МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра «Информационные системы»

ОТЧЕТ

по практической работе №1

по дисциплине «Программирование»

Тема: типы данных и их внутреннее представление в памяти

Студент гр. 0323 _____ Балашевич К.Д.

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.

Информация в памяти ЭВМ (компьютера) хранится в виде непрерывной последовательности ячеек памяти, называемыми битами. Каждый бит может находится в одном из двух (в большинстве ЭВМ) состояний, условно называемых нулём и единицей по аналогии с нулём и единицей в мат. логике.

Биты в данной последовательности объединены в более крупные ячейки, называемыми байтами. Каждый байт имеет свой уникальный адрес, используя который программное обеспечение ЭВМ может получить доступ к информации, в нем содержащейся. Длина байта зависит от архитектуры компьютера, операционной системы, используемого языка и компилятора. В большинстве современных ЭВМ байт имеет размерность 8 бит.

Для работы с данными в большинстве языков программирования используется понятие типов данных. Тип данных определяет:

- характер данных
- способ кодирования данных в памяти
- возможные значения данных
- правила обработки данных

В языке C++ определено шесть основных типов данных для представления целых, вещественных, символьных и логических величин. При этом, типы данных, используемые для хранения целых, символьных и логических величин называются целочисленными и используют одинаковые принципы кодирования данных в памяти.

К целочисленным типам данных относятся типы данных:

- int
- char
- bool

и др.

К вещественным типам данных (числам с плавающей запятой) относятся типы данных:

- float
- double

и др.

Объём памяти, выделяемой под тип данных, а также диапазон значений могут быть скорректированы при помощи спецификаторов short, long, long long, signed, unsigned. Спецификаторы добавляются слева к названию типа и могут быть указаны в любом порядке. При этом спецификаторы short, long, long long могут быть указаны совместно со спецификаторами signed и unsigned, но не друг с другом. Аналогично не могут быть указаны совместно друг с другом и спецификаторы signed и unsigned.

Целочисленные типы (в языке с++) представляются в памяти следующим образом:

Первый бит – знаковый (для знаковых типов данных). В нём указывается положительное ли число (при указании в нем 0) или отрицательное (при указании 1).

Число записывается в память в двоичной системе исчисления в дополнительном коде. Дополнительный код положительного числа равен числу в прямом коде. Дополнительный код отрицательного числа получается путём прибавления единицы к младшему разряду отрицательного числа, записанного в обратном коде. Обратный код отрицательного числа, в свою очередь, получается путём инвертирования всех его разрядов.

Размер типа bool в стандарте C++ не определён, размер типа char всегда 1 байт. Размер типа int, по стандарту, не может быть менее 16 бит, типа long int

(аналогичен типу int со спецификатором long) – не менее 32 бит, типа long long int – не менее 64 бит.

Способ представления вещественных типов данных в стандарте языка c++ не определён. В большинстве имплементаций используется стандарт IEEE-754.

В соответствии со стандартом IEEE-754 число записывается в память в экспоненциальной форме. Под тип float отводится память в размере 32 бита (при этом размер байта значения не имеет). Первый бит – знаковый (по аналогии с типом integer). Далее следует порядок экспоненты, к которому предварительно прибавлено число 127. Отрицательный порядок указать в соответствии со стандартом IEEE-754 невозможно. Далее следует мантисса в нормализованной форме, при этом первый её разряд (всегда равен единице) не указывается.

Под порядок для типа float отводится 8 бит, под мантиссу – 23 бита. Аналогично представляется и тип double, за тем исключением, что под порядок отводится 11 бит, а под мантиссу – 52 бита. Способ представления типа данных long double зависит от имплементации.

Для определения объёма памяти, отведённой под тот или иной тип данных, в языке C++ присутствует операция sizeof. При этом значение, ею возвращаемое, представлено в байтах. Для получения же размера байта возможно использования константы CHAR_BIT, определённой в заголовочном файле climits.

Для работы с побитовый представлением данных в языке C++ присутствуют операции:

- ~a - побитового отрицания
- a&b – поразрядной конъюнкции
- a|b – поразрядной дизъюнкции
- a^b – поразрядной исключающей дизъюнкции
- a<<b – побитового сдвига влево
- a>>b - побитового сдвига вправо

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 (int, short int, unsigned int). При выводе необходимо визуально обозначить знаковый разряд и значащие разряды отступами ли 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.

Сделать вывод по проделанной работе.

Выполнение работы.

Для решения поставленной задачи была написана программа на языке C++. Итоговый код программы представлен в приложении А.

