Tokens, Expression & Control Structures

Tokens

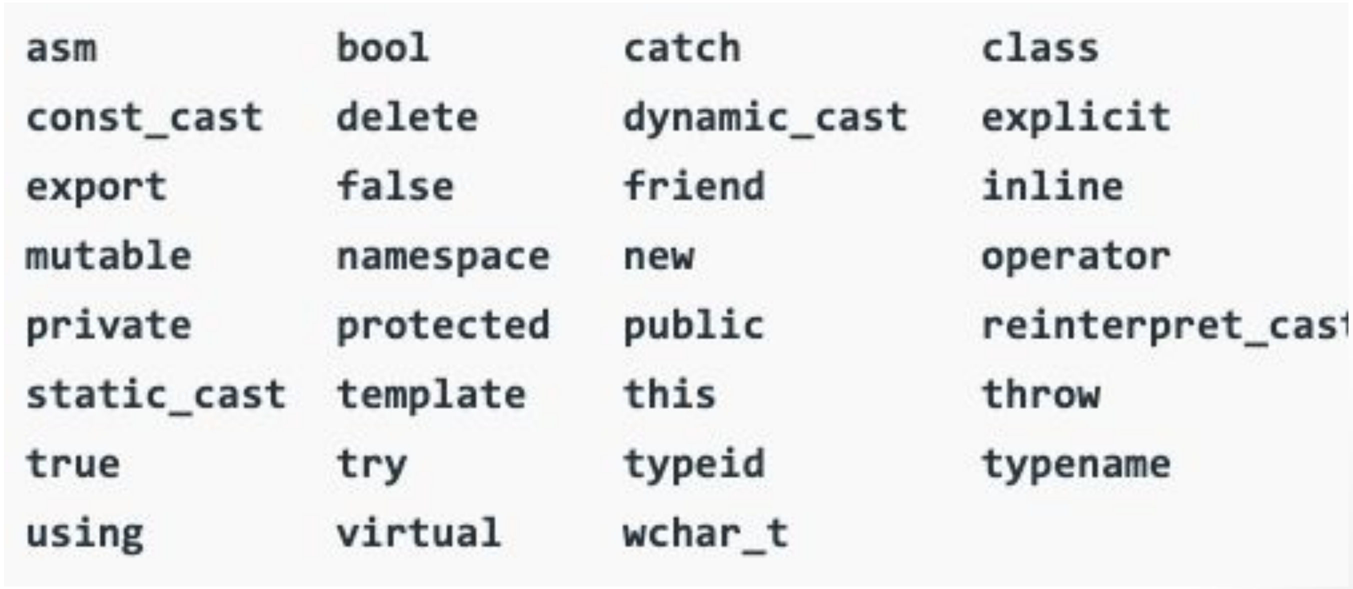
A token is the smallest element of a C++ program that is meaningful to the compiler. The C++ parser recognizes these five kinds of tokens:

- Keywords
- Identifiers
- Constants
- Strings
- Operators

Tokens are usually separated by white space.

Keywords

- Keywords are pre-defined or reserved words in a programming language. Each keyword is meant to perform a specific function in a program.
- Since keywords are referred names for a compiler, they can't be used as variable names.
- You cannot redefine keywords.



asm	bool	catch	class
const_cast	delete	dynamic_cast	explicit
export	false	friend	inline
mutable	namespace	new	operator
private	protected	public	reinterpret_cas
static_cast	template	this	throw
true	try	typeid	typename
using	virtual	wchar_t	

Identifiers

- Identifiers are used for the naming of variables, functions and arrays.
- These are user-defined names consisting of an arbitrarily long sequence of letters and digits with either a letter or the underscore(_) as a first character.
- Identifier names must differ in spelling and case from any keywords. You cannot use keywords as identifiers; they are reserved for special use.
- For Example:

  - int minutesPerHour = 60;
  - void print();
  - int _array[20];

Constants

- Constants are like normal variables. But, the only difference is, their values can not be modified by the program once they are defined.
- Constants refer to fixed values are called literals.
- Constants may belong to any of the data type.
- **Syntax:**

const data_type variable_name;

Types of Constants:

- Integer constants – Example: 0, 1, 1218, 12482
- Real or Floating-point constants – Example: 0.0, 1203.03, 30486.184
- Octal & Hexadecimal constants – Example: octal: (013)8 = (11)10, Hexadecimal: (013)16 = (19)10
- Character constants - Example: ‘a’, ‘A’, ‘z’
- String constants - Example: “Hello”

Strings

- Strings are array of characters that ends with a null character (‘\0’). This null character indicates the end of the string. Strings are always enclosed in double-quotes. Whereas, a **character** is enclosed in single quotes in C and C++.

