C++ Identifiers

Last Updated : 15 Sep, 2025

In C++ programming language, **identifiers** are the unique names assigned to variables, functions, classes, structs, or other entities within the program. Let's take a look at an example:

```
// Creating a variable
int val = 10;

// Creating a function
void func() {}
```

In the above code, the words **val** and **func** are identifiers. Basically, everything named by a programmer is an identifier and is used to refer to the entity later in the program.

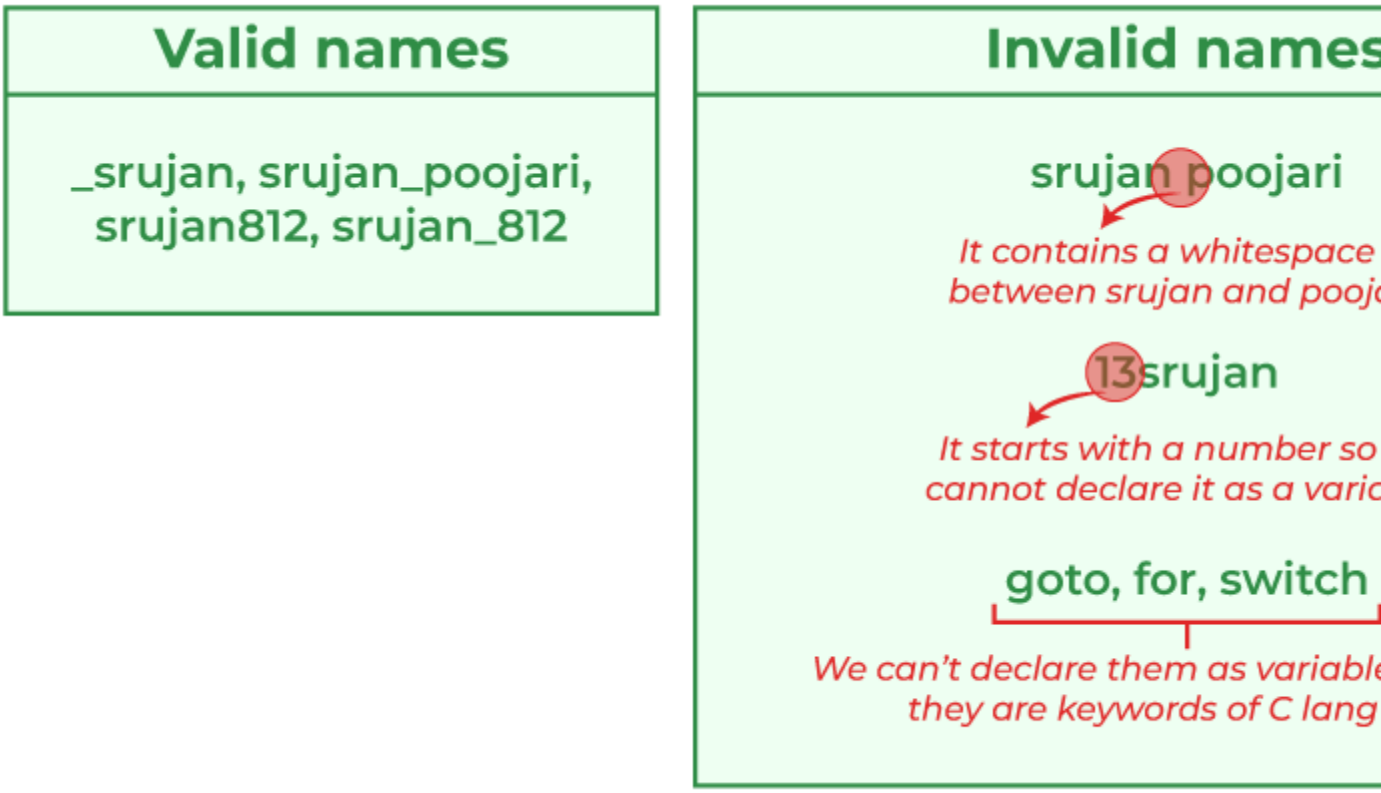
Rules for Naming an Identifier

We can use any word as an identifier as long as it follows the following rules:

- An identifier can consist of **letters** (A-Z or a-z), **digits** (0-9), and **underscores (_)**. Special characters and spaces are not allowed.
- An identifier can only begin with a **letter or an underscore only.**
- C++ has reserved **keywords** that cannot be used as identifiers since they have predefined meanings in the language. For example, **int** cannot be used as an identifier as it already has some predefined meaning in C++. Attempting to use these as identifiers will result in a compilation error.
- Identifier must be **unique** in its namespace.

Additionally, C++ is a case-sensitive language so the identifier such as **Num** and **num** are treated as different. The below images show some valid and invalid C++ identifiers.

To know more about identifiers naming rules, refer to this article – Naming Convention in C++



Example of Valid/Invalid Identifiers

Example

In this example, we have used the identifiers by following the guidelines and we use identifiers to name a class, function, integer data type, etc. If you are not aware of functions and classes in C++ then don't worry, you will learn them soon. The below code is run successfully which means we named them correctly.