Было проведено тестирование программы с использование различных ЭВМ и компиляторов. Результаты тестирования представлены в приложении Б. При этом были получены результаты, соответствующие расчётным.

Выводы.

Поскольку стандартом C++ не установлен объём памяти отводимый под различные типы данных, при написании программ, предназначенных для выполнения на различных ЭВМ предпочтительно использование для представления целых чисел либо типа char, либо типов с фиксированной длиной, определённых в заголовочном файле cstdint, введённые стандартом C++11, во всех случаях, когда важен размер памяти, отводимый под данный тип.

ПРИЛОЖЕНИЕ А
ПОЛНЫЙ КОД ПРОГРАММЫ

```
#include <iostream>
#include <climits>

using namespace std;

int menuMain (int request);
int menuAskQuestion ();
void printSizeof ();
void printBinInt ();
void printBinFloat ();
void printBinDouble ();

int main ()
  {
    setlocale(LC_ALL, "ru_RU.utf8");
    int i;
    i=menuMain (5);
    while (i)
      {
        i=menuMain (i);
      }
    return 0;
  }

void printSizeof ()
  {
      cout << endl << "На данном компьютере под следующие типы данных
      (со спецификаторами и без) отводится память в размере (в
      байтах):" << endl << endl << "int\t\t" << sizeof (int) << endl
      << "short int\t" << sizeof (short int) << endl << "long int\t"
      << sizeof (long int) << endl << "float\t\t" << sizeof (float)
      << endl << "double\t\t" << sizeof (double) << endl << "long
      double\t" << sizeof (long double) << endl << "char\t\t" <<
      sizeof (char) << endl << "bool\t\t" << sizeof (bool) << endl <<
      endl;
  }

void printBinInt ()
  {
    int Int;
```

```
    int mask=(1<<((sizeof(int)*CHAR_BIT)-1));
    cout << endl << "Введите целое число: ";
    cin >> Int;
    cout << endl << "Двоичное представление в памяти введённого
    числа (знаковый разряд отделён пробелом):" << endl;
    cout << (Int & mask ? "1 " : "0 ");
    Int<<=1;
    for (int unsigned i=0; i<((sizeof(int)*CHAR_BIT)-1); i++)
      {
        cout << (Int & mask ? "1" : "0");
        Int<<=1;
      }
    cout << endl;
  }

void printBinFloat ()
  {
    union
     {
       float Float;
       char Char[3];
     };
    char mask=1<<7;
    int bitCount=32;
    cout << endl << "Введите вещественное число: ";
    cin >> Float;
    cout << endl << "Двоичное представление в памяти введённого числа
    (знаковый разряд, порядок и мантисса разделены пробелами):" <<
    endl;
    for (int i=3; i>=0; i--)
      {
        for (int j=0; j < CHAR_BIT; j++)
          {
            cout << (Char[i] & mask ? "1" : "0");
            Char[i]<<=1;
            if (bitCount==24||bitCount==32) cout << " ";
            bitCount--;
            if (!bitCount) break;
          }
        if (!bitCount) break;
      }
    cout << endl;
  }
```

```
void printBinDouble ()
  {
    union
      {
        double Double;
               char Char[7];
      };
    char mask=1<<7;
    int bitCount=64;
    cout << endl << "Введите вещественное число: ";
    cin >> Double;
    cout << endl << "Двоичное представление в памяти введённого числа
    (знаковый разряд, порядок и мантисса разделены пробелами):" <<
    endl;
    for (int i=7; i>=0; i--)
      {
        for (int j=0; j < CHAR_BIT; j++)
          {
            cout << (Char[i] & mask ? "1" : "0");
            Char[i]<<=1;
            if (bitCount==53||bitCount==64) cout << " ";
            bitCount--;
            if (!bitCount) break;
          }
        if (!bitCount) break;
      }
    cout << endl;
  }

int menuMain (int request)
  {
    int k;
    switch (request)
      {
        case 0:
          return 0;
          break;
        case 1:
          printSizeof ();
          return 5;
          break;
        case 2:
          printBinInt();
          k=menuAskQuestion();
```

```
            if (k==2||!k) return 5;
              else return 2;
            break;
          case 3:
            printBinFloat();
            k=menuAskQuestion();
            if (k==2||!k) return 5;
              else return 3;
            break;
          case 4:
            printBinDouble();
            k=menuAskQuestion();
            if (k==2||!k) return 5;
              else return 4;
            break;
          case 5:
            cout << "Главное меню:" << endl << "0) Вызод из программы"
            << endl << "1) Вывод объёма памяти, отводимого под
            различные типы данных" << endl << "2) Вывод на экран
            двоичного представления в памяти целого числа" << endl <<
            "3) Вывод на экран двоичного представления в памяти типа
            float" << endl << "4) Вывод на экран двоичного представле
            ния в памяти типа double" << endl << endl << "Введите
            желаемый пункт: ";
            int k;
            cin >> k;
            return k;
            break;
          default:
            return 5;
            break;
        }
    }

int menuAskQuestion ()
    {
      int k;
      cout << endl << "Меню:" << endl << "1) Продолжить" << endl <<
      "2) Вернуться в главное меню" << endl << endl << "Введите желаемый
      пункт: ";
      cin >> k;
      return k;
    }
```

