Основы программирования

Синтаксис языка C#

Типы данных

Программа – это набор действий по обработке некоторых данных.

Все данные хранятся в оперативной памяти, связанной с программой.

При описании данных необходимо указывать:

- каким образом они представляются в памяти,
- сколько памяти занимают,
- каков допустимый диапазон значений,
- какие действия над ними можно выполнять.

Эту информацию содержит описание типа, которому принадлежат данные. Поэтому в программе не может быть величин, не принадлежащих какому-либо типу данных.

- C# строго типизированный язык;
- переменные, выражения обязательно должны иметь тип;
- контроль типов осуществляет компилятор;
- контроль типов способствует предотвращению ошибок и повышает надежность программ.

Типы данных

- внутреннее представление данных, а следовательно, и множество их допустимых значений
- допустимые действия над данными (операции и функции)

Классификации типов данных

- *встроенные или стандартные* (относятся к ядру языка программирования) и *пользовательские* (определяются программистом, как правило, на основе стандартных)
- *значащие* (хранятся в статической памяти) и *ссылочные* (в динамической памяти)
- *статические* (определяются на этапе их описания) и *динамические* (определяются в процессе работы программы)
- *простые* и *структурированные* (агрегируют группу данных простых и структурированных типов)

Где хранятся данные

- значащие типы хранятся в статической памяти (в стэке `[stack]`)

  высокое быстродействие, память выделяется на этапе компиляции для потока, ограниченный объем

- ссылочные типы хранятся в динамической памяти (в куче `[heap]`)

  более медленная, память выделяется на этапе выполнения, объем существенно больше

Система типов C#

- типы значений (простые типы)

  - содержат данные.

- ссылочные типы

  - хранят ссылки на фактические данные.
  Также называются объектами.

- типы указателей (не рассматриваем)

Система типов C#. Типы значений

Категория		Описание
Типы значений	Простые типы	Целые со знаком: sbyte, short, int, long
		Целые без знака: byte, ushort, uint, ulong
		Символы Юникода: char
		IEEE с плавающей запятой: float, double
		Десятичный с повышенной точностью: decimal
		Логический: bool
	Перечисляемые типы	Пользовательские типы вида enum E {...}
	Типы структуры	Пользовательские типы вида struct S {...}
	Обнуляемые типы	Расширения любых других типов значений, включающие значение null

Система типов C#. Ссылочные типы

Категория		Описание
Ссылочные типы	Типы классов	Первичный базовый класс для всех типов: `object`
		Строки Юникода: `string`
		Пользовательские типы вида `class C {...}`
	Типы интерфейса	Пользовательские типы вида `interface I {...}`
	Типы массивов	Одно- и многомерные, например `int[]` и `int[,]`
	Типы делегатов	Пользовательские типы, например вида `delegate int D(...)`

Внутреннее представление

- Целый тип
  - целое число в двоичном коде

- Вещественный тип
  - две части – мантисса и порядок, каждое со знаком. Длина мантиссы определяет точность числа, порядок – его диапазон

Целые типы данных

Тип данных (ключевое слово)	Описание	Стандартный класс библиотеки .NET	Занимаемый объем памяти, байт	Диапазон возможных значений
sbyte	Байт со знаком	SByte	1	-128..127
byte	Байт без знака	Byte	1	0..255
short	Короткое целое со знаком	Int16	2	-32768..32767
ushort	Короткое целое без знака	UInt16	2	0..65535
int	Целое со знаком (или просто целое)	Int32	4	$-2*10^9..2*10^9$
uint	Целое без знака	UInt32	4	$0..4*10^9$
long	Длинное целое со знаком	Int64	8	$-9*10^{18}..9*10^{18}$
ulong	Длинное целое без знака	UInt64	8	$0..18*10^{18}$
char	Символьный (код Unicode-символа)	Char	2	$0000..FFFF_{16}$

Вещественные типы данных

Тип данных (ключевое слово)	Описание	Стандартный класс библиотеки .NET	Занимаемый объем памяти, байт	Диапазон возможных значений
Вещественные типы данных				
float	Вещественное число	Single	4	1.5E-45...3.14E+38
double	Вещественное число с двойной точностью	Double	8	5E-324…1.7E+308

Различия double или decimal

- для обычных дробных чисел можно взять тип `float`, для очень больших дробных чисел - тип `double`.
- несмотря на большую разрядность типа `decimal` по сравнению с типом `double`, тип `double` может хранить большее значение. Однако значение `decimal` может содержать до 28 знаков после запятой, тогда как значение типа `double` - всего 15-16 знаков после запятой.
- `decimal` чаще находит применение в финансовых вычислениях, тогда как `double` - в математических операциях.

Различия double или decimal

Общие различия между этими двумя типами можно выразить следующей таблицей:

	Decimal	Double
Наибольшее значение	$\sim 10^{28}$	$\sim 10^{308}$
Наименьшее значение (без учета нуля)	10^{-28}	$\sim 10^{-323}$
Знаков после запятой	28	15-16
Разрядность	16 байт	8 байт
Операций в секунду	сотни миллионов	миллиарды

Символьный тип данных

char	Символьный (код Unicode-символа)	Char	2	$0000..FFFF_{16}$

```
char ch;
ch = 10;  // Ошибка, это работать не будет.
```

Логический тип данных

Тип данных (ключевое слово)	Описание	Стандартный класс библиотеки .NET	Занимаемый объем памяти, байт	Диапазон возможных значений
bool	Логический тип	Boolean	1	true, false

Строковый тип данных

Тип данных (ключевое слово)	Описание	Стандартный класс библиотеки .NET	Занимаемый объем памяти, байт	Диапазон возможных значений
string	Строка	String	Длина не ограничена	Состоит из символов

Другие типы данных

Тип данных (ключевое слово)	Описание	Стандартный класс библиотеки .NET	Занимаемый объем памяти, байт	Диапазон возможных значений
object	Базовый объектный тип	Object		
enum	Перечисление			
struct	Структура			

Использование суффиксов

все вещественные литералы рассматриваются как значения типа `double`.

Чтобы явно указать, что литерал представляет тип `float` или тип `decimal`, необходимо к литералу добавлять суффикс: `F/f` - для `float` и `M/m` - для `decimal`.

```
1  float a = 3.14F;
2  float b = 30.6f;
3
4  decimal c = 1005.8M;
5  decimal d = 334.8m;
```

Использование суффиксов

Все целочисленные литералы рассматриваются как значения типа `int`.

Чтобы явным образом указать, что целочисленный литерал представляет значение типа `uint`, надо использовать суффикс `U/u`, для типа `long` - суффикс `L/l`, а для типа `ulong` - суффикс `UL/ul`:

```
1  uint a = 10U;
2  long b = 20L;
3  ulong c = 30UL;
```

Присваивание значений

При присваивании величины значащего типа копируются сами данные, а при присваивании величины ссылочного типа копируется ссылка на данные.

При сравнении величин значащего типа сравниваются хранимые данные, то есть величины равны, если они хранят одинаковые данные.

При сравнении величин ссылочного типа сравниваются ссылки на данные, то есть величины равны, если они ссылаются на одни и те же данные.

Значащие и ссылочные типы данных

Величины `a` и `b` являются значимыми и равны между собой, поскольку хранят одинаковые значения.

Величины `c`, `d` и `e` относятся к ссылочному типу. Величина `d` равна величине `e` -- они ссылаются на одни и те же данные, а `c` и `e` не равны, так как ссылаются на разные данные, даже несмотря на то. что содержимое этих данных одинаково.