**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. 4372 |  | Максимов Е. В. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2024

**Цель работы.**

Разработать алгоритм и создать программу, которая будет отображать объем памяти, выделяемой для различных типов данных. Программа должна выводить на экран двоичное представление целого числа во всех его разрядах, а также аналогичные данные для типов float и double.

**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.

Алгоритм представления double немного отличается. Под вещественное число с двойной точностью отводится 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.

| Ввод пользователем и обработка данных | Работа алгоритма и вывод на экран |
| --- | --- |
| Выбор выполняемого задания | |
| При запуске программы перед пользователем появляется выбор задания. | После выбора задания выполняется переход на это задание. |
| Вывод количества памяти (в байтах), отводимого под различные типы данных со спецификаторами и без. | |
| Пользователь вводит тип данных, количество памяти отведенного под которое, он хочет узнать, затем выводится количество памяти. | Размер типа определяется оператором sizeof(), вывод представлен на рисунке: |
| Вывод на экран двоичного представления в памяти целого числа | |
| Пользователь вводит целое число | Применение битового умножения и сдвигов позволяет вывести на экран отображение целого числа в памяти: |
| Вывод на экран двоичного представления в памяти вещественного числа | |
| Пользователь вводит вещественное число | Применение битового умножения и сдвигов позволяет вывести на экран отображение вещественного числа в памяти: | |
| Вывод на экран двоичного представления в памяти (все разряды) типа double. | |
| Пользователь вводит вещественное число двойной точности | Применение битового умножения и сдвигов позволяет вывести на экран отображение вещественного числа двойной точности в памяти. Используется массив из двух целый чисел, размер которого равен типу double: | |
| Изменение произвольных битов | |
| Пользователь вводит номер и значение бита в числе | Изменение бита с указанным номером на указанное значение для всех введённых ранее чисел с возможностью повторить операцию: | |

**Постановка задачи.**

Необходимо разработать алгоритм и написать программу, которая будет определять и выводить на экран объём памяти (в байтах), выделяемой на компьютере для различных типов данных:

* int, short int, long int;
* float, double, long double;
* char;
* bool.

Также программа должна выводить на экран двоичное представление (все разряды) целых чисел, чисел типа float и double.

**Выводы.**

В ходе работы были изучены различные типы данных, включая те, что не имеют спецификаторов, а также разряды чисел. Также были освоены логические операции, такие как побитовый сдвиг (<<) и поразрядная конъюнкция (&).

Кроме того, была создана программа, способная определять объем памяти (в байтах), выделяемый для различных типов данных. Она выводит на экран двоичное представление целых чисел, а также типов float и double в памяти.

Приложение А

рабочий код

#include <iostream>

#include <conio.h>

#include <string>

using namespace std;

int main() {

setlocale(0, "");

char num\_zad;

cout << "\033[32m"<< "Введите номер задания или \"q\" для завершения работы: " << "\33[0m";

cin >> num\_zad;

if (num\_zad != '1' && num\_zad != '2' && num\_zad != '3' && num\_zad != '4' && num\_zad != 'q')

num\_zad = 'e';

while (num\_zad != 'q') {

// 1 задание

if (num\_zad=='1')

{

string h, n;

int t;

int mark=0;

cout << "\033[33m"<<"Введите тип данных: " << "\033[31m";

cin >> h;

if (h == "short" || h == "long") {

cin >> n;

t = 1;

}

else

t = 0;

if (t) {

if (n == "int" && h == "short") {

cout << "short int " << "\033[0m" << "- " << sizeof(short int) << " байт\n";

mark = 1;

}

if (n == "int" && h == "long") {

cout << "long int " << "\033[0m" << "- " << sizeof(long int) << " байт\n";

mark = 1;

}

if (n == "double" && h == "long") {

cout << "long double " << "\033[0m" << "- " << sizeof(long double) << " байт\n";

mark = 1;

}

}

else {

if (h == "int") {

cout << "int " << "\033[0m" << "- " << sizeof(int) << " байт\n";

mark = 1;

}

if (h == "float") {

cout << "float " << "\033[0m" << "- " << sizeof(float) << " байт\n";

mark = 1;

