C++ Standard

<https://isocpp.org/>

<http://www.open-std.org/jtc1/sc22/wg21/>

<http://www.open-std.org/JTC1/SC22/WG21/docs/papers/2015/n4567.pdf>

C++ Keywords

alignas (since C++11)

alignof (since C++11)

and

and\_eq

asm

auto(1)

bitand

bitor

bool

break

case

catch

char

char16\_t (since C++11)

char32\_t (since C++11)

class

compl

concept (concepts TS)

const

constexpr (since C++11)

const\_cast

continue

decltype (since C++11)

default(1)

delete(1)

do

double

dynamic\_cast

else

enum

explicit

export(1)

extern

false

float

for

friend

goto

if

inline

int

long

mutable

namespace

new

noexcept (since C++11)

not

not\_eq

nullptr (since C++11)

operator

or

or\_eq

private

protected

public

register

reinterpret\_cast

requires (concepts TS)

return

short

signed

sizeof

static

static\_assert (since C++11)

static\_cast

struct

switch

template

this

thread\_local (since C++11)

throw

true

try

typedef

typeid

typename

union

unsigned

using(1)

virtual

void

volatile

wchar\_t

while

xor

xor\_eq

(1) - meaning changed in C++11

In addition to keywords, there are two identifiers with special meaning, which may be used as names of objects or functions, but have special meaning in certain contexts.

override (C++11)

final (C++11)

Also, all identifiers that contain a double underscore \_\_ in any position and each identifier that begins with an underscore followed by an uppercase letter is always reserved and all identifiers that begin with an underscore are reserved for use as names in the global namespace. See identifiers for more details.

* the identifiers that are keywords cannot be used for other purposes;
* the identifiers with a double underscore anywhere are reserved;
* the identifiers that begin with an underscore followed by an uppercase letter are reserved;
* the identifiers that begin with an underscore are reserved for use at global namespace.

1. <http://en.cppreference.com/w/cpp/keyword>

2. <http://en.cppreference.com/w/cpp/language/identifiers>

C++ Preprocessor Directives

#if

#elif

#else

#endif

#defined

#ifdef

#ifndef

#define

#undef

#include

#line

#error

#pragma

C++ Operators, Precedence and Associativity

|  |  |  |  |
| --- | --- | --- | --- |
| **Precedence** | **Operator** | **Description** | **Associativity** |
| **1** | :: | Scope resolution | Left-to-right |
| **2** | ++ | Suffix/postfix increment | Left-to-right |
| -- | Suffix/postfix decrement |
| *type*()   *type*{} | Functin-style type cast |
| () | Function call |
| [] | Array subscript |
| . | Member access (by reference) |
| -> | Member access (through pointer) |
| typeid() | Run-time type information |
| const\_cast,  dynamic\_cast,  reinterpret\_cast,  static\_cast | Type cast |
|  |
|  |
|  |
| **3** | ++ | Prefix increment | Right-to-left |
| -- | Prefix decrement |
| + | Unary plus |
| - | Unary minus |
| ! | Logical NOT |
| ~ | Bitwise NOT |
| (*type*) | C-style cast |
| \* | Indirection (dereference) |
| & | Address-of |
| sizeof | Size-of |
| new   new[] | Dynamic memory allocation |
| delete   delete[] | Dynamic memory deallocation |
| **4** | .\* | Pointer-to-member | Left-to-right |
| ->\* | Pointer-to-member |
| **5** | \* | Multiplication | Left-to-right |
| / | Division |
| % | Remainder (modulo) |
| **6** | + | Addition | Left-to-right |
| - | Subtraction |
| **7** | << | Bitwise left shift | Left-to-right |
| >> | Bitwise right shift |
| **8** | < | Less than | Left-to-right |
| <= | Less than or equal to |
| > | Greater than |
| >= | Greater than or equal to |
| **9** | == | Equal to | Left-to-right |
| != | Not equal to |
| **10** | & | Bitwise AND | Left-to-right |
| **11** | ^ | Bitwise XOR (exclusive or) | Left-to-right |
| **12** | | | Bitwise OR (inclusive or) | Left-to-right |
| **13** | && | Logical AND | Left-to-right |
| **14** | || | Logical OR | Left-to-right |
| **15** | ?: | Ternary conditional | Right-to-left |
| **16** | = | Direct assignment (provided by default for C++ classes) | Right-to-left |
| += | Assignment by sum |
| -= | Assignment by difference |
| \*= | Assignment by product |
| /= | Assignment by quotient |
| %= | Assignment by remainder |
| <<= | Assignment by bitwise left shift |
| >>= | Assignment by bitwise right shift |
| &= | Assignment by bitwise AND |
| ^= | Assignment by bitwise XOR |
| |= | Assignment by bitwise OR |
| **17** | throw | throw operator (exceptions throwing) | Right-to-left |
| **18** | , | Comma | Left-to-right |

