**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 представление в памяти

|  |  |  |
| --- | --- | --- |
| Студент гр. 3373 |  | Кубасов Д.И. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2023

**Цель работы.**

Изучение и организация типов данных; получение практических навыков работы с типами данных; изучение преставления в памяти компьютера переменных различных типов.

**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.

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).

Дополнительный код получается образованием обратного кода с последующим прибавлением единицы к его младшему разряду.

Увидеть, каким образом тип данных представляется на компьютере, можно при помощи логических операций: побитового сдвига (<<) и поразрядной конъюнкции (&). Пример:

cout(a & mask ? 1 : 0); // если 1, то возвращается 1, иначе 0

a <<= 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 a. 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 (исходным 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.

Изображение выглядит как текст, снимок экрана, линия, Шрифт

Автоматически созданное описание

Рисунок 1 – Представление переменной типа «double» в памяти компьютера

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.

**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 А.

Сама программа состоит из главной функции main:

int main()

{

setlocale(0, "");

int ff = 1;

int menu, a;

int order = sizeof(int) \* 8;

unsigned int mask = 1 << order - 1;

union {

int integerA;

float floatB = 3.14;

};

union {

int integerD[2];

double doubleB = 3.14;

};

while (ff == 1) {

cout << "Какой пункт вы хотите запустить?\n"

<< "1 - Вывести размера типов данных\n"

<< "2 - Вывести представление int в памяти\n"

<< "3 - Вывести представление float в памяти\n"

<< "4 - Вывести представление double в памяти\n"

<< "5 - Выход\n\n";

cin >> menu;

switch (menu)

{

case 1:

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

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

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

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

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

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

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

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

break;

case 2:

cout << "Введите число типа int:\n";

cin >> a;

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

cout << ((a & mask) ? 1 : 0);

mask = mask >> 1;

if (i == 0 or i == 7 or i == 15 or i == 23) {

cout << ' ';

}

}

cout << "\n\n";

break;

case 3:

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

cin >> floatB;

mask = 1 << order - 1;

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

cout << ((integerA & mask) ? 1 : 0);

mask = mask >> 1;

if (i == 0 or i == 1 or i == 8) {

cout << ' ';

}

}

cout << "\n\n";

break;

case 4:

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

cin >> doubleB;

mask = 1 << order - 1;

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

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

cout << ((integerD[i] & mask) ? 1 : 0);

mask = mask >> 1;

if (i == 1 && (j == 0 or j == 1 or j == 11)) {

cout << ' ';

}

}

mask = 1 << order - 1;

}

cout << "\n\n";

break;

case 5:

exit(1);

break;

default:

cout << "Ошибка! Попробуйте еще раз.";

cout << "\n\n";

break;

}

}

}

Данная процедура выводит список выполняемых пунктов и предлагает пользователю выбрать один из них.

При выборе 1-го пункта программа переходит в case 1. В нем с помощью функции sizeof выводятся размеры всех типов данных.

При выборе 2-го пункта программа переходит в case 2. Далее программа запрашивает у пользователя число и с помощью побитового сравнения выводит битовое представление данного числа в компьютере. Также для читаемости все разряды разделяются пробелами: отдельно бит знака и далее оставшиеся биты по 8.

При выборе 3-го пункта программа переходит в case 3. Для начала программа с помощью директивы union выделяет память под хранение двух переменных: вещественного и целого типа. Далее программа запрашивает у пользователя вещественное число и с помощью побитового сравнения выводит битовое представление данного числа в компьютере. Также для читаемости все разряды разделяются пробелами: отдельно бит знака мантиссы, бит знака порядка, а также отдельно биты порядка и биты мантиссы.

