

## Семинар #4: Типы данных. Домашнее задание.

### Основные типы и их обычные размеры на 64-х битных системах

| тип                | размер (байт) | диапазон значений ( $2^{\#bits}$ )                        | спецификатор |
|--------------------|---------------|---|--------------|
| char               | 1             | от -128 до 127  | %hi          |
| short              | 2             | от -32768 до 32767  | %hi          |
| int                | 4             | примерно от -2-х миллиардов до 2-х миллиардов             | %i           |
| long               | 4 или 8       | такой же как у int или long long в зависимости от системы | %li          |
| long long          | 8             | примерно от $-10^{19}$ до $10^{19}$                       | %lli         |
| unsigned char      | 1             | от 0 до 255   | %hu          |
| unsigned short     | 2             | от 0 до 65535   | %hu          |
| unsigned int       | 4             | от 0 до $2^{32} \approx 4 * 10^9$                         | %u           |
| unsigned long      | 4 или 8       | такой же как у unsigned int или unsigned long long        | %lu          |
| unsigned long long | 8             | от 0 до $2^{64} \approx 2 * 10^{19}$                      | %llu         |
| size_t             | 8             | от 0 до $2^{64} \approx 2 * 10^{19}$                      | %zu          |

| тип                                   | размер (байт) | значимые цифры | диапазон экспоненты  | спецификатор |
|---------------------------------------|---------------|----------------|----------------------|--------------|
| float                                 | 4             | 6              | от -38 до 38         | %f           |
| double                                | 8             | 15             | от -308 до 308       | %lf          |
| long double                           | от 8 до 16    | $\geq 15$      | не хуже чем у double | %Lf          |
| печать только 3-х чисел после запятой | -             | -              | -                    | %.3f         |
| печать без нулей на конце             | -             | -              | -                    | %g           |
| печать в научной записи               | -             | -              | -                    | %e           |

| тип       | размер (байт) | спецификатор |
|-----------|---------------|--------------|
| указатель | 8             | %p           |

### Задача 1. Факториал

Для вычисления факториала была написана следующая простая программа.

```
#include <stdio.h>
int fact(int n)
{
    int result = 1;
    for (int i = 1; i <= n; ++i)
        result *= i;
    return result;
}

int main()
{
    int k;
    scanf("%i", &k);
    printf("%i\n", fact(k));
}
```

Однако, выяснилось, что эта программа правильно работает только для  $k$  от 0 до 12. При больших  $k$  программа выдаёт неверный ответ. Почему это происходит? Немного измените программу, чтобы она работала для  $k$  до 20 включительно.

| ВХОД | ВЫХОД               |
|------|---------------------|
| 5    | 120                 |
| 13   | 6227020800          |
| 17   | 355687428096000     |
| 20   | 2432902008176640000 |

## Задача 2. Размещения

В комбинаторике размещением (из  $n$  по  $k$ )  $A_n^k$  называется упорядоченный набор из  $k$  различных элементов из некоторого множества различных  $n$  элементов. Размещения вычисляются следующим образом:  $A_n^k = \frac{n!}{(n-k)!}$ . Напишите программу, которая будет вычислять размещения при условии, что  $A_n^k < 2^{64}$ . Проверьте вашу функцию на следующих значениях:

| ВХОД  | ВЫХОД                |
|-------|----------------------|
| 5 2   | 20                   |
| 20 10 | 670442572800         |
| 30 12 | 41430393164160000    |
| 60 11 | 13679492361575040000 |

## Задача 3. Часть года

Напишите функцию, `float yearfrac(int year, int day)` которая принимает номер года `year` и номер дня с начала года `day` и возвращает прошедшую долю года. В этой задаче считайте, что високосный год, это год, чей номер делится на 4 (хотя это не совсем так).

| year | day | yearfrac(year, day) |
|------|-----|---------------------|
| 2019 | 300 | 0.82192             |
| 2019 | 100 | 0.27397             |
| 2020 | 100 | 0.27322             |

## Задача 4. Объём $n$ -мерного шара

Формула для  $n$ -мерного объёма  $n$ -мерного шара имеет вид:

$$V_n(R) = \begin{cases} \frac{2(\frac{n-1}{2})! \cdot (4\pi)^{\frac{n-1}{2}}}{n!} R^n, & \text{если } n - \text{нечётное} \\ \frac{\pi^{\frac{n}{2}}}{\frac{n}{2}!} R^n, & \text{если } n - \text{чётное} \end{cases}$$

