**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. |  | Саевич И. И. 4372 |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2021

**Цель работы.**

Изучение стандартных типов данных; получение практических навыков работы с ними и их внутренним представлением в памяти компьютера;

**Основные теоретические положения.**

C++ — язык со статической типизацией. У каждой переменной на этапе компиляции должен быть чётко определённый тип данных. Про каждый тип данных заранее известно, сколько места в памяти занимает переменная такого типа.

Основные типы данных часто называют арифметическими, поскольку их можно использовать в арифметических операциях. Для описания основных типов определены следующие ключевые слова:

· **int** (целый);

· **float** (вещественный);

· **double** (вещественный тип с двойной точностью);

· **bool** (логический);

· **char** (символьный).

Существует четыре спецификатора типа, уточняющих внутреннее представление и диапазон значений стандартных типов:

· **short** (короткий);

· **long** (длинный);

· **signed** (знаковый);

· **unsigned** (беззнаковый).

Размер типа int не определяется стандартом, а зависит от компьютера и компилятора. Для 16-разрядного процессора под величины этого типа отводится 2 байта, а для 32-разрядного - 4 байта.

Спецификатор short перед именем типа указывает компилятору, что под число требуется отвести 2 байта независимо от разрядности процессора.

Спецификатор long означает, что целая величина будет занимать 4 байта. Таким образом, на 16-разрядном компьютере эквиваленты int и short int, а на 32-разрядном — int и long int.

По умолчанию все целочисленные типы считаются знаковыми, то есть спецификатор signed можно опускать. В знаковых типах старший бит числа интерпретируется как знаковый (О — положительное число, 1 — отрицательное). Для представления отрицательных чисел в С++ используется дополнительный код. Чтобы получить его, надо инвертировать все биты кроме знакового, а затем к младшему биту прибавить единицу.

Спецификатор unsigned позволяет представлять только положительные числа, при этом старший разряд рассматривается как часть кода числа.

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. Спецификатор unsigned неприменим к вещественным типам.

**Постановка задачи.**

Необходимо разработать алгоритм и написать программу, которая позволяет:

1. Вывести, сколько памяти (в байтах) на 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.**

Код программы представлен в приложении A.

Программа состоит из главной функции main и четырех дополнительных функций, каждая из который отвечает соответственно за одно из четырех заданий. В главной функции мы считываем номер задания, которое хочет увидеть пользователь и вызываем соответствующую функцию.

Функция First() выводит размер стандартных типов в байтах при помощи функции sizeof().

Функция Second() считывает число типа int, которое вводит пользователь. Оно сохраняется в памяти компьютера в двоичном виде. Затем мы создаём переменную mask и присваиваем ей значение 1 (в двоичном виде выглядит как 31 ноль и единичка). При помощи операции побитового сдвига сдвигаем единичку на 31 бит влево (получается единичка и 31 ноль). Затем при помощи цикла for и операции поразрядной конъюнкции “сравниваем” каждый бит искомого числа с единичкой. В конце каждой итерации цикла сдвигаем маску на один бит вправо. В итоге получаем двоичное представление в памяти числа, которое ввел пользователь.

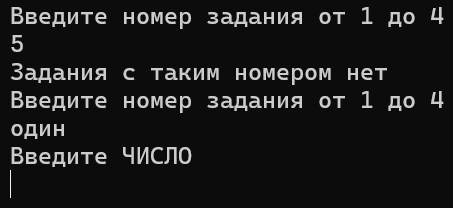
Алгоритм функции Third() почти такой же, с той лишь разницей, что мы используем структуру union. Эта структура позволяет биты с одного и того же участка памяти интерпретировать по-разному. Внутри структуры union мы создаем переменную F типа float и переменную tool типа int. Переменную F инициализируем значением, введенным пользователем, а переменная tool интерпретирует те же биты но как число типа int. Необходимость использования структуры union обосновывается невозможностью использования побитовых операций над вещественными типами данных. Таким образом, использовав алгоритм из функции Second() над переменной tool, мы получим двоичное представление вещественного числа float, введённого пользователем.

Алгоритм функции Fourth() почти такой же, но теперь вместо вспомогательной переменной типа int придется использовать массив из двух целых чисел, так как размер double – 8 байт, а размер int – всего 4 байта. Для реализации известного алгоритма используем два цикла for, первый из которых будет отвечать за перебор двух элементов массива, а второй – за сдвиг маски.

