

Семинар #2: Часть 2: Типы данных.

Часть 1: Целочисленные типы данных.

В данной таблице приведены стандартные размеры целочисленных типов. Важно помнить, что за исключением типов `char` и `unsigned char` (размер которых гарантированно равен 1 байту), размеры остальных типов могут варьироваться в зависимости от архитектуры системы и компилятора.

тип	размер (байт)	диапазон значений ($2^{\#bits}$)	спецификатор
<code>char</code>	1	от -128 до 127	<code>%hi</code>
<code>short</code>	2	от -32768 до 32767	<code>%hi</code>
<code>int</code>	4	примерно от -2-х миллиардов до 2-х миллиардов	<code>%i</code>
<code>long</code>	4	примерно от -2-х миллиардов до 2-х миллиардов	<code>%li</code>
<code>long long</code>	8	примерно от -10^{19} до 10^{19}	<code>%lli</code>
<code>unsigned char</code>	1	от 0 до 255	<code>%hhu</code>
<code>unsigned short</code>	2	от 0 до 65535	<code>%hu</code>
<code>unsigned int</code>	4	от 0 до $2^{32} \approx 4 * 10^9$	<code>%u</code>
<code>unsigned long</code>	4	от 0 до $2^{32} \approx 4 * 10^9$	<code>%lu</code>
<code>unsigned long long</code>	8	от 0 до $2^{64} \approx 2 * 10^{19}$	<code>%llu</code>
<code>size_t</code>	8	от 0 до $2^{64} \approx 2 * 10^{19}$	<code>%zu</code>

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

```
int a = 10;
printf("%zu\n", sizeof(a));    // скорее всего напечатает 4
printf("%zu\n", sizeof(int));  // скорее всего напечатает 4
```

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 вычислительной 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
<code>float</code>	4	6	от -38 до 38	<code>%f</code>
<code>double</code>	8	15	от -308 до 308	<code>%lf</code>
<code>long double</code>	от 8 до 16	≥ 15	не хуже чем у <code>double</code>	<code>%Lf</code>
печатать только 3-х чисел после запятой	-	-	-	<code>%.3f</code>
печатать без нулей на конце	-	-	-	<code>%g</code>
печатать в научной записи	-	-	-	<code>%e</code>

Библиотека `math.h`

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

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

```
#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:

```
#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");
}
```