}

if (h == "double") {

cout << "double " << "\033[0m" << "- " << sizeof(double) << " байт\n";

mark = 1;

}

if (h == "char") {

cout << "char " << "\033[0m" << "- " << sizeof(char) << " байт\n";

mark = 1;

}

if (h == "bool") {

cout << "bool " << "\033[0m" << "- " << sizeof(bool) << " байт\n";

mark = 1;

}

if (mark==0)

cout << "\033[31mНекорректный ввод!\033[0m";

}

cout << endl;

cout << "\033[32m" << "Введите номер задания или \"q\" для завершения работы: " << "\33[0m";

cin >> num\_zad;

if (num\_zad != '1' && num\_zad != '2' && num\_zad != '3' && num\_zad != '4' && num\_zad != 'q')

num\_zad = 'e';

}

// 2 задание

if (num\_zad == '2')

{

int answer2;

int bit\_number2;

char bit\_value2;

int number1;

int mas2;

cout << "\033[33m" << "Введите целое число: "<<"\033[0m";

cin >> number1;

cout << endl;

unsigned int(mask1) = 1 << 31;

cout << "Двоичное представление в памяти: " << "\033[31m";

for (int i = 32; i != 0; --i) {

putchar(number1 & mask1 ? '1' : '0');

if (i == 32 || i % 8 == 1) {

cout << " ";

if (i == 32)

cout << "\033[35m";

}

mask1 >>= 1;

}

cout << endl << endl;

cout << "\033[0mЕсли хотите заменить бит, введите \033[33m1\033[0m, иначе введите \033[31m0\033[0m: ";

cin >> answer2;

while (answer2 == 1) {

cout << endl << "Введите номер бита: ";

cin >> bit\_number2;

cout << endl;

cout << "Введите на что заменить бит: ";

cin >> bit\_value2;

cout << endl;

mask1 = 1 << 31;

if ((bit\_value2 == '1') or (bit\_value2 == '0')) {

if (bit\_value2 == '1') {

mas2 = 1 << 32 - 1 - bit\_number2;

number1 = number1 ^ mas2;

cout << number1 << endl;

for (int i = 32; i != 0; --i) {

putchar(number1 & mask1 ? '1' : '0');

if (i == 32 || i % 8 == 1) {

cout << " ";

if (i == 32)

cout << "\033[35m";

}

mask1 >>= 1;

}

cout << "\033[31m";

}

else {

mas2 = ~(1 << 32 - 1- bit\_number2);

number1 = (number1 & mas2);

cout << number1 << endl;

for (int i = 32; i != 0; --i) {

putchar(number1 & mask1 ? '1' : '0');

if (i == 32 || i % 8 == 1) {

cout << " ";

if (i == 32)

cout << "\033[35m";

}

mask1 >>= 1;

}

cout << "\033[31m";

}

}

else {

cout << "\033[31mНекорректный ввод!\033[0m";

break;

}

cout << endl << endl << "Если хотите ещё раз заменить бит, введите 1, иначе введите \033[31m0\033[0m: ";

cin >> answer2;

}

cout << endl<<endl;

cout << "\033[32m" << "Введите номер задания или \"q\" для завершения работы: " << "\33[0m";

cin >> num\_zad;

if (num\_zad != '1' && num\_zad != '2' && num\_zad != '3' && num\_zad != '4' && num\_zad != 'q')

num\_zad = 'e';

}

// 3 задание

if (num\_zad == '3')

{

char mas3[33]{};

int answer3;

int bit\_number3;

char bit\_value3;

union {

int b;

float number2;

};

cout << "\033[33m" << "Введите вещественное число: "<<"\033[0m";

cin >> number2;

cout << endl;

unsigned int(mask2) = 1 << 31;

cout << "Двоичное представление в памяти: "<<"\033[31m";

for (int i = 32; i != 0; --i) {

putchar(b & mask2 ? '1' : '0');

if (b & mask2)

mas3[32 - i] = '1';

else

mas3[32- i] = '0';

if (i == 32 || i == 32 - 8 || i == 32 - 8 \* 2 || i == 32 - 8 \* 3) {

cout << " ";

if (i == 32)

cout << "\033[36m";

if (i==32-8)

cout << "\033[35m";

