

Отчет о выполнении лабы по флоатам

Калашников Михаил, Б03-205

0. unsigned int \rightarrow binary

```
void print_bin(unsigned int x) {
    unsigned int y;

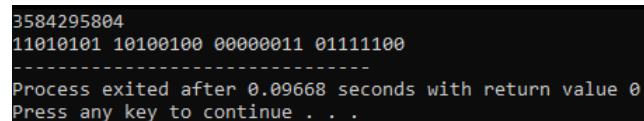
    y = -1;
    y >>= 1;
    y++;

    for (int i = 1; i < 33; i++) {
        cout << ((x & y) != 0);
        y >>= 1;

        if (i % 8 == 0) {
            cout << '_';
        }
    }
}
```

Данная функция выводит в терминал побитовое разложение числа путем сравнения каждой ячейки памяти, занятой числом с соответствующей степенью двойки. В начале создается $y = 2^{31}$. Логический побитовый оператор И сравнивает y с введенным числом. В случае, если результат отличен от нуля, бит, занятый единицей в y , был занят единицей и в введенном числе.

Пример работы функции:



```
3584295804
11010101 10100100 00000011 01111100
-----
Process exited after 0.09668 seconds with return value 0
Press any key to continue . . .
```

1. float \rightarrow binary

```
union float_bin {
    float f;
    unsigned int i;
};

int main()
{
    float a;
    float_bin fb;

    cin >> a;

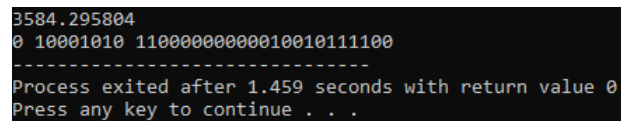
    fb.f = a;

    print_bin(fb.i);

    return 0;
}
```

В данной задаче используется все та же функция `print_bin()` из прошлого пункта. Для чтения побитового представления числа с плавающей точкой создадим объект `union float_bin`. Сохраним в него `float`, и передадим его в функцию как `unsigned int`.

Пример работы:



```
3584.295804
0 10001010 11000000000010010111100
-----
Process exited after 1.459 seconds with return value 0
Press any key to continue . . .
```

2. Переполнение мантиссы

```
int main() {
    float a = 1;
    float_bin fb;

    cout << fixed;
    cout.precision(2);

    for (int i = 0; i < 40; i++) {
        cout << i << ' ' << a << endl;

        fb.f = a;
        print_bin(fb.i);

        cout << endl << endl;

        a *= 10;
    }

    return 0;
}
```

Используя функцию `print_bin()` и структуру `float_bin`, будем выводить на экран побитовое разложение числа с плавающей точкой, увеличивая его в 10 раз на каждой итерации. При переходе с 10^{10} до 10^{11} , в переменную записывается число, отличное от ожидаемого результата. Это связано с дискретностью типа `float`. На 39 итерации переменная принимает значение `inf`.

Пример работы:

```
8 100000000.00
0 10011001 01111101011110000100000

9 100000000.00
0 10011100 11011100110101100101000

10 100000000.00
0 10100000 00101010000001011111001

11 99999997952.00
0 10100011 01110100100001110110111

12 99999995904.00
0 10100110 11010001101010010100101
```

```
37 10000000567641112624826207124609564672.00
0 11111001 11100001011110111000011

38 100000006944061726476491472742798852096.00
0 11111101 00101100111011010011010

39 inf
0 11111111 000000000000000000000000
```

3. Бесконечный цикл

```
#include <iostream>
using namespace std;

int main()
{
    cout << fixed;
    cout.precision(0);

    for (float i = 0; i <= 200000000; i++) {
        if (i + 1 == i) {
            cout << i << endl;
        }
    }

    return 0;
}
```

Напишем данную программу. Условие, при котором цикл станет бесконечным, выполнится при $i = 16777216 = 2^{24}$. Число $i + 1 = 16777217$ не может быть представлено переменной типа `float`.

4. График π

Для нахождения значения числа π я выбрал 4 формулы:

$$(a) \frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \dots = \frac{\pi}{4}$$

$$(b) \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \frac{1}{5^2} + \dots = \frac{\pi^2}{6}$$

$$(c) \frac{1}{1^2} - \frac{1}{2^2} + \frac{1}{3^2} - \frac{1}{4^2} + \frac{1}{5^2} - \dots = \frac{\pi^2}{12}$$

$$(d) \sum_{k=0}^{\infty} \frac{(-1)^k}{3^k(2k+1)} = \frac{\pi}{2\sqrt{3}}$$

Ниже приведен код, вычисляющий π по формуле (с).

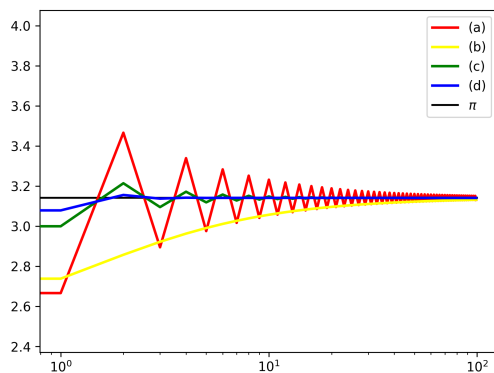
```
#include <iostream>
#include <fstream>
#include <cmath>
using namespace std;

int main()
{
    ofstream f("4_3.txt", ios::out);
    f << fixed;
    f.precision(22);

    float pi = 0, t, n;

    for (int i = 1; i < 10001; i++) {
        n = i;
        t = (i % 2 * 2 - 1);
        t = t / n / n;
        pi += t;
        f << i << '\t' << sqrt(pi * 12) << endl;
    }
    return 0;
}
```

С помощью библиотеки `matplotlib`, построим график зависимости значения вычисленного числа π от количества итераций.



5. Время π

Преобразуем скрипт из прошлого пункта так, чтобы в файл записывалось время и номер итерации при которых вычисляется каждая цифра числа π . Пример опять же приведу только для пункта (с):

```
#include <iostream>
#include <cmath>
#include <fstream>
#include <chrono>
using namespace std;

int PI[12] = {3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5, 9};

int main()
{
    ofstream f("5_3.txt", ios::out);

    f << fixed;
    f.precision(22);

    cout << fixed;
    cout.precision(22);

    float pi = 0, t, n;
    int dn = 0;
    unsigned long int d = 0, ten = 1;

    auto start = chrono::high_resolution_clock::now();

    for (int i = 1; i < 100001; i++) {
        n = i;

        t = (i % 2 * 2 - 1);
        t /= n * n;

        pi += t;

        d = sqrt(pi * 12) * ten;
        d %= 10;

        if (d == PI[dn]) {
            auto end = std::chrono::high_resolution_clock::now();
            int time = (end - start).count();
            f << dn + 1 << '\t' << i << '\t' << time << endl;
            cout << i << '\t' << dn + 1 << '\t' <<
                sqrt(pi * 12) << '\t' << time << endl;

            dn++;
            ten *= 10;
        }
    }
    return 0;
}
```

По полученным данным построим два графика – количество цифр числа π в зависимости от количества итераций и в зависимости от времени.

