МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Информационных систем

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Программирование»

Тема: ТИПЫ ДАННЫХ И ИХ ВНУТРЕННЕЕ ПРЕДСТАВЛЕНИЕ В ПАМЯТИ

Студент(ка) гр. 0324 _____ Косенко А.Р.

Преподаватель _____ Глущенко А.Г.

Санкт-Петербург

2020

Цель работы.

Знакомство с внутренним представлением различных типов данных, которые использует компьютер при их обработке.

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):

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Знак числа

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:

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0

Знак числа

Дополнительный код получается образованием обратного кода с последующим прибавлением единицы к его младшему разряду. Дополнительный код числа −3:

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1

Знак числа

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
```

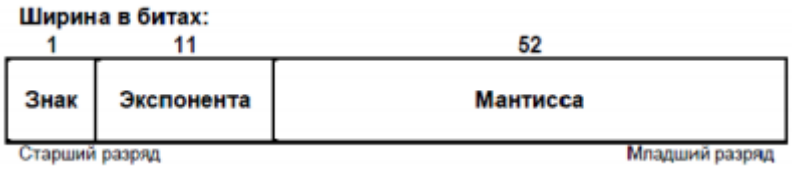
Putchar возвращает один символ в консоль. Альтернатива - cout. В представленном способе, маска - то, с чем сравнивается значение. И побитовый сдвиг применяется для value. Таким

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.

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.

Для выполнения первой работы я использовала оператор sizeoff, который вычислил тип данных и их размер в байтах.

Во второй задаче я создала маску, а далее использовала логические операций: побитовый сдвиг (<<) и поразрядную конъюнкцию (&). В результате вывода я получила двоичное число.

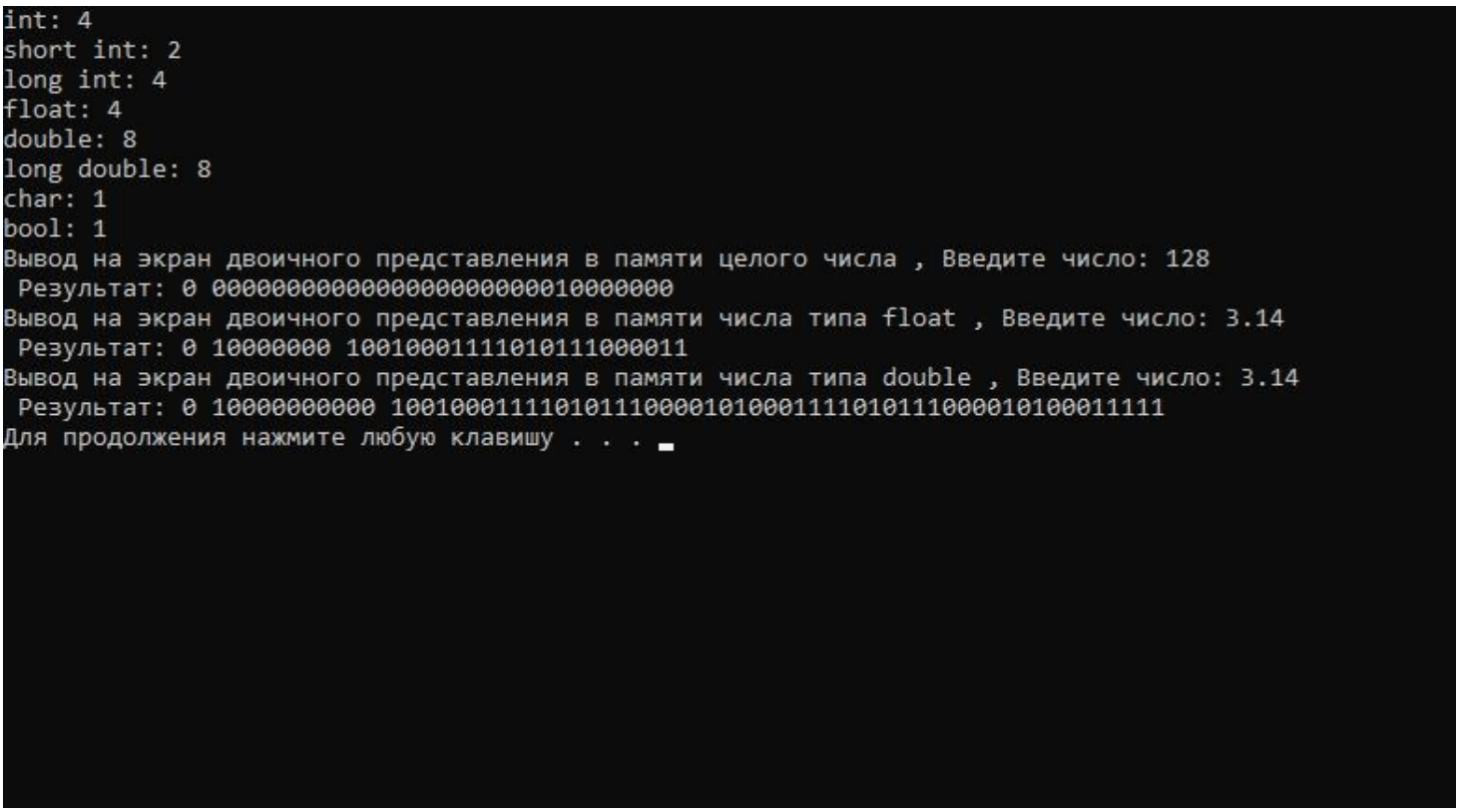
В третей задаче я использовала Union- объединение. Таким способом я посмотрела на тип на float глазами int, используя содержимое одной и той же области памяти. Далее я вводила новую переменную маски и делала побитовое перемножение со сдвигом этого значения.

В четвертой задаче я использовала массив из двух int, под который было отведено 8 байт. Далее , как в 3 задаче, я вводила новую переменную маски и делала побитовое перемножение со сдвигом этого значения.

