OTZ

OPENTECHZ

A dream to realize your dream

C++

By Parthasarathi Swain

C++ Character Set

The C++ character set includes all characters used to write valid C++ programs. It's similar to learning alphabets before writing English sentences.

Letters:

- Uppercase: A-Z
- Lowercase: a-z

Digits:

- 0-9

Special Symbols:

- + - * / = < > ! % ^ & | ~ ? : ; , . # ( ) { } [ ] \\ \" \'

Whitespace Characters:

- Space, Tab (\t), Newline (\n)

Other Characters:

- Escape sequences like \n, \t, \\, \', \"

Example 1: *Displaying character values*

```
#include <iostream>
using namespace std;

int main() {
    char ch1 = 'A';
    char ch2 = 'z';
    cout << "Characters: " << ch1 << " and " << ch2 << endl;
    return 0;
}
```

```
Characters: A and z
```

Example 2 : Using digits and symbols

```
#include <iostream>
using namespace std;

int main() {
    int a = 10, b = 5;
    cout << "Sum: " << a + b << endl;
    return 0;
}
```

```
Sum: 15
```

What are Escape Sequences in C++?

Escape sequences are special characters used to control output format. They begin with a **backslash \** and are mainly used in **printf()** and **cout**.

Common Escape Sequences:

- \n → New line
- \t → Horizontal tab
- \\\\ → Backslash
- \\" → Double quote
- \\' → Single quote
- \a → Alert sound (beep)
- \b → Backspace
- \r → Carriage return (start of line)
- \f → Form feed (page break)

Example Program Using cout

```
#include <iostream>
using namespace std;

int main() {
    cout << "Hello\nWorld\n\n";
    cout << "Name:\tJohn\n\n";
    cout << "Path: C:\\Program Files\\App\n\n";
    cout << "She said: \"Welcome\"\n\n";
    cout << "It's a good day.\n\n";
    cout << "Beep sound\a\n\n";
    cout << "Test\bX\n\n";
    cout << "Start\rEnd\n\n";
    cout << "Page 1\fPage 2\n";

    return 0;
}
```

Example Program Using printf

```
#include <cstdio>

int main() {
    printf("Hello\nWorld\n\n");
    printf("Name:\tJohn\n\n");
    printf("Path: C:\\Program Files\\App\n\n");
    printf("She said: \"Welcome\"\n\n");
    printf("It's a good day.\n\n");
    printf("Beep sound\a\n\n");
    printf("Test\bX\n\n");
    printf("Start\rEnd\n\n");
    printf("Page 1\fPage 2\n");

    return 0;
}
```

What are Tokens?

In **C++**, **tokens** are the **smallest building blocks** of a program. They are like **words in a sentence**, giving structure and meaning.

1. Identifiers

Names given by the programmer to variables, functions, classes, etc.
Example: total, getData(), Student

2.Keywords

Predefined, reserved words with special meaning.
Example: int, float, if, while, return

3. Constants

Fixed values that do not change during execution.
Example: 100, 3.14, 'A', "Hello“

4. Operators

Symbols that perform operations on variables/values.
Example: +, -, *, /, ==, &&

5. Separators

Characters that separate tokens.
Example: ,, ;, (), {}, []

What is an Identifier?

- An **identifier** is the **name** used to identify **variables**, **functions**, **classes**, **objects**, **arrays**, etc.
- It is **created by the programmer** and is **not predefined** like keywords.

