**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**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 4373 |  | Музыченко И.А. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2024

**Цель работы.**

Изучение типов данных и их внутреннего представления в памяти, получение практических навыков работы с разными типами данных и памятью компьютера.

**Основные теоретические положения.**

**Составной тип: тип**, который не является скалярным типом. Составные типы включают типы массивов, типы функций, типы классов (или структуры), типы объединения, перечисления, ссылки и указатели на нестатические члены класса.

**Переменная**: символическое имя количества данных. Имя можно использовать для доступа к данным, которые он ссылается на всю область кода, в которой она определена. В C++ *переменная* часто используется для ссылки на экземпляры скалярных типов данных, а экземпляры других типов обычно называются *объектами*.

**Объект.** Для простоты и согласованности в этой статье используется объект*терминов*для ссылки на любой экземпляр класса или структуры. Если он используется в общем смысле, он включает все типы, даже скалярные переменные.

В отличие от некоторых других языков, в C++ нет универсального базового типа, от которого наследуются все остальные типы. Язык включает множество *фундаментальных типов*, также известных как *встроенные типы*. Эти типы включают числовые типы, такие как **int**, **double**, **bool**. Большинство целочисленных фундаментальных типовимеют **unsigned** версии, которые изменяют диапазон значений, которые переменная может хранить. Например, **int**32-разрядное целое число со знаком может представлять значение от -2 147 483 648 до 2 147 483 647. Объект **unsigned int**, который также хранится как 32 бита, может хранить значение от 0 до 4 294 967 295. Общее количество возможных значений в каждом случае одинаково, отличается только диапазон.

Все программы C++ должны иметь функцию main . При попытке скомпилировать программу C++ без main функции компилятор вызывает ошибку. (Библиотеки и static библиотеки динамической main компоновки не имеют функции.) Функция main заключается в том, где начинается выполнение исходного кода, но перед вводом main программы в функцию все static члены класса без явных инициализаторов заданы равным нулю. В Microsoft C++глобальные static объекты также инициализированы перед записью main.

Результатом **sizeof** оператора является тип size\_t, целочисленный тип, определенный в файле include <stddef.h>. Этот оператор позволяет избежать задания зависимых от компьютера размера данных в программах.

**Поточный ввод-вывод** в C++ выполняется с помощью функций сторонних библиотек. В С++, как и в С, нет встроенных в язык средств ввода-вывода.

В С для этих целей используется библиотека stdio.h. В С++ разработана новая библиотека ввода-вывода iostream, использующая концепцию объектно-ориентированного программирования:

#include <iostream>

Библиотека iostream определяет три стандартных потока:

* cin  стандартный входной поток (stdin в С)
* cout  стандартный выходной поток (stdout в С)
* cerr  стандартный поток вывода сообщений об ошибках (stderr в С)

Для их использования в Microsoft Visual Studio необходимо прописать строку:

using namespace std;

В противном случае каждый вызов функций cin, cout будет дополняться пространством имен std и оператором разрешения контекста ::

**std::cin >> Имя;  
std::cout << Имя;**

Для выполнения операций ввода-вывода переопределены две операции поразрядного сдвига:

* >>  получить из входного потока
* <<  поместить в выходной поток

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 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.

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.

**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. При выводе необходимо визуально обозначить знаковый разряд и значащие разряды отступами или цветом.

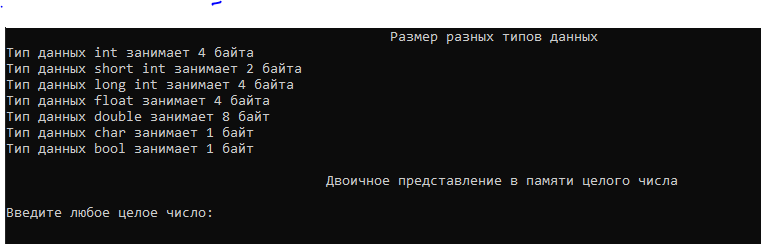
3) Вывести на экран двоичное представление в памяти (все разряды) типа float. При выводе необходимо визуально обозначить знаковый разряд мантиссы, знаковый разряд порядка (если есть), мантиссу и порядок.

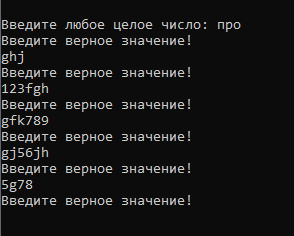
4) Вывести на экран двоичное представление в памяти (все разряды) типа double. При выводе необходимо визуально обозначить знаковый разряд мантиссы, знаковый разряд порядка (если есть), мантиссу и порядок.

