

Таблица 1 — Числовые типы данных C++

Целые		С плавающей запятой (вещественные)
Знаковые	Беззнаковые	
<code>char</code>	<code>unsigned char</code>	
<code>short int</code>	<code>unsigned short int</code>	
<code>int</code>	<code>unsigned int</code>	
<code>long int</code>	<code>unsigned long</code>	
<code>long long int</code>	<code>unsigned long long int</code>	
		<code>float</code>
		<code>double</code>
		<code>long double</code>

Тип `size_t` — беззнаковое целое, предназначен для индексов, размеров и т. п.

Таблица 2 — Типичные характеристики числовых типов C++ и их аналоги в Delphi 7

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

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. Для беззнаковых типов диапазон от 0 до удвоенной границы знакового 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 подстроки в строке (результат — номер символа от начала строки):	
— 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"). В таблице 4 приведено несколько особых символов (их можно употреблять и в составе строк).

Таблица 4 — Специальные символы C++

Символ	Значение	Символ	Значение
\n	Переход на следующую строку.	\'	Одинарная кавычка (').
\\	Обратная косая черта (\).	\"	Двойная кавычка (").
\t	Табуляция — переход к следующей позиции, кратной 8 (для выравнивания).		

Таблица 5 — Операторы языка C++

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

Операторы сокращенного присваивания: `+=`, `-=`, `*=`, `/=`, `%=`, `&=`, `|=`, `^=`, `&&=` и `||=` имеют смысл: `x = x + a` \Leftrightarrow `x += a` и т. д.

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. В C++ условие считается истинным, если оно равно true или это целое число, не равное 0.
while (условие) тело-цикла	while условие do тело-цикла;	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
<pre>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 }</pre>	<pre>case выражение of вариант-1: действия-1; вариант-2: действия-2; вариант-3, вариант-4: действия-34; else действия-по-умолч. end;</pre>	<p>Оператор выбора: если значение <i>выражения</i> — <i>вариант-1</i>, выполняются <i>действия-1</i>, если <i>вариант-2</i> — выполняются <i>действия-2</i> и т. д., иначе — действия по умолчанию. Варианты могут быть целочисленными константами или символами.</p> <p>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 <i>действия-34</i> в случае, если выражение равно <i>варианту-3</i> или <i>варианту-4</i>. Предпочтительно использовать break всегда, кроме подобных случаев. Объявлять переменные внутри switch можно только в действиях-блоках (в фигурных скобках).</p>
<pre>// текст до конца строки /* текст */</pre>	<pre>// текст до конца строки { текст } (* текст *)</pre>	Комментарий. Однострочный комментарий отсутствует в Pascal (и в C), но присутствует в Delphi (и C++).
<pre>cout << "x = " << x; cout << "Строка.\n";</pre>	<pre>Write('x = ', x); WriteLn('Строка.');</pre>	Вывод данных. В строках могут встречаться специальные последовательности (escape sequences), начинающиеся с обратной косой черты (\, backslash), см. таблицу 4.
<pre>cin >> x >> y; getline(cin, some_string);</pre>	<pre>Read(x, y); ReadLn(some_string);</pre>	Значения разделяются пробелами, табуляциями, переводами строк. Второй способ — для чтения строки с пробелами.
<pre>for (тип имя : вектор) тело-цикла</pre>		Проход по всем элементам <i>вектора</i> (типа <code>vector<тип></code>).

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

Операция	Синтаксис
Объявление вектора чисел:	
— пустого:	<code>vector<double> numbers;</code>
— из 10 элементов:	<code>vector<double> numbers(10);</code>
— из 20 элементов-троек:	<code>vector<double> numbers(20, 3);</code>
— из чисел 1 и 2:	<code>vector<double> numbers { 1, 2 };</code>
Размер вектора:	
— получить:	<code>unsigned int n = numbers.size();</code>
— изменить (сделать 30):	<code>numbers.resize(30);</code>
Доступ к элементам:	
— к начальному:	<code>double first = numbers[0];</code> <code>double first = numbers.front();</code>
— к пятому (от нулевого):	<code>double fifth = numbers[5];</code>
— к последнему:	<code>double last = numbers.back();</code> <code>last = numbers[numbers.size() - 1];</code>
Сделать 5-й элемент равным 42:	<code>numbers[5] = 42;</code>
<i>Примечание:</i> отсчет элементов вектора начинается с нуля, поэтому 5-м элементом называется 6-й по счету (0, 1, 2, 3, 4, 5). Элемент с индексом 0 называют или нулевым, или начальным, чтобы не путать с первым (с индексом 1, вторым по счету).	
Добавить элемент в конец:	<code>numbers.push_back(42);</code>
Удалить последний элемент:	<code>numbers.pop_back();</code>

```

1      #include <iostream>
2
3      using namespace std;
4
5      int main()
6      {
7          // Вывод строки на экран.
8          cout << "Hello, world!\n";
9      }
```

Листинг 2 — Элементарная программа на C++

В последующих листингах строки 1—3 опущены для краткости, но они нужны.

```
1      #include <vector>
2
3      int main()
4      {
5          vector<double> xs;
6
7          int n;
8          cout << "Enter sample size: ";
9          cin >> n;
10
11         cout << "Enter " << n << " sample values: ";
12         xs.resize(n);
13         for (int i = 0; i < n; ++i) {
14             cin >> xs[i];
15         }
16
17         double mean = 0;
18         for (double x : xs) {
19             mean += x;
20         }
21         mean /= xs.size();
22         cout << "Mean is " << mean << ".\n";
23
24         double variance = 0;
25         if (n > 1) {
26             for (double x : xs) {
27                 double d = x - mean;
28                 variance += d*d;
29             }
30             variance /= n - 1;
31         }
32         cout << "Variance is " << variance << ".\n";
33     }
```

Листинг 3 — Программа для вычисления статистических оценок

```
1      int main()
2      {
3          cout << "Enter your name and age: ";
4          string name;
5          unsigned int age;
6          cin >> name >> age; // Shepard 34
7          const unsigned int next = age + 1;
8          cout << "Hello, " << name << ", next year "
9              << "you will be " << next << " years old.\n";
10     }
```

Листинг 4 — Ввод и вывод данных