**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 |  | Егоров М.А. |
| 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.**

Любая программа предназначена для обработки каких-либо данных, например, чисел или текстов. Данные могут быть различного вида или типа и, в зависимости от их типа, с ними можно выполнять разные действия.

Тип данных характеризует:

* объем памяти, выделяемый под данные;
* их внутреннее представление в памяти компьютера;
* набор допустимых операций (действий);
* множество допустимых значений.

Все типы данных можно подразделить на простые — они предопределены стандартом языка, и сложные (или составные) — задаются пользователем. Данные простого типа нельзя разложить на более простые составляющие без потери сущности данного. Простые типы данных создают основу для построения более сложных типов: массивов, структур, классов. Простые типы в языке C++ — это целые, вещественные типы, символьный и логический тип и тип **void**.

Целый тип данных предназначен для представления в памяти компьютера обычных целых чисел. Основным и наиболее употребительным целым типом является тип **int**. Гораздо реже используют его разновидности: **short** (короткое целое) и **long** (длинное целое). Также к целым типам относится тип **char** (символьный). По умолчанию все целые типы являются *знаковыми*, т.е. старший бит в таких числах определяет знак числа: 0 — число положительное, 1 — число отрицательное. Кроме знаковых чисел на C++ можно использовать *беззнаковые*. В этом случае все разряды участвуют в формировании целого числа. При описании беззнаковыхцелыхпеременных добавляется слово **unsigned** (без знака).

В стандарте C++ нет типа данных, который можно было бы считать действительно символьным. Для представления символьной информации есть два типа данных, пригодных для этой цели, — это типы **char** и **wchar\_t**, хотя оба эти типа по сути своей вообще-то являются целыми типами.

Тип **char** используется для представления символов в соответствии с системой кодировки ASCII (American Standard Code for Information Interchange — Американский стандартный код обмена информации). Это семибитный код, его достаточно для кодировки 128 различных символов с кодами от 0 до 127. Символы с кодами от 128 до 255 используются для кодирования национальных шрифтов, символов псевдографики и др.

Логический (булевый) тип обозначается словом **bool**. Данные булевого типа могут принимать только два значения: **true** и **false**. Значение **false** обычно равно числу 0, значение **true** — числу 1. Под данные булевого типа отводится 1 байт.

Особенностью вещественных (действительных) чисел является то, что в памяти компьютера они практически всегда хранятся приближенно, а при выполнении арифметических операций над такими данными накапливается вычислительная погрешность.

Имеется три вещественных типа данных: **float**, **double** и **long double**. Основным считается тип **double**. Так, все математические функции по-умолчанию работают именно с типом **double**. В таблице ниже приведены основные характеристики вещественных типов:

Тип **void** — самый необычный тип данных языка C++. Множество значений этого типа пусто, т.е. нельзя переменной такого типа присвоить какое-нибудь значение. Более того, нельзя даже описать переменную этого типа. Зачем же нужно то, чем вроде бы невозможно воспользоваться?

**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.**

Поставленная задача была решена, был написан код на языке программирования C++. Код представлен в приложении А.

Также было проведено тестирование программы, результаты представлены в приложении Б и соответствуют расчетным.

**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 menu ();

void memorycapacity ();

void bitINT ();

void bitFLOAT ();

void bitDOUBLE ();

int main()

{

setlocale(LC\_ALL, "ru\_RU.utf8");

int i = 5;

while (i)

{

int k;

switch (i)

{

case 0:

return 0;

break;

case 1:

memorycapacity();

i = 5;

break;

case 2:

bitINT();

k = menu();

if (k == 2 || !k) i = 5;

else i = 2;

break;

case 3:

bitFLOAT();

k = menu();

if (k == 2 || !k) i = 5;

else i = 3;

break;

case 4:

bitDOUBLE();

k = menu();

if (k == 2 || !k) i = 5;

else i = 4;

break;

case 5:

cout << endl << "МЕНЮ:" << endl <<

"1. Вывод объёма памяти в байтах, 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 <<

"0. Выход" << endl << endl <<

"Выберите пункт: ";

int k;

cin >> k;

i = k;

break;

default:

i = 5;

break;

}

}

return 0;

}

void memorycapacity ()

{

cout << endl <<"Под различные типы данных на этом компьютере отводится памяти 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 bitINT ()

{

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 bitFLOAT ()

{

union

{

float Float;

char Char[3];

};

char mask=1<<7;

int bitCount=32;

cout << endl << "Введите вещественное число: ";

cin >> Float;

cout << endl << "Двоичное представление данного числа:" << 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 bitDOUBLE ()

{

union

{

double Double;

char Char[7];

};

char mask=1<<7;

int bitCount=64;

cout << endl << "Введите вещественное число: ";

cin >> Double;

cout << endl << "Двоичное представление данного числа:" << 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 menu ()

{

int k;

cout << endl << "Меню:" << endl <<

"1) Продолжить" << endl <<

"2) Вернуться в главное меню" << endl << endl <<

"Выберите пункт: ";

cin >> k;

return k;

