**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 гр. 3373 |  | Долгий А. М. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2023

**Цель работы.**

Разработать алгоритм и написать программу, которая позволяет:

Научиться выводить сколько памяти (в байтах) на вашем компьютере отводится под различные типы данных со спецификаторами и без; выводить на экран двоичное представление в памяти (все разряды) целого и дробного чисел.

**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.**

Необходимо 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. (\*)

(\*) - Задание повышенной сложности. Подобные задания могут использовать методы и структуры следующих модулей.

**Выполнение работы.**

Код программы представлен в приложении А.

Для вывода объёма занимаемой памяти используется функция ShowSize, не имеющая аргументов и использующая встроенные функции cout и sizeof.

Далее определяются функции, работающие практически аналогично друг другу (IntToBin и FloatToBin):

На вход поступает число, определяются переменные для количества разрядов и маски (для побитового сравнения). Далее, в цикле for используется тритарная операция для сравнения числа с маской, а также вывод 0 или 1, которые представляют собой биты, и вывод разделителя через putchar.

Далее определяются IntAction и FloatAction функции, которые используются для зацикливания вызова функций перевода и упрощения функции main. Кроме того, здесь же определяется функция ChoseAction функция, которая выводит текстовое меню и на основе ввода либо вызывает соответствую функцию, либо повторяет попытку ввода (при некорректном вводе).

Скриншоты представлены далее в таблице

|  |  |
| --- | --- |
| Ввод пользователем и обработка данных | Работа алгоритма и вывод на экран |
| Меню | |
| При запуске программы перед пользователем появляется меню, где он может посмотреть, сколько памяти отводится на каждый тип данных, или выбрать тип данных для представления | Меню:    Проверка на ввод символов, которые не входят в диапазон выбора: |

Продолжение Таблицы

|  |  |
| --- | --- |
| Представление числа в памяти | |
| При вводе пользователем корректного значения пункта меню, пользователь может ввести число (целочисленное или вещественное, в зависимости от выбора в меню) | Как только введено число, программа выводит его представление в памяти и пользователь может еще раз ввести число: |

|  |  |
| --- | --- |
| Выход из программы | |
| Пользователь может ввести символ “X” или “x”, чтобы выйти из программы | Пример ввода буквы, который приводит к выходу (в режиме отладки, иначе консоль закрывается сразу): |

**Выводы.**

Я научился выводить сколько памяти (в байтах) на компьютере отводится под различные типы данных со спецификаторами и без и выводить на экран двоичное представление в памяти (все разряды) целого и дробного чисел.

Приложение А

рабочий код

#include <iostream>

#include <string>

#include <sstream>

#include <conio.h>

using namespace std;

bool isFloat( string myString ) {

std::istringstream iss(myString);

float f;

iss >> noskipws >> f; // noskipws considers leading whitespace invalid

// Check the entire string was consumed and if either failbit or badbit is set

return iss.eof() && !iss.fail();

}

//

// Пункт 1. Вывод того, сколько памяти (в байтах) на вашем компьютере отводится под различные типы данных со спецификаторами и без

// Для этого используется функция ShowSize, вызываемая в main

//

void ShowSize() {

cout << "Size of int: " << sizeof(int) << "\n";

cout << "Size of short int: " << sizeof(short int) << "\n";

cout << "Size of long int: " << sizeof(long int) << "\n";

cout << "Size of float: " << sizeof(float) << "\n";

cout << "Size of double: " << sizeof(double) << "\n";

cout << "Size of long double: " << sizeof(long double) << "\n";

cout << "Size of char: " << sizeof(char) << "\n";

cout << "Size of bool: " << sizeof(bool) << "\n";

}

//

// Пункт 2. Вывод на экран двоичного представления в памяти целого числа (все разряды)

// Пункт 3. Вывод на экран двоичное представление в памяти типа float (все разряды)

// Для этого используются функции IntToBin и FloatToBin соответсвенно, вызываемые в main

//

void IntToBin(int value,int customMove)

{

unsigned int order = sizeof(int)\*8; // Количество разрядов

unsigned int mask = 1 << order - 1; // Маска побитового сравнения

if (customMove != 0);

int maxint = INT\_MAX;

value = value & maxint;

value >>= customMove;

maxint = -INT\_MAX;

value = value | maxint;

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

{

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

mask >>= 1; // Побитовый сдвиг числа

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

{

putchar('δ'); // Разделитель

}

}

}

void FloatToBin(float deg, int customMove)

{

unsigned int order = 32; // Количество разрядов

unsigned int mask = 1 << order - 1;// Маска побитового сравнения

union

{

float valuefloat;

int value;

};

valuefloat = deg;

if (customMove != 0);

int maxint = INT\_MAX;

value = value & maxint;

value >>= customMove;

maxint = -INT\_MAX;

value = value | maxint;

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

{

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

mask >>= 1; // Побитовый сдвиг числа

if (i == 1 || i == 2 || i == 9)

putchar('δ'); // Разделитель

}

}

//

// Action функции, используемые для создания меню и упрощения функции main

//

int IntAction() {

string deg;

char strbuffer[\_MAX\_ITOSTR\_BASE10\_COUNT];

int intI;

int customMove;

string customMoveInput;

