**онлайн-Платформа STEPIK**

**отчет**

**по практической работе №1**

**по курсу «Основы алгоритмизации и программирования на языке C++»**

**Тема: Типы данных и их внутреннее представление в памяти**

|  |  |  |
| --- | --- | --- |
| Студент(ка) гр. | Простаков А.В (User ID: 89907427) |  |
| Преподаватель | Глущенко А.Г. |  |

Москва

2023

**Цель работы.**

Изучение внутреннего представления в памяти различных типов данных и размера памяти, отводимого для их хранения.

**Основные теоретические положения.**

*Внутреннее представление величин целого типа* *int* – целое число в двоичном коде. Для кодирования целых чисел со знаком применяются прямой, обратный и дополнительный коды.

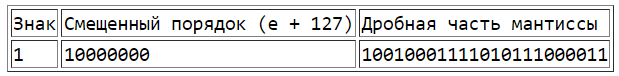
Для хранения в памяти величин целого типа *int* в памяти отводится 4 байта. Также существует тип данных величин целого типа со спецификатором *short*, для хранения которых в памяти отводится 2 байта независимо от разрядности процессора. Для 16-разрядного процессора под величину типа int отводится 2 байта, а при указании спецификатора *long* 4 байта. Надо отметить, что размер типа *int* не определяется стандартом, а зависит от компьютера и компилятора.

*Внутреннее представление величин вещественного типа* *float* и *double* значительно отличается от представления величин целого типа. Внутреннее представление вещественного числа состоит из трех частей *(рис. 1)*:

- Знаковый разряд

- Смещенный порядок

- Дробная часть мантиссы



*Рис. 1. Внутреннее представление типа данных float (32 бита) по стандарту* **IEEE 754** *на примере числа -3.14.*

Для наглядности приведем алгоритм получения всех трех частей для записи вещественного числа с плавающей точкой *-3.14* в двоичном коде:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| - | 3 | . | 14 | \* 10^0 |

*Учитываем порядок*

С учетом порядка: x = - 3.14

*Нормализация*

Нормализуем мантиссу (приведем к виду 1.xxx...x \* 2^exp)

|  |  |  |
| --- | --- | --- |
| Шаг | Мантисса | \* 2^двоичный порядок |
| 0 | 3.14 | \* 2^0 |
| 1 | 1.57 | \* 2^1 |

x = -1.57 \* 2^1

*Перевод в двоичный вид*

Переводим нормализованную мантиссу в двоичный вид (p = 23)

|  |  |  |
| --- | --- | --- |
| Шаг i | Двоичная мантисса | остаток \* 2^-i |
| 0 | 1. | + 0.57 \* 2^-0 |
| 1 | 1.1 | + 0.14 \* 2^-1 |
| 2 | 1.10 | + 0.28 \* 2^-2 |
| 3 | 1.100 | + 0.56 \* 2^-3 |
| 4 | 1.1001 | + 0.12 \* 2^-4 |
| 5 | 1.10010 | + 0.24 \* 2^-5 |
| 6 | 1.100100 | + 0.48 \* 2^-6 |
| 7 | 1.1001000 | + 0.96 \* 2^-7 |
| 8 | 1.10010001 | + 0.92 \* 2^-8 |
| 9 | 1.100100011 | + 0.84 \* 2^-9 |
| 10 | 1.1001000111 | + 0.68 \* 2^-10 |
| 11 | 1.10010001111 | + 0.36 \* 2^-11 |
| 12 | 1.100100011110 | + 0.72 \* 2^-12 |
| 13 | 1.1001000111101 | + 0.44 \* 2^-13 |
| 14 | 1.10010001111010 | + 0.88 \* 2^-14 |
| 15 | 1.100100011110101 | + 0.76 \* 2^-15 |
| 16 | 1.1001000111101011 | + 0.52 \* 2^-16 |
| 17 | 1.10010001111010111 | + 0.04 \* 2^-17 |
| 18 | 1.100100011110101110 | + 0.08 \* 2^-18 |
| 19 | 1.1001000111101011100 | + 0.16 \* 2^-19 |
| 20 | 1.10010001111010111000 | + 0.32 \* 2^-20 |
| 21 | 1.100100011110101110000 | + 0.64 \* 2^-21 |
| 22 | 1.1001000111101011100001 | + 0.28 \* 2^-22 |
| 23 | 1.10010001111010111000010 | + 0.56 \* 2^-23 |

*Округление (ieee-754)*

|  |  |  |  |
| --- | --- | --- | --- |
| Шаг | Мантисса | Бит p + 1 (=24) | Остаток (sticky) |
| 24 | 1.10010001111010111000010 | 1 | 0.12 \* 2^-24 |

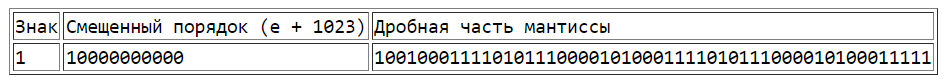
Остаток больше 0  
Увеличиваем мантиссу на lsb

*Результат*

Экспонента со смещением: 1 + 127 = 128

|  |  |  |
| --- | --- | --- |
| Знак | Смещенный порядок (e + 127) | Дробная часть мантиссы |
| 1 | 10000000 | 10010001111010111000011 |

Основное отличие представления типа данных *float* *и double* заключается в количестве бит данных, отводимых для значения порядка (8 и 11 бит для *float* и *double* соответственно) и для мантиссы (23 и 52 бита для *float* и *double* соответственно), а также в величине смещения для порядка (127 и 1023 для *float* и *double* соответственно). На *рис. 2* приведен пример представления в памяти числа -*3.14* с типом данных *double*.