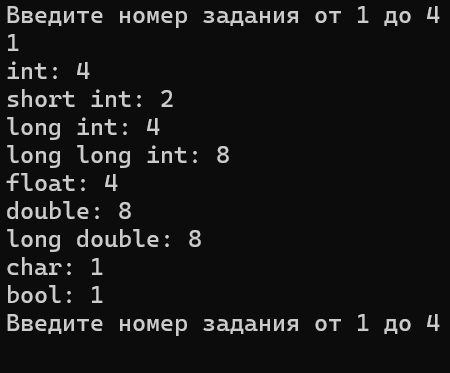
При запуске программы пользователь видит окно, где он может выбрать задание:



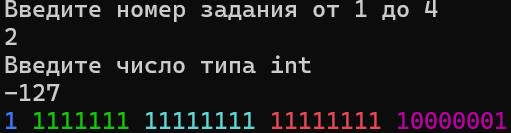
Проверка на корректность ввода:



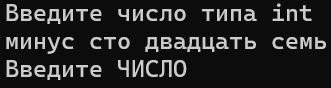
Задание номер 1:



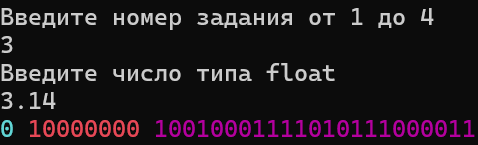
Теперь пользователь может ввести номер другого задания. Задание 2:



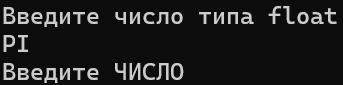
Пользователь вводит число типа int, представление в памяти которого хочет увидеть. Отступами и цветом обозначены байты, а также знаковый бит. Также осуществляется проверка на корректность ввода:



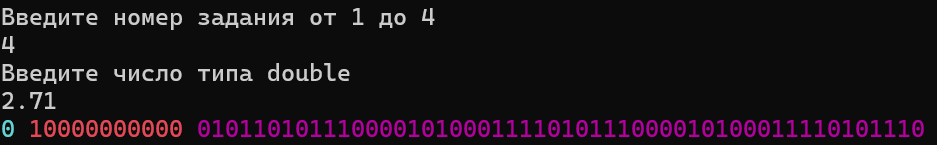
Задание 3:



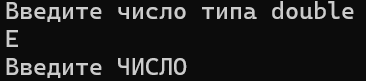
Отступами и цветом обозначены знаковый бит, порядок и мантисса. Также осуществляется проверка на корректность ввода:



Задание 4:



Отступами и цветом обозначены знаковый бит, порядок и мантисса. Также осуществляется проверка на корректность ввода:



**Выводы.**

В ходе работы мы смогли изучить стандартные типы данных и их внутреннее представление в памяти компьютера, получить практические навыке по работе с ними.

Приложение А

рабочий код

#include <iostream>

#include <windows.h>

using namespace std;

void First()

{

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

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

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

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

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

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

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

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

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

}

void Second()

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

int color = 9;

int A;

int order = sizeof(int) \* 8;

unsigned int mask = 1 << order - 1;

cout << "Введите число типа int" << endl;

cin >> A;

while (cin.fail())

{

cin.clear();

cin.ignore(1000, '\n');

cout << "Введите ЧИСЛО" << endl;

cin >> A;

}

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

{

SetConsoleTextAttribute(hConsole, color);

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

mask >>= 1;

if (i % 8 == 0 || i == 1)

{

putchar(' ');

color++;

}

}

SetConsoleTextAttribute(hConsole, 7);

cout << endl;

}

void Third()

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

int color = 11;

union

{

int tool;

float F;

};

int order = sizeof(float) \* 8;

unsigned int mask = 1 << order - 1;

cout << "Введите число типа float" << endl;

cin >> F;

while (cin.fail())

{

cin.clear();

cin.ignore(1000, '\n');

cout << "Введите ЧИСЛО" << endl;

cin >> F;

}

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

{

SetConsoleTextAttribute(hConsole, color);

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

mask >>= 1;

if (i == 9 || i == 1)

{

putchar(' ');

color++;

}

}

SetConsoleTextAttribute(hConsole, 7);

cout << endl;

}

void Fourth()

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

int color = 11;

union {

int tool[2];

double D;

};

int order = sizeof(D) \* 8;

unsigned int mask = 1;

mask <<= order - 1;

cout << "Введите число типа double" << endl;

cin >> D;

while (cin.fail())

{

cin.clear();

cin.ignore(1000, '\n');

cout << "Введите ЧИСЛО" << endl;

cin >> D;

}

for (int i = 1; i >= 0; --i)

{

for (int j = 0; j < order / 2; ++j)

{

SetConsoleTextAttribute(hConsole, color);

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

mask >>= 1;

if ((i == 1 && j == 0) || (i == 1 && j == 11))

{

putchar(' ');

color++;

}

}

mask = 1 << order - 1;

}

SetConsoleTextAttribute(hConsole, 7);

cout << endl;

}

int main()

{

setlocale(0, "");

int number;

while (true)

{

cout << "Введите номер задания от 1 до 4" << endl;

cin >> number;

while (cin.fail())

{

cin.clear();

cin.ignore(1000, '\n');

cout << "Введите ЧИСЛО" << endl;

cin >> number;

}

switch (number)

{

case 1:

First();

break;

case 2:

Second();

break;

case 3:

Third();

break;

case 4:

Fourth();

break;

case 666:

return 0;

default:

cout << "Задания с таким номером нет" << endl;

}

}

}