ПРИЛОЖЕНИЕ Б
РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Тестирование программы осуществлялось на компьютере с процессором x64 под управлением ОС Windows 10 Pro (версия 2004, 64-бит) с использованием следующих компиляторов:

- g++ (GCC) 9.3.0
- Оптимизирующий компилятор Microsoft (R) C/C++ версии 19.16.27043 для x86

А также на компьютере с процессором x64 под управлением ОС Windows 10 Pro (версия 1709, 32-бит) с использованием следующих компиляторов:

- g++ (GCC) 10.2.0
- Оптимизирующий 32-разряднрый компилятор Microsoft (R) C/C++ версии 16.00.30319.01 для 80x86

Результаты тестирования первой части программы представлены в табл. 1.

Таблица 1 – Результаты тестирования первой части программы

Используемый компилятор	Ожидаемый результат	Полученный результат
GCC (64-бит ОС)	int 4 short int 2 long int 8 float 4 double 8 long double 16 char 1 bool 1	int 4 short int 2 long int 8 float 4 double 8 long double 16 char 1 bool 1
Microsoft (64-бит ОС)	int 4 short int 2 long int 4 float 4 double 8 long double 8 char 1 bool 1	int 4 short int 2 long int 4 float 4 double 8 long double 8 char 1 bool 1

GCC (32-бит ОС)	int 4 short int 2 long int 4 float 4 double 8 long double 12 char 1 bool 1	int 4 short int 2 long int 4 float 4 double 8 long double 12 char 1 bool 1
Microsoft (32-бит ОС)	int 4 short int 2 long int 4 float 4 double 8 long double 8 char 1 bool 1	int 4 short int 2 long int 4 float 4 double 8 long double 8 char 1 bool 1

При тестировании последующих частей программы были получены идентичные результаты вне зависимости от использованного компилятора.

Результаты тестирования второй части программы представлены в табл. 2.

Таблица 1 – Результаты тестирования второй части программы

Введённое значение	Ожидаемый результат	Полученный результат
0	0 00000000000000000000000000000000	0 00000000000000000000000000000000
16	0 00000000000000000000000000010000	0 00000000000000000000000000010000
2147483647	0 11111111111111111111111111111111	0 11111111111111111111111111111111
-2147483648	1 00000000000000000000000000000000	1 00000000000000000000000000000000
-568958	1 11111111111011101010001100000010	1 11111111111011101010001100000010

Результаты тестирования третьей части программы представлены в табл. 3.

Таблица 3 – Результаты тестирования третьей части программы

Введённое значение	Ожидаемый результат	Полученный результат
0	0 00000000 00000000000000000000000	0 00000000 00000000000000000000000
3.1416	0 10000000 10010010000111111111001	0 10000000 10010010000111111111001
3.402823466e38	0 11111110 11111111111111111111111	0 11111110 11111111111111111111111
1.175494351e-38	0 00000001 00000000000000000000000	0 00000001 00000000000000000000000
-256	1 10000111 00000000000000000000000	1 10000111 00000000000000000000000

Результаты тестирования четвёртой части программы представлены в табл. 4.

Таблица 4 – Результаты тестирования четвёртой части программы (пробелы обозначены символом нижнего подчёркивания

Введённое значение	Ожидаемый результат	Полученный результат
0	0_00000000000_0000000000000000000000000000000000000000000000000000	0_00000000000_0000000000000000000000000000000000000000000000000000
3.1416	0_10000000000_1001001000011111111100101110010010001110100010100111	0_10000000000_1001001000011111111101011100100100011101000101001111
1.7976931348623158e308	0_11111111110_1111111111111111111111111111111111111111111111111111	0_11111111110_1111111111111111111111111111111111111111111111111111
2.2250738585072014e-308	0_00000000001_0000000000000000000000000000000000000000000000000000	0_00000000001_0000000000000000000000000000000000000000000000000000
-256	1_10000000111_0000000000000000000000000000000000000000000000000000	1_10000000111_0000000000000000000000000000000000000000000000000000