```cpp
// Kosenko Anastasia, 0324
// https://github.com/nattakkoss/laba

#include <iostream>
using namespace std;
int main() {
setlocale(LC_ALL, "Russian"),
// 1 задание

cout<<"int: "<<sizeof(int)<<"\n";
cout<<"short int: "<<sizeof(short int)<<"\n";
cout<<"long int: "<<sizeof(long int)<<"\n";
cout<<"float: "<<sizeof(float)<<"\n";
cout<<"double: "<<sizeof(double)<<"\n";
cout<<"long double: "<<sizeof(long double)<<"\n";
cout<<"char: "<<sizeof(char)<<"\n";
cout<<"bool: "<<sizeof(bool)<<"\n";

// 2 задание
int h;
unsigned int mask = 1 << 31;
cout << "Вывод на экран двоичного представления в памяти целого числа , Введите число: ";
cin >> h;//Вводим целое число
cout << " Результат: ";
for (int i = 0; i <= 31; i++)
 {
 if (i == 1)
  {
   cout << " ";
  }
putchar(h & mask ? '1' : '0');//если 1, то возвращается 1, иначе 0
mask >>= 1;// побитовый сдвиг вправо на 1 бит
 }
cout << "\n";

// 3 задание

union //посмотрим на float глазами int, используя содержимое одной и той же области памяти
{
int a;
float b;
};

unsigned int maskfloat = 1 << 31;

cout << "Вывод на экран двоичного представления в памяти числа типа float , Введите число: ";
cin >> b;//Вводим число типа float
cout << " Результат: ";
for (int j = 0; j <= 31; j++)
{
  if (j == 1 || j == 9) //Визуально обозначаем знаковый разряд мантиссы, знаковый разряд порядка (если есть), мантиссу и порядок. Для float под мантиссу отводит
  {
   cout << " ";
  }
putchar(a & maskfloat ? '1' : '0');
a<<= 1; // побитовый сдвиг влево на 1 бит
}
cout << "\n";

// 4 задание

union
```

```
61
62  union
63    {
64    int arr[2]; // Используем массив из двух int, под который будет отведено 8 байт.
65    double k;
66    };
67  unsigned int doublemask = 1 << 31;
68  cout << "Вывод на экран двоичного представления в памяти числа типа double , Введите число: ";
69  cin >> k; // Вводим число типа double
70  cout << " Результат: ";
71  for (int l = 0; l<= 31; l++)
72    {
73    if (l== 1 || l== 12) // Визуально обозначаем знаковый разряд мантиссы, знаковый разряд порядка (если есть), мантиссу и порядок. Д
74     {
75      cout << " ";
76     }
77  putchar(arr[1] & doublemask ? '1' : '0');
78  arr[1] <<= 1;
79    }
80  for (int l = 0; l <= 31; l++)
81    {
82     putchar(arr[0] & doublemask? '1' : '0');
83     arr[0] <<= 1;
84    }
85  cout << "\n";
86  }
```

Результат работы программы.


int: 4
short int: 2
long int: 4
float: 4
double: 8
long double: 8
char: 1
bool: 1
Вывод на экран двоичного представления в памяти целого числа , Введите число: 128
Результат: 0 00000000000000000000000010000000
Вывод на экран двоичного представления в памяти числа типа float , Введите число: 3.14
Результат: 0 10000000 10010001111010111000011
Вывод на экран двоичного представления в памяти числа типа double , Введите число: 3.14
Результат: 0 10000000000 1001000111101011100001010001111010111000010100011111
Для продолжения нажмите любую клавишу . . .