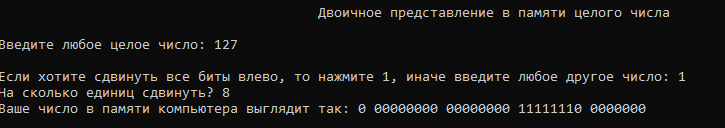
**Выполнение работы.**

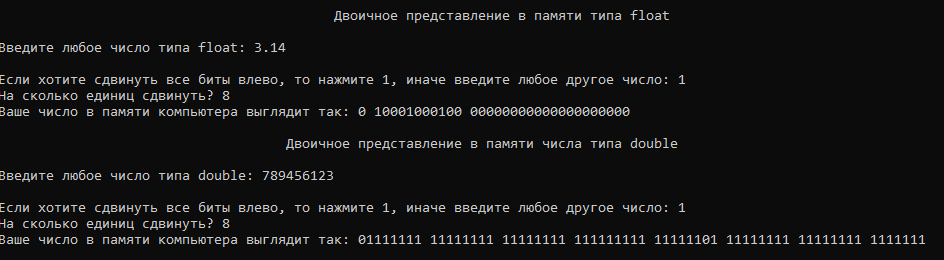
Код программы представлен в приложении А.

При запуске программы перед пользователем появляется окно, где он должен вводить значения.

 Проверка на ввод данных:



При вводе пользователем корректного числа программа продолжает свою работу. Далее пользователь может сдвинуть все биты влево на определенное значение единиц. Далее программа выводит двоичное представление в памяти заданного числа.

Те же действия повторяются для типа данных float и double.

**Выводы.**

Я узнала, в каком виде в компьютере хранятся разные типы данных, научилась выводить на консоль их двоичное представление в памяти, считывать данные с консоли и проверять их на ошибку ввода, использовать функции sizeof(), union, массивы и т.д.

Приложение А

рабочий код

﻿#include <iostream>

#include <iomanip>

using namespace std;

int sizeOfTypes() {

cout << setfill(' ') << setw(74) << "Размер разных типов данных" << "\n";

cout << "Тип данных int занимает " << sizeof(int) << " байта" << endl;

cout << "Тип данных short int занимает " << sizeof(short int) << " байта" << endl;

cout << "Тип данных long int занимает " << sizeof(long int) << " байта" << endl;

cout << "Тип данных float занимает " << sizeof(float) << " байта" << endl;

cout << "Тип данных double занимает " << sizeof(double) << " байт" << endl;

cout << "Тип данных char занимает " << sizeof(char) << " байт" << endl;

cout << "Тип данных bool занимает " << sizeof(bool) << " байт" << "\n";

return (0);

}

int mistake(int n) {

while (cin.fail() || cin.peek() != '\n') {

cout << "Введите верное значение!" << endl;

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cin >> n;

}

return n;

}

int mistakeFloat(float n) {

while (cin.fail() || cin.peek() != '\n') {

cout << "Введите верное значение!" << endl;

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cin >> n;

}

return n;

}

int moveToLeft(int n) {

cout << "\n" << "Если хотите сдвинуть все биты влево, то нажмите 1, иначе введите любое другое число: ";

int answer;

cin >> answer;

answer = mistake(answer);

if (answer) {

int amount;

cout << "На сколько единиц сдвинуть? ";

cin >> amount;

amount = mistake(amount);

n <<= amount;

}

return n;

}

int main() {

setlocale(0, "");

sizeOfTypes();

cout << "\n" << setfill(' ') << setw(85) << "Двоичное представление в памяти целого числа " << "\n";

cout << "\n" << "Введите любое целое число: ";

int number;

cin >> number;

number = mistake(number);

number = moveToLeft(number);

unsigned int mask = 1 << 31;

cout << "Ваше число в памяти компьютера выглядит так: ";

for (int i = 0; i < 32; i++) {

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

mask >>= 1;

if (i % 8 == 0 || i == 31) {

putchar(' ');

}

}

cout << endl;

cin.clear();

cout << "\n" << setfill(' ') << setw(85) << "Двоичное представление в памяти типа float " << endl;

cout << "\n" << "Введите любое число типа float: ";

union {

int tool;

float numberFloat;

};

cin >> numberFloat;

numberFloat = mistakeFloat(numberFloat);

numberFloat = moveToLeft(numberFloat);

unsigned int maskFloat = 1 << 31;

cout << "Ваше число в памяти компьютера выглядит так: ";

for (int i = 0; i < 32; i++) {

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

maskFloat >>= 1;

if (i == 0 || i == 11) {

putchar(' ');

}

}

cout << endl;

cin.clear();

cout << "\n" << setfill(' ') << setw(85) << "Двоичное представление в памяти числа типа double" << endl;

cout << "\n" << "Введите любое число типа double: ";

union {

unsigned int Double[sizeof(double)];

double numberDouble;

};

cin >> numberDouble;

numberDouble = mistake(numberDouble);

numberDouble = moveToLeft(numberDouble);

int firstMask = 1 << 31;

int secondMask = 1 << 31;

cout << "Ваше число в памяти компьютера выглядит так: ";

for (int i = 0; i < 32; i++) {

putchar(Double[i] & firstMask ? '1' : '0');

firstMask >>= 1;

if ((i-7) % 8 == 0 && i != 31) {

putchar(' ');

}

}

for (int j = 0; j < 32; j++) {

putchar(Double[32 + j] & secondMask ? '1' : '0');

secondMask >>= 1;

if (j % 8 == 0 || j == 0) {

putchar(' ');

}

}

cout << endl;

return(0);

}