**8, 9** Relational operators.

<http://acm2014.cct.lsu.edu/localdoc/cppreference/en/cpp/language/operator_precedence.html>

<http://en.cppreference.com/w/cpp/language/operator_precedence>

<https://en.wikipedia.org/wiki/Operators_in_C_and_C%2B%2B>

<http://cpansearch.perl.org/src/KAZUHO/cppref-0.07/orig/operator_precedence.html>

<http://www.learncpp.com/cpp-tutorial/31-precedence-and-associativity/>

Escape Sequences

**Escape sequence Description**

**\'** single quote

**\"** double quote

**\?** question mark

**\\** backslash

**\a** audible bell

**\b** backspace

**\f** form feed - new page

**\n** line feed - new line

**\r** carriage return

**\t** horizontal tab

**\v** vertical tab

**\nnn** arbitrary octal value

**\xnn** arbitrary hexadecimal value

**\unnnn** universal character name

(arbitrary [Unicode](http://en.wikipedia.com/wiki/Unicode) value);

may result in several characters

**\Unnnnnnnn** universal character name  
(arbitrary [Unicode](http://en.wikipedia.com/wiki/Unicode) value);  
may result in several characters

**\0** null character

C++ Digraphs and trigraphs

|  |  |  |
| --- | --- | --- |
| **Primary** | **Digraph** | **Trigraph** (until C++17) |
| { | <% | ??< |
| } | %> | ??> |
| [ | <: | ??( |
| ] | :> | ??) |
| # | %: | ??= |
| \ |  | ??/ |
| ^ |  | ??' |
| | |  | ??! |
| ~ |  | ??- |

C++ Types

void

bool

char

signed char

unsigned char

wchar\_t

char16\_t

char32\_t

short

short int

signed short

signed short int

unsigned short

unsigned short int

int

signed

signed int

unsigned

unsigned int

long

long int

signed long

signed long int

unsigned long

unsigned long int

long long

long long int

signed long long

signed long long int

unsigned long long

unsigned long long int

float

double

long double

std::string

C++ Versions

C++98

C++03

C++11

C++14

C++17

Bu versiyonlar arasındaki farklar.

Kötü kodları bulurken hangi versiyonun kullanılacağı belirlenmeli.

Balanced Parentheses

**;** (semicolon), **::** (scope operator), **{ }** (braces), **( )** (parentheses), **[ ]** (brackets) C++’da kullanıldığı yerler