Rules

1. Must begin with a letter (A-Z or a-z) or an underscore (_)
   Example: name, _value are valid

2. After the first character, digits (0-9) can also be used
   Example: mark1, student_22

3. Cannot use C++ keywords as identifiers
   Example: int, float, class cannot be used as variable names

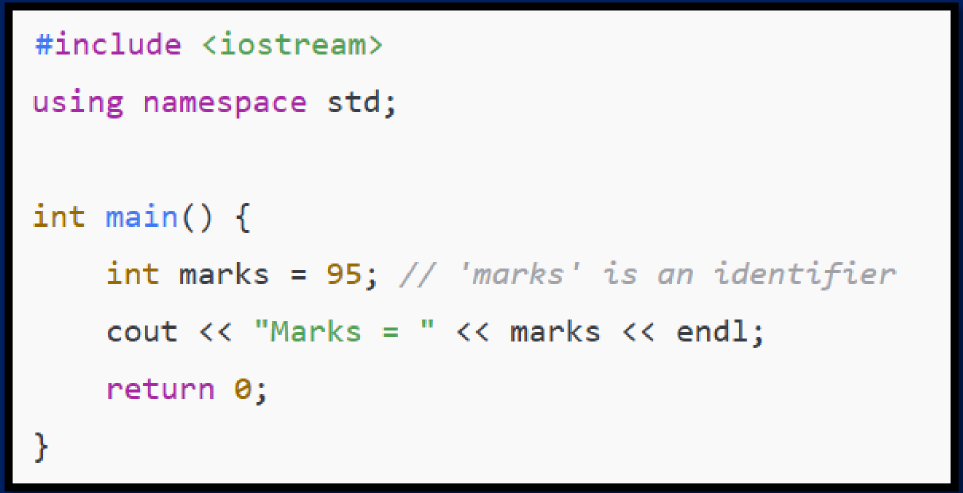
4. No special characters allowed except underscore (_)
   Characters like @, #, $, -, . are not allowed

5. Identifiers are case-sensitive
   Example: Total and total are treated as two different identifiers

6. Should be meaningful and descriptive
   Example: Use totalMarks instead of tm for better readability

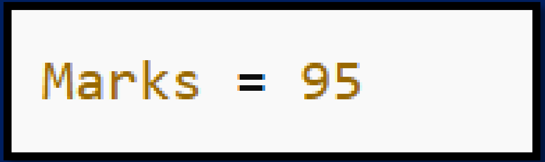
7. No space is allowed in an identifier
   Example: firstName is valid, first name is invalid

Example 1:



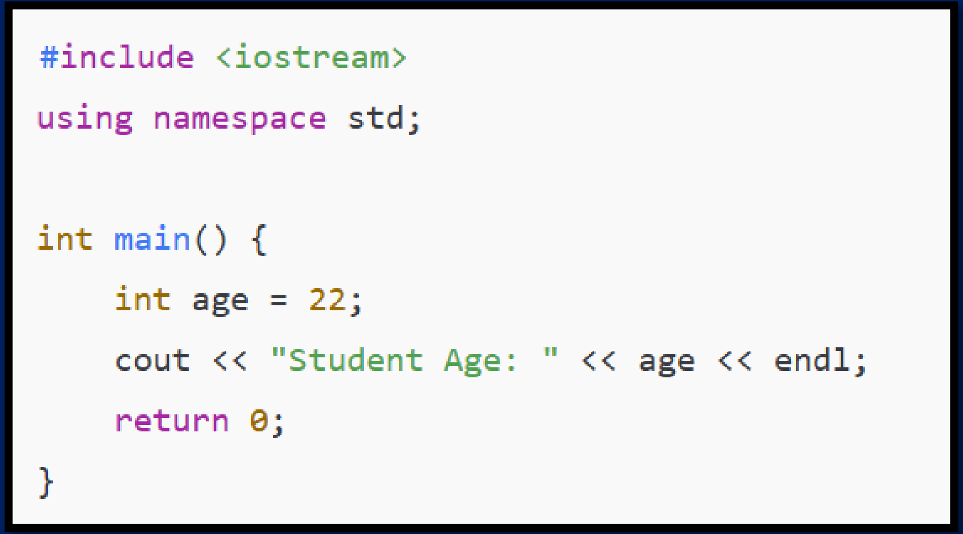
```cpp
#include <iostream>
using namespace std;

int main() {
    int marks = 95; // 'marks' is an identifier
    cout << "Marks = " << marks << endl;
    return 0;
}
```



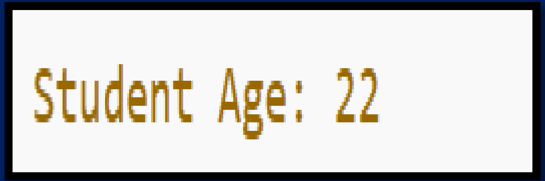
```
Marks = 95
```

Example 2 :



```cpp
#include <iostream>
using namespace std;

int main() {
    int age = 22;
    cout << "Student Age: " << age << endl;
    return 0;
}
```



```
Student Age: 22
```

What are Keywords?