*Рис.2. Рис. 1. Внутреннее представление типа данных double (64 бита) по стандарту* **IEEE 754** *на примере числа -3.14*

Для хранения в памяти величин целого типа *float* и *double* в памяти отводится 4 и 8 байт соответственно.

**Постановка задачи**

Разработать алгоритм и написать программу, которая позволяет:

Задача 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.

Задача 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.

Задача 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.

Задача 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 в редактируемом формате представлен в приложении А.

|  |
| --- |
| Блок описания кода и использованных алгоритмов. Скриншоты работы программы |
| Задача 1) |
| Определение объема хранения в байтах для каждого типа данных определяется с помощью оператора *sizeof*.  Переменная *value* объявляется с типом данных *int*. Далее осуществляется явное преобразование типа данных, которое имеет следующий формат: |
| Задача 2) |
| Для выведения в консоль двоичного представления типа данных *int* использовались побитовые логические операции: *побитовый сдвиг (<<)* и *поразрядная конъюнкция (&).* Каждый бит заданного числа «логически умножался» на «единичный бит» *маски-* числа с первым «ненулевым» битом, у которого число разрядов с помощью побитого сдвига влево делается равным числу разрядов «исследуемого» нами числа с типом данных *int*. |
| Задача 3) |
| Для выведения в консоль двоичного представления чисел типа *float* используется ключевое слово *union*, которое позволяет интерпретировать несколькими разными способами содержимое одной и той же области памяти. После того как мы «как бы» назначили двум переменным разного типа данных один и тот же участок памяти (фактически это конечно же одна и та же переменная, интерпретируемая разными способами), мы производим *побитовые* операции над целочисленной переменной из объединения *union* для вывода ее двоичного кода в консоль. |
| Задача 4) |
| Для выведения в консоль двоичного представления чисел типа *double* используется тот же прием, что и в задаче 3. Однако, числа типа *double* занимают 8 байт, поэтому объединение c одним числом типа *int* (4 байта) не позволяют вывести в консоль все число. Проблема решается с помощью массива из двух чисел типа *int*. Как известно, данные любой линейной структуры записываются в память последовательно, поэтому выводя оба элемента массива в консоль мы получаем полностью 8-байтовое представление числа *double* в памяти. Стоит отметить, что порядок записи байтов на моем компьютере обратный (*little endian*), поэтому массив выводится от последнего к первому элементу. |

**Выводы**

В данной практической работе рассмотрено представление в памяти различных типов данных, объем памяти, отводимой под каждый тип данных.

Также приведены приемы вывода в консоль двоичного представления числе всех типов, в том числе, чисел с плавающей точкой (стандарт IEEE 754 ).

**Приложение А**

**рабочий код**

|  |
| --- |
| *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.* |
| #include<iostream>;  using namespace std;  int main()  {  setlocale(LC\_ALL, "Rus");  int value;  cout << "Размер памяти (байт) для хранения типа данных int: " << sizeof(value) << endl;  cout << "Размер памяти (байт) для хранения типа данных short int: " << sizeof((short int)value) << endl;  cout << "Размер памяти (байт) для хранения типа данных long int: " << sizeof((long int)value) << endl;  cout << "Размер памяти (байт) для хранения типа данных float: " << sizeof((float)value) << endl;  cout << "Размер памяти (байт) для хранения типа данных double: " << sizeof((double)value) << endl;  cout << "Размер памяти (байт) для хранения типа данных long double: " << sizeof((long double)value) << endl;  cout << "Размер памяти (байт) для хранения типа данных char: " << sizeof((char)value) << endl;  cout << "Размер памяти (байт) для хранения типа данных bool: " << sizeof((bool)value) << endl;  } |
| *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.* |
| int main()  {  setlocale(LC\_ALL, "Rus");  int value = -127;  int temp;  int placeCount, k;  unsigned int order = 32;  unsigned int mask = 1 << order - 1;  k = 0;  if (value < 0)  {  k = 1;  }  temp = value;  do  {  placeCount = temp / 2;  temp = placeCount;  k = k + 1;  }  while (abs(placeCount) > 0);  cout << "Число значащих разрядов: " << k << endl;  for (int i = 1; i <= order; i++)  {  putchar(value & mask ? '1' : '0');  value <<= 1;  if (i == 1)  {  putchar(' ');  }  if (i== (order - k))  {  putchar(' ');  }  }  } |

|  |
| --- |
| *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.* |
| #include<iostream>;  using namespace std;  int main()  {  setlocale(LC\_ALL, "Rus");  unsigned int order = 32;  unsigned int mask = 1 << order - 1;  union {  int tool;  float numb\_f = -2.718;  };    cout <<"Число в десятичной системе счисления: "<<numb\_f << endl;    cout << " Число, записанное согласно стандарту ieee-754: ";  for (int i = 1; i <= order; i++)  {  putchar(tool & mask ? '1' : '0');  tool <<= 1;  if (i==1)  {  putchar(' ');  }  if (i == 9)  {  putchar(' ');  }  }  } |
| *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. (\*)* |
| #include<iostream>;  using namespace std;  int main()  {  setlocale(LC\_ALL, "Rus");  unsigned int order = 64;  unsigned int mask = 1 << order - 1;  union {  int arr[2];  double numb\_f = 228;  };  cout <<"Число в десятичной системе счисления: "<<numb\_f << endl;  cout << " Число, записанное согласно стандарту ieee-754: ";  for (int n = 1; n >=0; n--)  {  for (int i = 1; i <= order / 2; i++)  {  putchar(arr[n] & mask ? '1' : '0');  arr[n] <<= 1;  if ((n==1) && (i == 1))  {  putchar(' ');  }  if ((n==1) && (i == 12))  {  putchar(' ');  }  }  }  } |