Таблица 1 — 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`	

Таблица 2 — 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`	

Логический тип — **`bool`** со значениями **`true`** (истина) и **`false`** (ложь).

Информацию о строковом типе **`string`** см. в таблице 3 (с. 2).

Переменные объявляются не в специальной секции, как **`var`** в Pascal, а в (почти) любом месте программы. Если перед именем типа или переменной записано **`const`**, изменить значение нельзя (а с **`constexpr`** оно должно быть известно при компиляции).

```
int negative = -42;
unsigned int count = 66;
double pi = 3.1415926535;
char letter = 'A';
string name = "Dmitry";
bool condition = true;
```

```
constexpr int zero = 0;
const char Z = 'Z';
double const S = 2*pi*pi;
```

Листинг 1 — Примеры объявления и инициализации переменных

[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.

Таблица 3 — Некоторые операции над строковым типом `string`

Операция	**Синтаксис**
Объявление (пустая строка):	`string name;`
Присваивание значения:	`name = "Dmitry";`
Сравнение:	
— равенство:	`name == "Dmitry"  // Не равно: !=`
— словарный порядок:	`name < "Nikolay"  // Можно <=, >, >=`
Определение длины:	`unsigned int length = name.size();`
Доступ к символу:	
— к начальному:	`char first_letter = name[0];`
— к последнему:	`char last_letter = name[name.size()-1];`
Сцепление строк:	`name + "Kozliuk"  // "Dmitry Kozliuk"`
— нельзя:	`"Dmitry" + "Kozliuk" // Ошибка!`
— также нельзя:	`"Dmitry" + ' '`
— однако можно:	`name + ' '`
Удаление символов:	`string text = "minus five";` `text.erase(0, 6);  // name == "five"`
Взятие подстроки:	
— с начала:	`name.substr(0, 6)  // "Dmitry"`
— с конца:	`name.substr(7)  // "Kozliuk"`
— из середины:	`name.substr(7, 1)  // "K"`
Поиск в строке:	
— с начала:	`name.find("i")  // 2,  "Dm i try…"`
— с конца:	`name.rfind("i")  // 11, "…Kozl i uk"`
— после 3-го символа:	`name.find("i", 3)  // 11`
— того, чего в ней нет:	`name.find("S")  // string::npos`

В C++ строки и символы различаются. Символьные константы пишутся в одинарных кавычках (`'A'`), строковые — в двойных (`"A"`), эти выражения не взаимозаменяемы. В таблице 4 указно несколько особых символов (их можно употреблять и в составе строк).

Таблица 4 — Специальные символы C++

Символ	**Значение**
`'\n'`	Переход на следующую строку.
`'\t'`	Табуляция (для выравнивания текста).
`'\\'`	Обратная косая черта (\).

Таблица 5 — Операторы языка C++

Операторы	Аналоги в Pascal	Примечания
`+ - * /`	`+ - * /`	
`%`	**mod**	Остаток от деления.
`>> <<`	**shr shl**	Битовые сдвиги вправо и влево. В C++ эти операторы используются также для ввода-вывода.
`++ --`	`Inc() Dec()`	В C++ имеется префиксная форма (`++i`) и постфиксная (`i++`). В первом случае результат выражения — новое значение `i`, во втором — старое.
`=`	`:=`	Не следует эти операторы путать.
`==`	`=`	
`!=`	`<>`	Не равно.
`> < >= <=`	`> < >= <=`	
`& | ^ ~`	**and or xor not**	*Побитовая* конъюнкция («И»), дизъюнкция («ИЛИ»), сложение по модулю 2 («исключающее ИЛИ») и инверсия («НЕ»). Обязательно см. ниже.
`&& ||`	**and or**	Конъюнкция и дизъюнкция логических значений (например, в условиях **if**, **while** и т. п). Если первый операнд — истина (ложь), второй не вычисляется.
`!`	**not**	Инверсия условия (в **if**, **while** и т. п.): **if** `(!(x == 0))` эквивалентно **if** `(x != 0)`.
`c ? a : b`		Если `c` истинно, результатом выражения становится `a`, иначе `b`, а другая часть не вычисляется. Пример: `sqrt((N > 1) ? sum / (N-1) : 0)` — корень квадратный из `sum/(N-1)`, если `N > 1`, иначе из `0`.
`y = f(x); g(t);`	`y = f(x); g(t);`	Вызов функции `f` с передачей ей аргумента `x` и записью возвращаемого значения в переменную `y`; вызов функции или процедуры `g` с передачей ей аргумента `t` (возвращаемое значение игнорируется).