- **Keywords** are **reserved words** in C++.
- Each keyword has a **specific meaning** and **function** in the language.
- They **cannot be used** as identifiers (like variable or function names).

Total Keywords in C++ (C++17 Standard):

- **95 keywords** in total
  - 73 core keywords
  - 18 from updates (C++11, C++14, C++17)
  - 4 alternative tokens (e.g., and, or, not)

Category	Keywords
Data Types	int, float, double, char, bool, void
Control Flow	if, else, switch, case, default, goto
Loops	for, while, do, break, continue
Access Specifiers	public, private, protected
OOP Related	class, struct, union, this, virtual, new, delete
Exception Handling	try, catch, throw
Others	return, const, static, sizeof, typedef

Important Notes:

- **Case-sensitive** → Int is not the same as int
- Can not use keywords as

Example 1:

```cpp
#include <iostream>
using namespace std;

int main() {
    int a = 10; // 'int' is a keyword
    cout << "Value: " << a << endl;
    return 0;
}
```

```
Value: 10
```

Example 2 :

```cpp
#include <iostream>
using namespace std;

void greet() {
    cout << "Hello!" << endl;
}

int main() {
    greet(); // 'void' and 'return' are keywords
    return 0;
}
```

```
Hello!
```

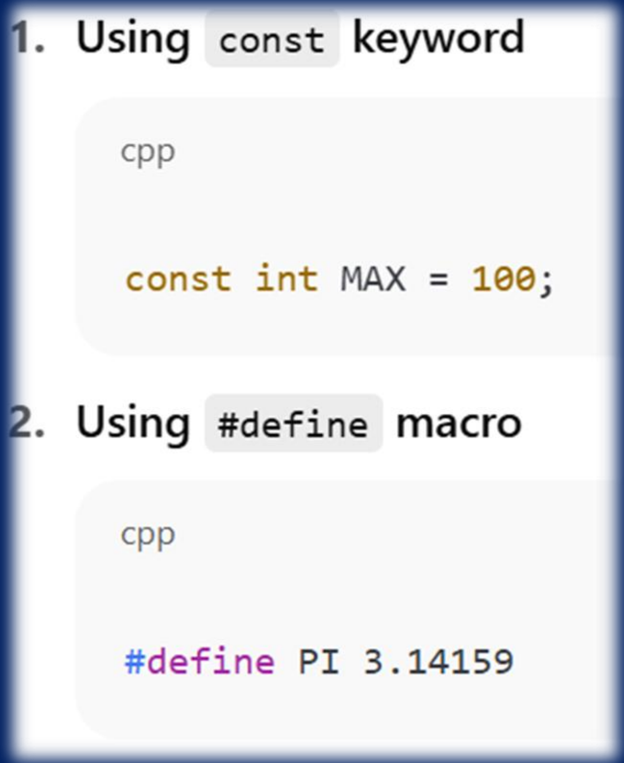
What is a Constant?

A **constant** is a **fixed value** that **does not change** during the execution of a program.