Напишите программу, которая по заданному  $n$  будет вычислять отношение объёма  $n$ -мерного куба к объёму вписанному в него  $n$ -мерного шара, то есть  $\frac{(2R)^n}{V_n(R)}$ . Вам может понадобиться функция `pow` из библиотеки `math.h`.

| ВХОД | ВЫХОД        |
|------|--------------|
| 1    | 1            |
| 2    | 1.27324      |
| 3    | 1.909859     |
| 6    | 12.384589    |
| 10   | 401.542796   |
| 15   | 85905.301384 |

## Задача 5. Вычисление $\pi$

Известно, что число  $\pi$  можно вычислить с помощью следующего ряда:

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \dots = \sum_{i=1}^{\infty} \frac{(-1)^{i+1}}{2i-1}$$

Используйте эту формулу, чтобы вычислить приблизительно число  $\pi$ . На вход должно подаваться целое число `n` - число членов суммируемой последовательности, а вам нужно вычислить приближённое значение:

$$\pi \approx 4 \cdot \sum_{i=1}^n \frac{(-1)^{i+1}}{2i-1}$$

## Задача 6. Гамма-функция

Гамма-функция – это обобщение понятия факториала на вещественные числа. Определяется следующим образом:

$$\Gamma(x) = \int_0^{\infty} t^{x-1} e^{-t} dt$$

Легко вывести, что  $\Gamma(n) = (n-1)!$  для натуральных  $n$ . Написать функцию, `double gamma(double x)`, которая будет вычислять значение гамма-функции в точке  $x$ , при  $x > 1$ . Для вычисления интеграла использовать метод трапеций с шагом `step = 1e-2`. Суммирование продолжать до тех пор пока площадь трапеции превышает `eps = 1e-10` (то есть  $10^{-10}$ ). `step` и `eps` задать как константы. Понадобятся функции `pow` и `exp` из библиотеки `math.h`.

| ВХОД    | ВЫХОД       |
|---------|-------------|
| 2       | 1.0         |
| 6       | 120.0       |
| 20      | 1.21645e+17 |
| 1.5     | 0.88623     |
| 2.5     | 1.32934     |
| 4.14159 | 7.188082    |

## Задача 7. Угол

На вход программе поступают компоненты двух векторов. Нужно найти угол между ними в градусах.

| ВХОД | ВЫХОД   |
|------|---------|
| 1 0  | 90      |
| 0 1  |         |
| 1 0  | 45      |
| 1 1  |         |
| -1 0 | 135     |
| 1 1  |         |
| -2 8 | 74.2913 |
| 7 4  |         |

Угол  $\alpha$  между векторами можно найти из формул для скалярного произведения:

$$\vec{v} \cdot \vec{u} = |\vec{v}| |\vec{u}| \cos(\alpha)$$

$$\vec{v} \cdot \vec{u} = v_x u_x + v_y u_y$$

Вам могут понадобиться следующие функции:

```
double distance(double x1, double y1, double x2, double y2) {
    return sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2));
}
double length(double x, double y) {
    return distance(x, y, 0, 0);
}
double scalar_product(double x1, double y1, double x2, double y2) {
    return x1 * x2 + y1 * y2;
}
const double pi = 3.14159265359;
double to_degrees(double rad) {
    return rad * 180 / pi;
}
```

## Задача 8. Два круга

Напишите программу, которая проверяет пересекаются ли 2 круга. Программа должна принимать на вход координаты центров кругов и их радиусы в следующем порядке:

x1 y1 r1  
x2 y2 r2

и печатать следующее:

- **Do not intersect** – если окружности не пересекаются (нет ни одной общей точки).
- **Touch** – если круги касаются друг друга (с точностью  $\epsilon = 10^{-5}$ ).
- **Intersect** – если круги пересекаются

| ВХОД            | ВЫХОД            |
|-----------------|------------------|
| 0 0 1<br>0 2 1  | Touch            |
| 0 0 1<br>1 1 1  | Intersect        |
| 0 0 3<br>5 5 4  | Do not intersect |
| 0 0 4<br>5 5 4  | Intersect        |
| -2 1 4<br>2 4 1 | Touch            |

## Задача 9. Бинарный поиск на вещественных числах

Пусть у нас есть монотонно возрастающая функция  $f(x)$ , а наша задача заключается в том, чтобы найти решение уравнения  $f(x) = 0$  на отрезке  $(l, h)$ . Причём  $f(l) < 0$ , а  $f(h) > 0$ .