При выборе 4-го пункта программа переходит в case 4. Для начала программа с помощью директивы union выделяет память под хранение двух переменных: вещественного числа и массива из двух элементов. Далее программа запрашивает у пользователя вещественное число и с помощью побитового сравнения каждого из элементов выводит битовое представление данного числа в компьютере. Также для читаемости все разряды разделяются пробелами: отдельно бит знака мантиссы, бит знака порядка, а также отдельно биты порядка и биты мантиссы.

При выборе 5-го пункта программа завершает своё выполнение.

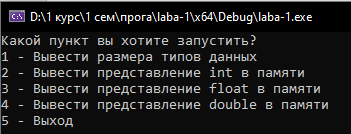


Рисунок 1 – Вывод меню

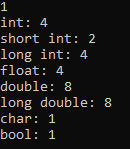


Рисунок 2 –Вывод размеров типов данных

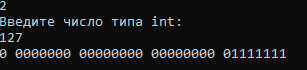


Рисунок 3 –Вывод представления целого числа в памяти компьютера с примером числа «127»

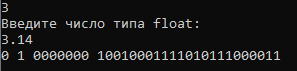


Рисунок 4 – Вывод представления вещественного числа в памяти компьютера с примером числа «3.14»

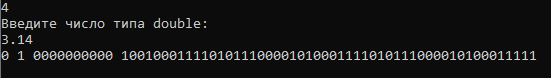


Рисунок 5 –Вывод представления вещественного числа двойной точности в памяти компьютера с примером числа «3.14»

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тип | Число | Представление в компьютере |
| 1 | Int | 127 | 0 0000000 00000000 00000000 01111111 |
| 2 | Float | 3,14 | 0 1 0000000 10010001111010111000011 |
| 3 | Double | 3,14 | 0 1 0000000000 1001000111101011100001010001111010111000010100011111 |

Таблица 1 – Примеры представлений некоторых чисел

**Выводы**

По результатам выполнения данной лабораторной работы:

1. Я узнал, какую размерность имеют переменные в моей среде программирования.
2. Я разобрался, как представляются целые и вещественные числа в моей среде программирования
3. Я выяснил, что отрицательные целые числа в моей среде программирования задаются в дополнительном коде, а порядок в вещественных числах – смещенный.

Приложение А

рабочий код

#include <iostream>

using namespace std;

int main()

{

setlocale(0, "");

int ff = 1;

int menu, a;

int order = sizeof(int) \* 8;

unsigned int mask = 1 << order - 1;

union {

int integerA;

float floatB = 3.14;

};

union {

int integerD[2];

double doubleB = 3.14;

};

while (ff == 1) {

cout << "Какой пункт вы хотите запустить?\n"

<< "1 - Вывести размера типов данных\n"

<< "2 - Вывести представление int в памяти\n"

<< "3 - Вывести представление float в памяти\n"

<< "4 - Вывести представление double в памяти\n"

<< "5 - Выход\n\n";

cin >> menu;

switch (menu)

{

case 1:

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

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

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

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

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

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

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

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

break;

case 2:

cout << "Введите число типа int:\n";

cin >> a;

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

cout << ((a & mask) ? 1 : 0);

mask = mask >> 1;

if (i == 0 or i == 7 or i == 15 or i == 23) {

cout << ' ';

}

}

cout << "\n\n";

break;

case 3:

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

cin >> floatB;

mask = 1 << order - 1;

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

cout << ((integerA & mask) ? 1 : 0);

mask = mask >> 1;

if (i == 0 or i == 1 or i == 8) {

cout << ' ';

}

}

cout << "\n\n";

break;

case 4:

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

cin >> doubleB;

mask = 1 << order - 1;

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

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

cout << ((integerD[i] & mask) ? 1 : 0);

mask = mask >> 1;

if (i == 1 && (j == 0 or j == 1 or j == 11)) {

cout << ' ';

}

}

mask = 1 << order - 1;

}

cout << "\n\n";

break;

case 5:

exit(1);

break;

default:

cout << "Ошибка! Попробуйте еще раз.";

cout << "\n\n";

break;

}

}

}