Types of Constants:

1. **Integer Constants**
   - Whole numbers without decimal
   - Example: 100, -25
2. **Floating-Point Constants**
   - Numbers with decimal points
   - Example: 3.14, -0.001
3. **Character Constants**
   - A single character enclosed in single quotes
   - Example: 'A', '9'
4. **String Constants**
   - A sequence of characters enclosed in double quotes
   - Example: "Hello", "C++ Programming"
5. **Boolean Constants**
   - Represents logical values
   - Example: true, false

Ways to Declare Constants:



1. Using `const` keyword

```cpp
const int MAX = 100;
```

2. Using `#define` macro

```cpp
#define PI 3.14159
```

Example 1:

```cpp
#include <iostream>
using namespace std;

int main() {
    const int speed = 60;
    cout << "Speed Limit: " << speed << endl;
    return 0;
}
```

```
Speed Limit: 60
```

Example 2 :

```cpp
#include <iostream>
using namespace std;

int main() {
    #define PI 3.14
    cout << "Value of PI: " << PI << endl;
    return 0;
}
```

```
Value of PI: 3.14
```

What is an Operator?

An operator is a symbol that performs an operation on variables and values.

Types of Operators:

1.Arithmetic Operators
- Perform basic math operations
- +, -, *, /, %
- **Example**: a + b, x * y

2.Relational (Comparison) Operators
- Compare two values
- ==, !=, >, <, >=, <=
- **Example**: a == b, x < y

3.Logical Operators
- Combine or invert logical values
- && (AND), || (OR), ! (NOT)
- **Example**: a > 0 && b < 10

4.Assignment Operators
- Assign values to variables
- =, +=, -=, *=, /=, %=
- **Example**: a = 5, b += 2

5.Increment/Decrement Operators
- Increase or decrease value by 1
- ++, -- (prefix/postfix)
- **Example**: ++x, y--

6.Bitwise Operators
- Operate on bits
- &, |, ^, ~, <<, >>
- **Example**: a & b, x << 2

7.Conditional (Ternary) Operator
- Short form of if-else
- ? :
- **Example**: max = (a > b) ? a : b;

8.Sizeof Operator
- Returns size of a data type or variable
- **Example**: sizeof(int), sizeof(arr)

Example 1:

```cpp
#include <iostream>
using namespace std;

int main() {
    int x = 10, y = 5;
    cout << "Product = " << x * y << endl;
    return 0;
}
```

```
Product = 50
```

Example 2 :

```cpp
#include <iostream>
using namespace std;

int main() {
    int age = 20;
    if (age >= 18 && age <= 25) {
        cout << "Eligible" << endl;
    }
    return 0;
}
```

```
Eligible
```

What is an Separators?

Separators are **symbols** used to **separate different parts** of a C++ program, like statements, parameters, and blocks of code. They help **structure** and **organize** the code properly.

Types of Operators:

1.Semicolon (;)
- Ends a statement
- **Example**: int a = 10;

2.Comma (,)
- Separates multiple variables or arguments
- **Example**: int x = 5, y = 10;

3.Parentheses (( ))
- Used in function calls, condition checks, loops
- **Example**: if (x > 0), sum(a, b)

4.Braces ({ })
- Define the beginning and end of a block of code
- **Example**:{

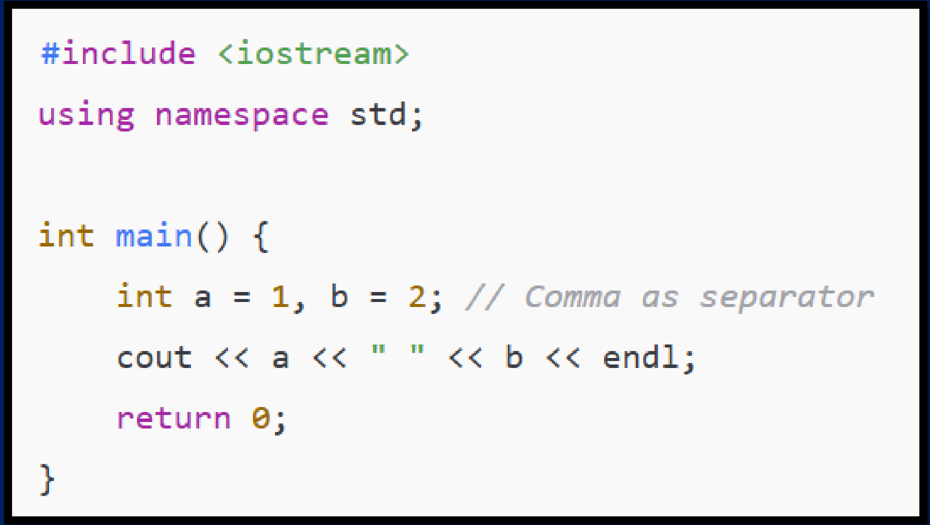
  }

5.Brackets ([ ])
- Used for array declarations and indexing
- **Example**: arr[0] = 5;

