Числовые типы данных C++

Целые			С плавающей запятой (вещественные)
Знаковые	Беззнаковые		
`signed char`	`unsigned char`		`float`
`short int`	`unsigned short int`		`double`
`int`	`unsigned int`		`long double`
`long int`	`unsigned long`		
`long long int`	`unsigned long long int`		

Типичные характеристики числовых типов C++ и их аналоги в Delphi 7

Тип C++	Диапазон[1, 2]	Размер	Аналог в Delphi 7 (32 бита)	
			Знаковый	Беззнаковый
`char`	±127	1 байт (8 бит)	`SmallInt`	`Char, Byte`
`short int`	±32 767	2 байта (16 бит)	`ShortInt`	`Word`
`int`	±2 147 483 647	4 байта (32 бита)	`Integer`	`Cardinal`
`long int`				
`long long int`	±9 223 372 036 854 775 807	8 байт (64 бита)	`LongInt`	`LongWord`
`float`	Зависит от точности значения, формат по IEEE 754.	4 байта	`Single`	
`double`		8 байт	`Real`	
`long double`		12 байт (96 бит)	`Extended`	

[1] Для знаковых типов верхняя граница на 1 меньше (по модулю), чем нижняя, например, для `char` — от −128 до +127. В таблице для краткости это не учтено.

[2] Для беззнаковых типов диапазон от 0 до удвоенной границы беззнакового типа, минус 1, например, для `unsigned char` — от 0 до +255.

Синтаксис оператора **if**… **else**:

```
if (выражение-условие-1)
      оператор-1,-выполняющийся,-если-условие-1-истинно
else if (выражение-условие-2)
      оператор-2,-выполняющийся,-если-условие-2-истинно
else
      оператор-n,-выполняющийся,-если-все-условия-ложны
```

Синтаксис оператора **switch**:

```
switch (выражение) {
case вариант-1:
      операторы-варианта-1
case вариант-2:
      операторы-варианта-2
…
case вариант-n:
      операторы-варианта-n
default:
      операторы-по-умолчанию
}
```

Синтаксис конструкций циклов:

- **while** (*выражение-условие-продолжения*) *оператор-тело-цикла*
- **do** *оператор-тело-цикла* **while** (*выражение-условие-продолжения*);
- **for** (*оператор-инициализации*; *выражение-условие-продолжения*; *выражение*)
  оператор-тело-цикла

Синтаксис тернарного (условного) выражения:

условие ? *выражение-1* : *выражение-2*.

Синтаксис определения функции:

```
тип-возвращаемого-значения имя-функции (
      тип-аргумента-1 аргумент-1, …, тип-аргумента-N аргумент-N) {
      тело-функции
}
```

Листинг 1 - Программа, печатающая на экране строку «Hello, world!»

```
#include <cstdio> /* Директива препроцессора. */

int main() {
      puts("Hello, world!");
      return 0;
      puts("Вот скачет Неуловимый Джо!");   // Никогда не выполнится.
}
```

Листинг 2 - Функция вычисления целой степени числа

```
double power(const double base, const unsigned int exponent) {
      double result = 1.0;
      for (unsigned int i = 0; i < exponent; ++i) {
            result *= base;
      }
      return result;
}
// …
// Где-либо в программе:
double x = 5.0;
double y = power(x, 3); // y = 125.0
```

Листинг 3 - Объявление и реализация функции

```
// Прототип функции:
double power(const double base, const int exponent);
// Реализация функции (возможно, в другом файле):
double power(const double base, const int exponent) { … }
```

.

Листинг 4 - Перегрузка функций

```
int power(int base, int exponent) { … }
int power(int base, unsigned int exponent) { … }
double power(double base, int exponent) { … }
// double power(int base, int exponent) { … }
// …
power(2, 5);    // int power(int, int)
power(2, 5u);   // int power(int, unsigned int)
power(2.0, 5);  // double power(double, int)
```

Листинг 5 - Функция вычисления целой степени числа, модифицированная

```
1  double power(const double base, const int exponent) {
2        double result = 1.0;
3        if (exponent < 0) {
4              return power(1.0 / base, -exponent);
5        } else {
6              for (unsigned int i = 0; i < exponent; i++)
7                    result *= base;
8        }
9        return result;
```

Листинг 6 - Функция вычисления целой степени числа, рекурсия

```
10 double power(const double base, const int exponent) {
11       if (exponent < 0) {
12             return power(1.0 / base, -exponent);
13       } else if (exponent > 0) {
14             return base * power(base, exponent - 1);
15       }
16       return 1.0;
17 }
```