}

mask2 >>= 1;

}

cout << endl << endl;

cout << "\033[0mЕсли хотите заменить бит, введите \033[33m1\033[0m, иначе введите \033[31m0\033[0m: ";

cin >> answer3;

while (answer3 == 1) {

cout << endl << "Введите номер бита: ";

cin >> bit\_number3;

cout << endl;

cout << "Введите на что заменить бит: ";

cin >> bit\_value3;

cout << endl;

if ((bit\_value3 == '1') or (bit\_value3 == '0')) {

mas3[bit\_number3 - 1] = bit\_value3;

cout << "\033[31m";

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

cout << mas3[i];

if (i == 0 || i == 8 || i == 8 \* 2 || i == 8 \* 3) {

if (i == 0)

cout << "\033[36m";

if (i == 8)

cout << "\033[35m";

cout << " ";

}

}

}

else {

cout << "\033[31mНекорректный ввод!\033[0m";

break;

}

cout << endl <<endl<< "Если хотите ещё раз заменить бит, введите 1, иначе введите \033[31m0\033[0m: ";

cin >> answer3;

}

cout << endl << endl;

cout << "\033[32m" << "Введите номер задания или \"q\" для завершения работы: " << "\33[0m";

cin >> num\_zad;

if (num\_zad != '1' && num\_zad != '2' && num\_zad != '3' && num\_zad != '4' && num\_zad != 'q')

num\_zad = 'e';

}

//4 задание

if (num\_zad == '4') {

cout << endl;

char mas4[64];

int answer4;

int bit\_number4;

char bit\_value4;

union {

double c;

int d[2];

};

cout << "\033[33m" << "Введите вещественное число: " << "\033[0m";

cin >> c;

cout << endl<<"Двоичное представление в памяти: " << "\033[31m";

int db = 0;

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

unsigned int mask3 = 1 << 31;

for (int i = 32; i != 0; --i) {

db++;

putchar(d[j] & mask3 ? '1' : '0');

if (d[j] & mask3)

mas4[32 + 32 \* (1 - j) - i] = '1';

else

mas4[32 + 32 \* (1 - j) - i] = '0';

if (db == 1 || db == 12) {

cout << " ";

if (db==1)

cout << "\033[36m";

if (db == 12)

cout << "\033[35m";

}

mask3 >>= 1;

}

}

cout << endl<<endl;

cout << "\033[0mЕсли хотите заменить бит, введите \033[33m1\033[0m, иначе введите \033[31m0\033[0m: ";

cin >> answer4;

while (answer4 == 1) {

cout <<endl<< "Введите номер бита: ";

cin >> bit\_number4;

cout << endl;

cout << "Введите на что заменить бит: ";

cin >> bit\_value4;

cout << endl;

if ((bit\_value4 == '1') or (bit\_value4 == '0')) {

mas4[bit\_number4 - 1] = bit\_value4;

cout << "\033[31m";

for (int i = 0; i != 64; ++i) {

if (i == 1 || i == 12) {

cout << " ";

if (i == 1)

cout << "\033[36m";

if (i == 12)

cout << "\033[35m";

}

cout << mas4[i];

}

}

else {

cout << "\033[31mНекорректный ввод!\033[0m";

break;

}

cout << endl << endl<<"Если хотите ещё раз заменить бит, введите 1, иначе введите \033[31m0\033[0m: ";

cin >> answer4;

}

cout << endl << endl;

cout << "\033[32m" << "Введите номер задания или \"q\" для завершения работы: " << "\33[0m";

cin >> num\_zad;

if (num\_zad != '1' && num\_zad != '2' && num\_zad != '3' && num\_zad != '4' && num\_zad != 'q')

num\_zad = 'e';

}

if (num\_zad =='e') {

cout << "\033[31mНекорректный ввод!\033[0m";

cout << endl << endl;

cout << "\033[32m" << "Введите номер задания или \"q\" для завершения работы: " << "\33[0m";

cin >> num\_zad;

if (num\_zad != '1' && num\_zad != '2' && num\_zad != '3' && num\_zad != '4' && num\_zad != 'q')

num\_zad = 'e';

cin.sync();

}

}

return(0);

}