//Input Number

cout << "Enter X to exit" << "\n" << "Enter valid integer" << "\n\n";

intAsk:

cin >> deg;

intI = atoi(deg.c\_str());

if (deg == "x" || deg == "X") { return(0); }

if (deg != to\_string(intI)) { cout << "\nInvalid integer entered, try again: \n"; goto intAsk; }

//Input customMove

BitwiseMoveAsk:

cout << "Enter valid custom bitwise move number\n";

cin >> customMoveInput;

customMove = atoi(customMoveInput.c\_str());

if (customMoveInput == "x" || customMoveInput == "X") { return(0); }

if (customMoveInput != to\_string(customMove)) { cout << "\nInvalid bitwise move number entered, try again: \n"; goto BitwiseMoveAsk; }

while (intI != NULL and intI <= INT\_MAX and intI >= INT\_MIN) { // Цикл выполняется, пока deg - валидное целочисленное

IntToBin(intI, customMove);

cout << "\n" << "Address: " << &intI << "\n\n";

loopintAsk:

cin >> deg;

intI = atoi(deg.c\_str());

if (deg == "x" || deg == "X") { return(0); }

if (deg != to\_string(intI)) { cout << "\nInvalid integer entered, try again: \n\n"; goto loopintAsk; }

loopBitwiseMoveAsk:

cout << "Enter valid custom bitwise move number\n";

cin >> customMoveInput;

customMove = atoi(customMoveInput.c\_str());

if (customMoveInput == "x" || customMoveInput == "X") { return(0); }

if (customMoveInput != to\_string(customMove)) { cout << "\nInvalid bitwise move number entered, try again: \n"; goto loopBitwiseMoveAsk; }

}

}

int FloatAction() {

string deg;

char strbuffer[\_MAX\_ITOSTR\_BASE10\_COUNT];

float floatF;

int customMove;

string customMoveInput;

//Input Number

cout << "Enter X to exit" << "\n" << "Enter valid float" << "\n\n";

floatAsk:

cin >> deg;

floatF = stof(deg.c\_str());

if (deg == "x" || deg == "X") { return(0); }

if (!isFloat(deg)) { cout << "\nInvalid float entered, try again: \n\n"; goto floatAsk; }

//Input customMove

BitwiseMoveAsk:

cout << "Enter valid custom bitwise move number\n";

cin >> customMoveInput;

customMove = atoi(customMoveInput.c\_str());

if (customMoveInput == "x" || customMoveInput == "X") { return(0); }

if (customMoveInput != to\_string(customMove)) { cout << "\nInvalid bitwise move number entered, try again: \n"; goto BitwiseMoveAsk; }

while (floatF != NULL and floatF <= INT\_MAX and floatF >= INT\_MIN) { // Цикл выполняется, пока deg - валидное вещественное

FloatToBin(floatF, customMove);

cout << "\n" << "Address: " << &floatF << "\n\n";

loopfloatAsk:

cin >> deg;

floatF = atof(deg.c\_str());

if (deg == "x" || deg == "X") { return(0); }

if (!isFloat(deg)) { cout << "\nInvalid float entered, try again: \n"; goto loopfloatAsk; }

loopBitwiseMoveAsk:

cout << "Enter valid custom bitwise move number\n";

cin >> customMoveInput;

customMove = atoi(customMoveInput.c\_str());

if (customMoveInput == "x" || customMoveInput == "X") { return(0); }

if (customMoveInput != to\_string(customMove)) { cout << "\nInvalid bitwise move number entered, try again: \n"; goto loopBitwiseMoveAsk; }

}

}

void ChoseAction() {

Chosequestion:

cout << "Chose one of the following options:\n1. Show size of the data types\n2. Integer to RAM view\n3. Float to RAM view\nX. Exit\n\n"; // Вывод меню

char NumberTypeInput;

NumberTypeInput = \_getch();

switch (NumberTypeInput) { // При вводе корректного варианта консоль очищается и вызывается соответствующая функция, иначе снова предлагается выбор

case('1'): system("cls"); ShowSize(); break;

case('2'): system("cls"); IntAction(); break;

case('3'): system("cls"); FloatAction(); break;

case('x'): system("cls"); break;

case('X'): system("cls"); break;

/\*case('q'): system("cls"); break; Добавить выход по q?

case('Q'): system("cls"); break;\*/

default: goto Chosequestion; break;

}

}

//

// Функция main, в которой вызываются вышеописанные функции

//

int main()

{

Mainstart:

ChoseAction(); // Вывод меню с выбором последующего действия

cout << "\nDo you want to exit the program?\nType in y/Y to exit, otherwise n/N\n\n";

char ExitTypeInput;

Exitquestion:

ExitTypeInput = \_getch();

switch (ExitTypeInput) {

case('y'): break;

case('Y'): break;

case('n'): system("cls"); goto Mainstart; break;

case('N'): system("cls"); goto Mainstart; break;

default: cout << "Enter valid option [y/Y] [n/N]\n\n"; goto Exitquestion;

}

}