Таблица 6 — Соответствия управляющих конструкций C++ и Pascal

Конструкция C++	Конструкция Pascal	Примечания
код; { *код* }	код **begin** *код* **end**	Код может быть произвольным или пустым. Любая из указанных конструкций подходит в качестве действия или тела цикла ниже.
if (*условие-1*) *действие-1* **else if** (*условие-2*) *действие-2* **else** *действие-3*	**if** *условие-1* **then** *действие-1* **elseif** *условие-2* **then** *действие-2* **else** *действие-3;*	Предложения **else if** (**elseif** в Pascal) и **else** могут отсутствовать.
while (*условие*) *тело-цикла*	**while** *условие* **do** *тело-цикла;*	В C++ условие считается истинным, если оно равно **true** или это целое число, не равное 0. Проверка x на кратность трём: **if** (x % 3) { }. Бесконечный цикл: **while** (1) { }.
do *тело-цикла* **while** (*условие*);	**repeat** *тело цикла* **until not** *условие;*	В C++ используется условие продолжения цикла, в Pascal — условие прекращения. Точка с запятой в конце обязательна.
for (*инициализация;* *условие;* *действие*) *тело-цикла*	*инициализация;* **while** *условие* **do** **begin** *тело-цикла;* *действие;* **end;**	Инициализация, условие или выражение могут быть опущены, если не нужны (опущенное условие считается всегда истинным). Бесконечный цикл: for (;;) тело-цикла.
for (**int** i = *A*; i < *B*; ++i) *тело-цикла*	**for** I := *A* **to** *B*-1 **do** *тело-цикла;*	Типовое использование цикла **for** для организации счетчика.
break;	Break;	В C++ операторы являются ключевыми словами, поэтому всегда набираются в нижнем регистре.
continue;	Continue;	

Конструкция C++	**Конструкция Pascal**	**Примечания**

switch (*выражение*) {
  case *вариант-1*:
  действия-1
  break;
  case *вариант-2*:
  действия-2
  break;
  case *вариант-3*:
  case *вариант-4*:
  действия-34
  break;
  default:
  действия-по-умолчанию
}
	case выражение **of**
	  вариант-1:
	  действия-1;
	  вариант-2:
	  действия-2;
	  вариант-3, вариант-4:
	  действия-34;
	  else
	  действия-по-умолч.
	end;
		Оператор выбора: если значение *выражения* — *вариант-1*, выполнятся *действия-1*, если *вариант-2* — выполнятся *действия-2* и т. д., иначе — действия по умолчанию. Варианты могут быть целочисленными константами или символами.
		В C++, если не используется оператор **break**, после выполнения действия управление не переходит за конструкцию **switch**, а продолжается вниз до конца (или до **break**). Так работает выбор *действия-34* в случае, если выражение равно *варианту-3* или *варианту-4*. Предпочтительно использовать **break** всегда, кроме подобных случаев. Объявлять переменные внутри **switch** можно только в действиях-блоках (в фигурных скобках).

// текст до конца строки /* текст */	// текст до конца строки { текст } (* текст *)	Комментарий. Однострочный комментарий отсутствует в Pascal (и в C), но присутствует в Delphi (и C++).
cout << "x = " << x; cout << "Строка.\n";	Write('x = ', x); WriteLn('Строка.');	Вывод данных. В строках могут встречаться специальные последовательности (escape sequences), начинающиеся с обратной косой черты (\, backslash): — перевод строки: \n (в Windows предпочтительнее \r\n), — горизонтальная табуляция: \t и другие.
cin >> x >> y; getline(cin, some_string);	Read(x, y); ReadLn(some_string);	Значения разделяются пробелами, табуляциями, переводами строк. Вторая форма — для чтения строки с пробелами.
for (*тип имя* : *вектор*) *тело-цикла*		Проход по всем элементам *вектора* (типа vector<*тип*>).

Таблица 7 — Краткая справка по использованию типа `vector<T>`

Операция	Синтаксис
Объявление вектора чисел:	
— пустого:	`vector<double> numbers;`
— из 10 элементов:	`vector<double> numbers(10);`
— из 20 элементов-троек:	`vector<double> numbers(20, 3);`
— из чисел 1 и 2:	`vector<double> numbers { 1, 2 };`
Размер вектора:	
— получить:	`unsigned int n = numbers.size();`
— изменить (сделать 30):	`numbers.resize(30);`
Доступ к элементам:	
— к начальному:	`double first = numbers[0];` `double first = numbers.front();`
— к пятому (от нулевого):	`double fifth = numbers[5];`
— к последнему:	`double last = numbers.back();` `last = numbers[numbers.size() - 1];`
Сделать 5-й элемент равным 42:	`numbers[5] = 42;`
Примечание: отсчет элементов вектора начинается с нуля, поэтому 5-м элементом называется 6-й по счету (0, 1, 2, 3, 4, **5**).	
Добавить элемент:	
— в конец:	`numbers.push_back(42);`
— в начало:	`numbers.push_front(42);`
Удалить элемент:	
— с начала:	`numbers.pop_front();`
— с конца:	`numbers.pop_back();`

```
1       #include "sdt.h"
2
3       int main()
4       {
5           // Вывод строки на экран.
6           cout << "Hello, world!\n";
7       }
```

Листинг 2 — Элементарная программа на C++

```
#include "sdt.h"

int main()
{
    vector<double> xs;

    int n;
    cout << "Enter sample size: ";
    cin >> n;

    cout << "Enter " << n << " sample values: ";
    xs.resize(n);
    for (int i = 0; i < n; ++i) {
        cin >> xs[i];
    }

    double mean = 0;
    for (double x : xs) {
        mean += x;
    }
    mean /= xs.size();
    cout << "Mean is " << mean << ".\n";

    double variance = 0;
    if (n > 1) {
        for (double x : xs) {
            double d = x - mean;
            variance += d*d;
        }
        variance /= n - 1;
    }
    cout << "Variance is " << variance << ".\n";
}
```

Листинг 3 — Программа для вычисления статистических оценок

```
#include "sdt.h"

int main()
{
     cout << "Enter your name and age: ";
     string name;
     unsigned int age;
     cin >> name >> age;                    // Shepard 34
     const unsigned int next = age + 1;
     cout << "Hello, " << name << ", next year "
          << "you will be " << next << " years old.\n";
}
```

Листинг 4 — Ввод и вывод данных