6.Colon (:)
- Used in labels (for case in switch or inheritance)
- **Example**: case 1:, class B : public A

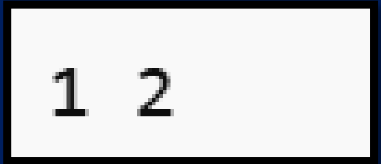
7.Hash (#)
- Used for preprocessor directives
- **Example**: #include <iostream>

Example 1:



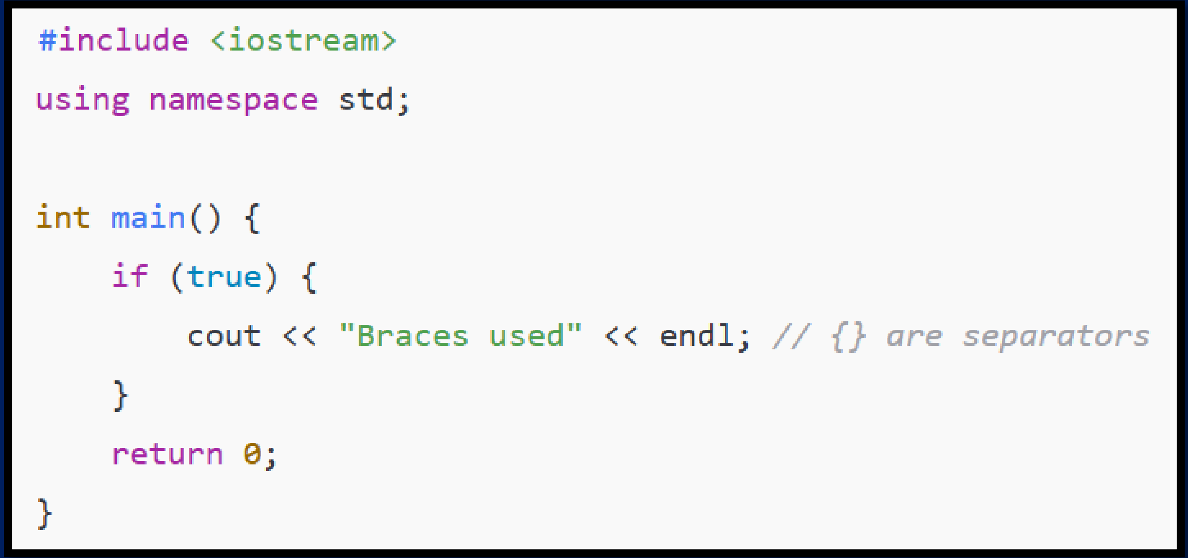
```cpp
#include <iostream>
using namespace std;

int main() {
    int a = 1, b = 2; // Comma as separator
    cout << a << " " << b << endl;
    return 0;
}
```



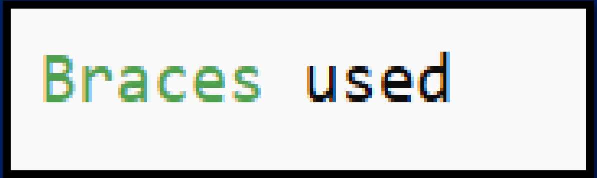
```
1 2
```

Example 2 :



```cpp
#include <iostream>
using namespace std;

int main() {
    if (true) {
        cout << "Braces used" << endl; // {} are separators
    }
    return 0;
}
```



```
Braces used
```

Thank You