Семинар #4: Типы данных.

Часть 1: Целочисленные типы данных.

Различные целочисленные типы языка C представлены в следующей таблице:

тип	размер (байт)	диапазон значений ($2^{\#bits}$)	спецификатор
char	1	от -128 до 127	%hhi
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	%hhu
unsigned short	2	от 0 до 65535	%hu
unsigned int	4	примерно от 0 до 4-х миллиардов	%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	4	примерно от 0 до 4-х миллиардов	%zu
10-тичная система	-	-	%d
8-ричная система	-	-	%o
16-ричная система	-	-	%x

Это наиболее распространённые значения размеров типов для 64-х битных систем, но на некоторых системах эти значения могут быть другими. Чтобы узнать эти значения можно использовать оператор `sizeof`.

Часть 2: Новое имя типа

В языке C существует возможность вводить новое имя для уже существующего типа с помощью ключевого слова `typedef`. Чтобы ввести новое имя для типа нужно написать следующее:

```
typedef староеимя новоеимя;
```

После этого для типа можно будет использовать и новое и старое имя.

Тип `size_t`

Тип `size_t` – это беззнаковый тип целых чисел, который выбирается таким образом, чтобы он вмещал размер любого массива. Внутри стандартной библиотеки может быть написано примерно следующее:

```
typedef unsigned long long size_t;
```

Но размер типа `size_t` может различаться в зависимости от вычислительно системы. Выясните чему он равен на вашей системе.

Часть 3: Числа с плавающей точкой. Библиотека `math.h`

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	≥ 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

Библиотека `math.h`

В библиотеке `math.h` содержатся множество полезных математических функций.

функция	что делает
`sqrt`	Вычисляет корень числа
`abs`	Вычисляет модуль целого числа
`fabs`	Вычисляет модуль числа с плавающей точкой
`exp`	Экспонента e^x
`log`	Натуральный логарифм $ln(x)$
`sin`, `cos`, `tan`	Синус, косинус и тангенс (радианы)
`asin`, `acos`, `atan`	Арксинус, арккосинус и арктангенс
`floor`	Округление до ближайшего меньшего целого числа
`ceil`	Округление до ближайшего большего целого числа
`pow(x, y)`	Возведение числа в `x` степень `y`

Точность чисел с плавающей точкой

Количество вещественных чисел на любом отрезке бесконечно, а количество возможных значений чисел с плавающей точкой ограничено, поэтому не каждое вещественное число можно закодировать числом `float` или `double`. Это означает, что числа с плавающей точкой всегда вычисляются с погрешностью. Поэтому сравнивать 2 таких числа оператором сравнения `==` очень опасно. Например, следующая программа напечатает `No`.

```c
#include <stdio.h>
int main()
{
    float a = 3 * 0.1;
    float b = 0.3;
    if (a == b)
        printf("Yes\n");
    else
        printf("No\n");
}
```

Такие числа всегда нужно сравнивать с некоторой точностью ϵ по формуле $|a - b| < \epsilon$. Вот так:

```c
#include <stdio.h>
#include <math.h>
int main()
{
    float eps = 1e-5;
    float a = 3 * 0.1;
    float b = 0.3;
    if (fabs(a - b) < eps)
        printf("Yes\n");
    else
        printf("No\n");
}
```

Часть 4: Указатели.

Указатель – это специальная переменная, которая хранит адреса.

```c
#include <stdio.h>
int main() {
    // Предположим у нас есть переменная:
    int x = 42;
    // Положение этой переменной в памяти характеризуется двумя числами - адресом и
    // размером переменной. Узнать их можно, используя операторы & и sizeof:
    printf("Size and address of x = %d and %llu\n",  sizeof(x), &x);
    // Обратите внимание, что для отображения адреса использовался модификатор %llu, так
    // как в 64 битных системах адрес это 64 битное число. Также можно было бы
    // использовать модификатор %p, для отображения адреса в 16 - ричном виде.

    // Для работы с адресами в языке C вводится специальный тип, который называется
    // указатель. Введём переменную для хранения адреса переменной x:
    int* address_of_x = &x;
    // Теперь в переменной address_of_x типа int* будет храниться число - адрес
    // переменной x. Если бы x был бы не int, а float, то для хранения адреса x нужно было
    // бы использовать тип float* .

    // Ну хорошо, у нас есть переменная, которая хранит адрес x. Как её дальше
    // использовать? Очень просто - поставьте звёздочку перед адресом, чтобы получить
    // переменную x.
    // *address_of_x это то же самое, что и  x
    *address_of_x += 10;
    printf("%d\n", x);
    // Запомните: &  -  по переменной получить адрес
    //            *  -  по адресу получить переменную
}
```

Задача

Пусть есть такой код:

```c
#include <stdio.h>
int main()
{
    float x = 10;
}
```

Создайте переменную `p` типа `float*` и сохраните в ней адрес переменной `x`. Используйте переменную `p`, чтобы изменить переменную `x`.