit(0);  
 }  
 cin.sync();  
 for (int i = 0; i <= count; i++) {  
 if (i == 1 || i == 9) {  
 putchar(' ');  
 }  
 mask = 1 << (count - i);  
 putchar(equalFloat & mask ? '1' : '0');  
 }  
 //4 Вывести на экран двоичное представление в памяти (все разряды) типа double.  
  
 union {  
 int arr[2];  
 double inputDouble;  
 };  
 cout << "\n" << "Enter a fractional number(double): ";  
 try {  
 cin >> inputDouble;  
 if (cin.fail()) {  
 throw 1;  
 }  
 } catch (int exeption) {  
 cout << "ERROR!!!";  
 exit(0);  
 }  
 cin.sync();  
 for (int i = 1; i >= 0; i--) {  
 for (int j = 0; j <= count; j++) {  
 if ((j == 1 || j == 12) && i == 1) {  
 putchar(' ');  
 }  
 mask = 1 << (count - j);  
 putchar(arr[i] & mask ? '1' : '0');  
 }  
 }  
 cout << "\n" << "If you want to run the program again press 1: ";  
 cin >> contin;  
 }  
 return 0;  
}

**Выводы:**

В процессе написания программы для практической работы было изучено:

1. Как компьютер представляет в памяти различные типы данных
2. Сколько занимает места в памяти тот или иной тип данных
3. Что такое блок union