}

Приложение Б

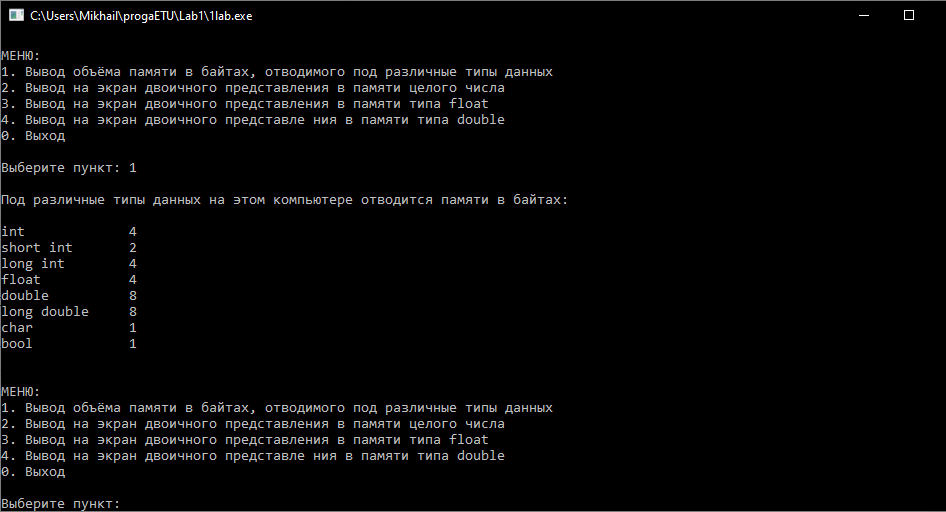
результаты тестирования

Тест 1. Вывод объема памяти в байтах, отводимого под различные типы данных.

Таблица 1 – Ожидаемый результат.

|  |  |
| --- | --- |
| **Тип данных** | **Объем отводимой памяти (байт)** |
| int | 4 |
| short int | 2 |
| long int | 4 |
| float | 4 |
| double | 8 |
| long double | 8 |
| char | 1 |
| bool | 1 |

Рисунок 1 – Результат



Тест 2. Вывод на экран двоичного представления в памяти целого числа

Таблица 2 – Ожидаемые результаты тестирования 2 части

|  |  |
| --- | --- |
| **Введённое значение** | **Ожидаемый результат** |
| 0 | 0 0000000000000000000000000000000 |
| 2147483647 | 0 1111111111111111111111111111111 |
| -2147483648 | 1 0000000000000000000000000000000 |

Рисунок 2 – Результат 1 ввода

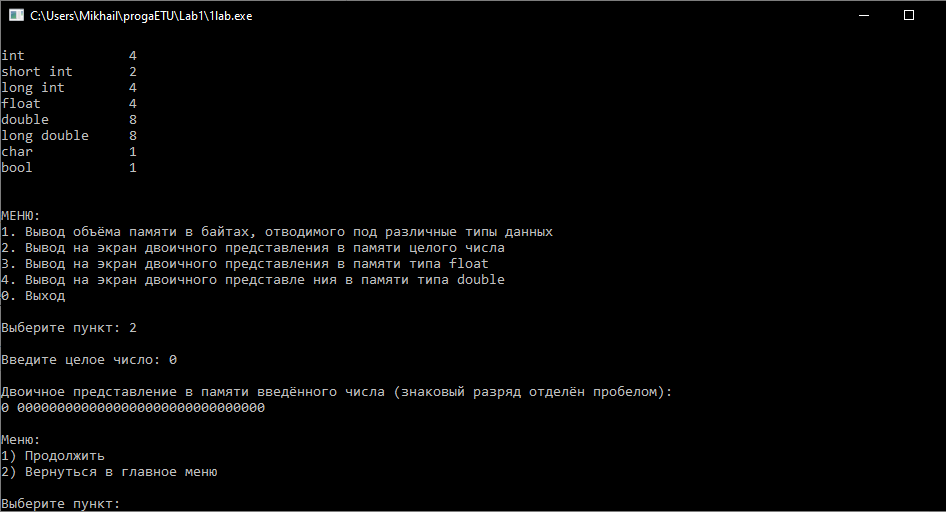


Рисунок 3 – Результат 2 ввода

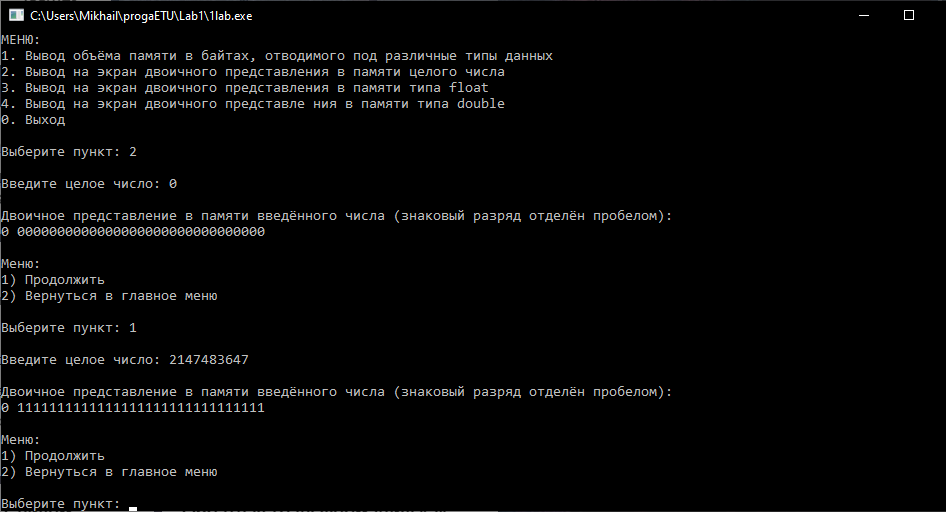


Рисунок 4 – Результат 3 ввода