//. (member operator) kullanılan yerler

class A

**{**

public**:**

int a**;**

**};**

int main**()**

**{**

A obj **=** **new** A**();**

obj**.**a **=** 1**;**

**return** 0**;**

**}**

// [ ] kullanımı

int array1**[**5**];**

int array2**[**5**]** **=** **{** 1**,** 2**,** 3**,** 4**,** 5 **};**

int array3**[]** **=** **{** 1**,** 2**,** 3**,** 4**,** 5 **};**

int array4**[]** **{** 1**,** 2**,** 3**,** 4**,** 5 **};**

int size **=** 5**;**

int**\*** array5 **=** **new** int**[**size**];**

// initialization

int a1 **=** 1**;**

int a2**(**1**);**

int a3**{**1**};**

int a4 **=** **{**1**};**

; (semicolon, statement terminator) kullanılan yerler

int i**;**

a **=** b **+** 1**;**

DoSomething**();**

**for(**int i **=** 0**;** i **<** length**;** i**++)**

**{**

**}**

**break;**

**continue;**

**return** 0**;**

**goto** label**;**

class A

**{**

void f**();**

**}**

class B

**{**

**};**

**namespace** a**{**

**namespace** b**{**

**namespace** c**{**

**namespace** d**{**

**namespace** e**{**

**namespace** f**{**

**}**

**}**

**}**

**}**

**}**

**}**

**namespace** abcdef **=** a**::**b**::**c**::**d**::**e**::**f**;**

**do**

**{**

// statement

**}while(** expression **);**

asm **(**"assembly code"**);**

**using** **namespace** std**;**

**using** std**::**cout**;**

cout **<<** "Hello World!"**;**

//preprocessing\_op\_or\_punc

ANTLRWorks

<http://tunnelvisionlabs.com/products/demo/antlrworks>

<http://www.antlr3.org/works/>

<https://github.com/antlr/antlrworks>

Static program analysis

<https://en.wikipedia.org/wiki/Static_program_analysis>

<https://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis>

C++ Identifiers

Object or variable name

Class, structure, or union name

Enumerated type name

Member of a class, structure, union, or enumeration

Function or class-member function

typedef name

Label name

Macro name

Macro parameter

Array

Type, type member, template, namespace

Identifiers are names for language entities.

An identifier is an arbitrary long sequence of digits, underscores, lowercase and uppercase Latin letters, and most Unicode characters (disallowed are control characters and characters in the basic source character set). A valid identifier must begin with a non-digit character (Latin letter, underscore, or Unicode non-digit character). Identifiers are case-sensitive (lowercase and uppercase letters are distinct), and every character is significant.

The following characters are allowed as any character of an identifier:

\_ a b c d e f g h i j k l m

n o p q r s t u v w x y z

A B C D E F G H I J K L M

N O P Q R S T U V W X Y Z

The following characters are allowed as any character in an identifier except the first:

0 1 2 3 4 5 6 7 8 9

C++ Statements

<http://en.cppreference.com/w/cpp/language/statements>

<https://msdn.microsoft.com/en-us/library/bzzyh1y4.aspx>

<http://www.cplusplus.com/doc/tutorial/control/>

<https://en.wikipedia.org/wiki/Statement_%28computer_science%29>

1) expression statement

2) compound statement

3) selection statement

4) iteration statement

5) jump statement

6) declaration statement

7) try block

Compiler

<https://en.wikipedia.org/wiki/Compiler>

Lexer

<https://en.wikipedia.org/wiki/Lexical_analysis>

Parser (syntactic analysis)

<https://en.wikipedia.org/wiki/Parsing#Computer_languages>

Preprocess

<https://en.wikipedia.org/wiki/Preprocessing>

Semantic analysis

Code generation

Code optimization

AST (Abstract syntax tree)

<https://en.wikipedia.org/wiki/Abstract_syntax_tree>

Parse Tree

<https://en.wikipedia.org/wiki/Parse_tree>

Parser Generator

<https://en.wikipedia.org/wiki/Compiler-compiler>

Online C++ IDE

c++ shell <http://cpp.sh/>

ideone <https://ideone.com/>

<https://www.codechef.com/ide>

<http://codepad.org/>

<http://www.tutorialspoint.com/codingground.htm>

Online Repository

[*https://bitbucket.org*](https://bitbucket.org)

[*https://github.com/*](https://github.com/)

[*http://sourceforge.net/*](http://sourceforge.net/)

[*https://www.assembla.com*](https://www.assembla.com)

[*https://code.google.com*](https://code.google.com)

[*http://www.codeplex.com/*](http://www.codeplex.com/)

[*http://projectlocker.com/*](http://projectlocker.com/)

[*https://www.codebasehq.com/*](https://www.codebasehq.com/)

[*https://www.visualstudio.com/en-us/features/version-control-vs.aspx*](https://www.visualstudio.com/en-us/features/version-control-vs.aspx)

The Definitive C++ Book Guide and List

<http://stackoverflow.com/questions/388242/the-definitive-c-book-guide-and-list?rq=1>