- **Declarations for String:**
- char string[20] = {‘g’, ’e’, ‘e’, ‘k’, ‘s’, ‘f’, ‘o’, ‘r’, ‘g’, ’e’, ‘e’, ‘k’, ‘s’, ‘\0’};
- char string[20] = “software engineering”;
- char string [] = “tokens”;
- when we declare char as “string[20]”, 20 bytes of memory space is allocated for holding the string value.
- When we declare char as “string[]”, memory space will be allocated as per the requirement during the execution of the program.

Operators

- Operators are symbols that trigger an action when applied to variables and other objects. The data items on which operators act upon are called operands.

Depending on the number of operands that an operator can act upon, operators are classified as follows:

- **Unary Operators:** Those operators that require only a single operand to act upon are known as unary operators. For Example increment and decrement operators.
- **Binary Operators:** Those operators that require two operands to act upon are called binary operators. **Binary operators are classified into :**
  - Arithmetic operators
  - Relational Operators
  - Logical Operators
  - Assignment Operators
  - Conditional Operators
  - Bitwise Operators

Special Symbols

The following special symbols are used having some special meaning and thus, cannot be used for some other purpose.[] () {}, ; * = #

- **Brackets[]:** Opening and closing brackets are used as array element reference. These indicate single and multidimensional subscripts.
- **Parentheses():** These special symbols are used to indicate function calls and function parameters.
- **Braces{}:** These opening and ending curly braces mark the start and end of a block of code containing more than one executable statement.
- **Comma (,):** It is used to separate more than one statements like for separating parameters in function calls.
- **Colon(:):** It is an operator that essentially invokes something called an initialization list.
- **Semicolon(;):** It is known as a statement terminator. It indicates the end of one logical entity. That's why each individual statement must be ended with a semicolon.
- **Asterisk (*):** It is used to create a pointer variable.
- **Assignment operator(=):** It is used to assign values.
- **Pre-processor (#):** The preprocessor is a macro processor that is used automatically by the compiler to transform your program before actual compilation.

Expressions

- C++ expression consists of operators, constants, and variables which are arranged according to the rules of the language. It can also contain function calls which return values. An expression can consist of one or more operands, zero or more operators to compute a value.
- Every expression produces some value which is assigned to the variable with the help of an assignment operator.

An expression can be of following types:

- Constant Expressions
- Integral Expression
- Float Expression
- Pointer Expression
- Relational Expression
- Logical Expression
- Bitwise Expression

- **Example:**

```
#include <iostream>
using namespace std;
int main()
{

   int x;                           // variable declaration
   int y=9;                         // variable initialization
   x=y+int(10.0);                   // integral expression
   y=x+float(10);                   // float expression
   cout<<((x-y)>=(x+y));   // relational expression

cout<<"Value of x : "<<x;   // displaying the value of x.
return 0;
}
```

Operators

- An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations.
- C++ is rich in built-in operators and provide the following types of operators −
- Arithmetic Operators: +, -, *, /, %, ++, --
- Relational Operators: ==, !=, >, <, <=, >=
- Logical Operators: &&, ||, !
- Bitwise Operators: &, |, ^ , ~, <<, >>

Scope Resolution Operator (::)

In C++, scope resolution operator is used for following purposes.

1. To access a global variable when there is a local variable with same name
2. To define a function outside a class
3. To access a class's static variables
4. In case of multiple Inheritance
5. For namespace
6. Refer to a class inside another class

Memory Dereferencing Operators

Using pointers we can access members of class

- (::*) Pointer to Member Declarator: To declare a pointer to a member of a class.
- (*) Pointer to Member operator: To access a member using object name and a pointer to that member.
- (->*) Pointer to object operator: To access a member using a pointer to the object and a pointer to that member.

Control Structures

These are just a way to specify flow of control in programs. Any algorithm or program can be more clear and understood if they use self-contained modules called as logic or control structures.

- It basically analyzes and chooses in which direction a program flows based on certain parameters or conditions. There are three basic types of logic, or flow of control, known as:

1. Sequence Structure (Straight Line)
2. Selection Structure(Branching/Conditional)
3. Loop Structure(Iteration or Repetition)