Для решения этой задачи можно применить метод бинарного поиска. Для этого находим значение функции в центре отрезка, то есть в точке  $m = \frac{l+h}{2}$ . Если в этой точке функция положительна или равна нулю, то изменяем значение  $h = m$ . Если же в этой точке функция отрицательна, то изменяем значение  $l = m$ . Таким образом отрезок, на котором находится решение был уменьшен в 2 раза. Повторяем эту процедуру до тех пор пока длина отрезка не станет меньше чем  $\epsilon = 10^{-10}$ .

Напишите программу, которая будет решать эту задачу. Функция  $f(x)$  и значения  $l$  и  $h$  должны задаваться в тексте программы.

| $f(x)$ , l, h  | ВЫХОД     |
|--|-----------|
| $f(x) = x^2 - 2$<br>l = 0, h = 2                       | 1.41421   |
| $f(x) = x^2 - 7$<br>l = 0, h = 7                       | 2.64575   |
| $f(x) = x^5 + 2x^4 + 5x^2 + 4x - 500$<br>l = 0, h = 10 | 3.05614   |
| $f(x) = e^x \ln(x) - 7$<br>l = 1, h = 5                | 2.1896095 |

## Задача 10. Размеры типов

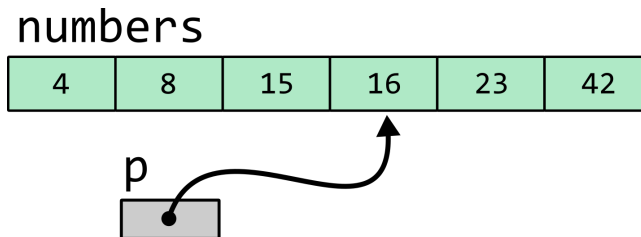
Напишите программу, которая будет печатать размеры следующих типов:

- |         |             |           |
|---------|-------------|-----------|
| • char  | • long long | • double  |
| • short | • size_t    | • int*    |
| • int   | • float     | • int[10] |

## Задача 11. Указатель в массиве

Пусть есть массив и указатель на 4-й элемент этого массива:

```
int numbers[6] = {4, 8, 15, 16, 23, 42};  
int* p = &numbers[3];
```



Чему равны следующие выражения:

- |                            |                          |  |
|----------------------------|--------------------------|--|
| 1. <code>numbers[5]</code> | 5. <code>p[0]</code>     | 9. <code>*(numbers+5)</code>                 |
| 2. <code>*p</code>         | 6. <code>p[1]</code>     | 10. <code>p - numbers</code>                 |
| 3. <code>*(p+1)</code>     | 7. <code>p[-2]</code>    | 11. <code>(short*)p - (short*)numbers</code> |
| 4. <code>*(p-2)</code>     | 8. <code>*numbers</code> | 12. <code>(char*)p - (char*)numbers</code>   |

Решение этой задачи – .txt файл со всеми ответами.

## Задача 12. Куб по указателю

Напишите функцию `cube`, которая будет принимать на вход указатель, содержащий адрес некоторой переменной типа `float`. Функция должна возводить в куб переменную, чей адрес хранит входящий указатель. Вызовите эту функцию из `main` и протестируйте её.

## Задача 13. Умножение массива на 2

Напишите функцию `void mult2_array(int* p, size_t n)`, которая принимает указатель на первый элемент некоторого массива и число `n`, равное размеру этого массива. Вам нужно, используя этот указатель, увеличить все элементы массива в 2 раза.

## Задача 14. Квадратное уравнение

Напишите функцию `int solve_quadratic(double a, double b, double c, double* px1, double* px2)`, которая должна решать квадратное уравнение с коэффициентами `a`, `b` и `c`. Результат функция должна записывать по адресам `px1` и `px2`. Функция должна возвращать:

- 0 - если корней нет. По адресам `px1` и `px2` ничего записывать в этом случае не надо.
- 1 - если есть один корень. Его нужно записать по адресу `px1`.
- 2 - если есть два корня. Их нужно записать по адресам `px1` и `px2`.

Все сравнения делать с точностью  $\epsilon = 10^{-10}$ .