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

// Here Car identifier is used to refer to below class
class Car {
    string Brand;
    string model;
    int year;
};

// getSum identifier is used to call the below
// function
```

```
void getSum(int a, int b) {
    int _sum = a + b;
    cout << "The sum is: " << _sum;
}

int main() {

    // Identifiers used as variable names
    int studentAge = 20;
    double accountBalance = 1000.50;
    string student_Name = "Karan";

    getSum(2, 10);

    return 0;
}
```

Output

```
The sum is: 12
```

Identifier Naming Conventions

Naming conventions are not the rules enforced by C++ language but are suggestions for naming variables by the programming community for easier understanding. Some of the naming conventions are as follows:

For Variables:

- Use camelCase (for constants, you can use UPPER_SNAKE_CASE)
- Start with lowercase alphabet.
- Use descriptive, meaningful names.
- For example, frequencyCount, personName

For Functions:

- Use camelCase.
- Use Verb or verb phrases for naming.
- For example, getName(), countFrequency(), etc.

For Classes:

- Use PascalCase
- Use Nouns or noun phrases for naming.
- For example, Car, Person, etc

Once again, above are some suggestions for naming identifiers do not rule. They also depend on the pro

C++ Keywords

Last Updated : 17 Sep, 2025

Keywords are the reserved words that have special meanings. Since their meanings are reserved, we cannot redefine them or use them for a different purpose.

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

// 'int' is a keyword
int main() {

    // 'int' is a keyword
    int age = 20;

    // 'if' is a keyword
    if (age > 18) {
        cout << "Adult";
    }

    // 'return' is a keyword
    return 0;
}
```

Output

```
Adult
```

How to Identify C++ Keywords

1. **Syntax Highlighting**: Most modern IDEs (like Visual Studio, CLion, Code::Blocks) highlight keywords in a different color. This makes them stand out from variables or function names.
2. **Compiler Errors**: If you mistakenly use a keyword as a variable name, your code won't compile. **Example:**

```
int return = 10;   // Error: 'return' is reserved
```

Note: The number of keywords C++ has evolved over time as new features were added to the language. **For example**, C++ 98 had 63 keywords, C++ 11 had 84 keywords, C++.

Keywords vs Identifiers

So, there are some properties of keywords that distinguish keywords from identifiers. They listed in the below table

Keywords	Identifiers
Keywords are predefined/reserved words	identifiers are the values used to define different programming items like a variable, integers, structures, and unions.
It defines the type of entity.	It classifies the name of the entity.
A keyword contains only alphabetical characters,	an identifier can consist of alphabetical characters, digits, and underscores.
It should be lowercase.	It can be both upper and lowercase.
No special symbols or punctuations are used in keywords and identifiers.	No special symbols or punctuations are used in keywords and identifiers. The only underscore can be used in an identifier.
Example: int, char, while, do.	**Example:** geeksForGeeks, geeks_for_geeks, gfg, gfg12.

Variables in C++

Last Updated : 16 Jan, 2026

In C++, variable is a name given to a memory location. It is the basic unit of storage in a program. The value stored in a variable can be accessed or changed during program execution.

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

int main() {
    // Creating a single character variable
    int num = 3;

    // Accessing and printing above variable
    cout << num << endl;

    // Update the value
    num = 7;

    // Printing the updated value
    cout << num;

    return 0;
}
```

Creating a Variable

Creating a variable and giving it a name is called variable definition (sometimes called variable declaration). The syntax of variable definition is:

type name;
where, type is the type of data that a variable can store, and name is the name assigned to the variable. Multiple variables of the same type can be defined as:

type name1, name2, name3 ....;
The data type of a variable is selected from the list of data types supported by C++.

Initializing

A variable that is just defined may not contain some valid value. We have to initialize it to some valid initial value. It is done by using an assignment operator '=' as shown:

int num;
num = 3;
Definition and initialization can also be done in a single step as shown:

int num = 3;
The integer variable **num** is initialized with the value 3. Values are generally the literals of the same type.
Note: *The value we assign should be of the same type as the variable, i.e. we can't assign a decimal value (e.g. 3.14) to an integer type variable.*

Accessing and Updating

The main objective of a variable is to store the data so that it can be retrieved or update any time. Accessing can be done by simply using its assigned name and updating the value using '**=' (assignment operator)**.

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

int main(){
    int num = 3;
    cout << num << endl;
    // Updating the value
    num = 7;
    cout << num;
    return 0;
}
```

Output

```
3
7
```

Rules For Naming Variable

The names given to a variable are called **identifiers**. There are some rules for creating these identifiers (names):

- A name can only contain **letters** (A-Z or a-z), **digits** (0-9), and **underscores (_)**.
- It should start with a **letter or an underscore only.**
- It is case sensitive.
- The name of the variable should not contain any whitespace and special characters (i.e. #, $, %, *, etc).

- We cannot use C++ keyword (e.g. float, double, class) as a variable name.

How are variables used?

Variables are the names given to the memory location which stores some value. These names can be used in any place where the value it stores can be used. **For example,** we assign values of the same type to variables. But instead of these values, we can also use variables that store these values.

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

int main() {
    int num1 = 10, num2;

    // Assigning num1's value to num2
    num2 = num1;
    cout << num1 << " " << num2;
    return 0;
}
```

Output

```
10 10
```

Addition of two integers can be done in C++ using '+' operator as shown:

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

int main() {
    cout << 10 + 20;
    return 0;
}
```

Output

```
30
```

We can do the above operation using the variables that store these two values.

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

int main() {
```

```
    int num1 = 10, num2 = 20;
    cout << num1 + num2;
    return 0;
}
```

Output

```
30
```

Constant Variables

In C++, a constant variable is one whose value cannot be changed after it is initialized. This is done using the **const** keyword.

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

int main() {
    const int num = 10;
    cout << num;
    return 0;
}
```

Output

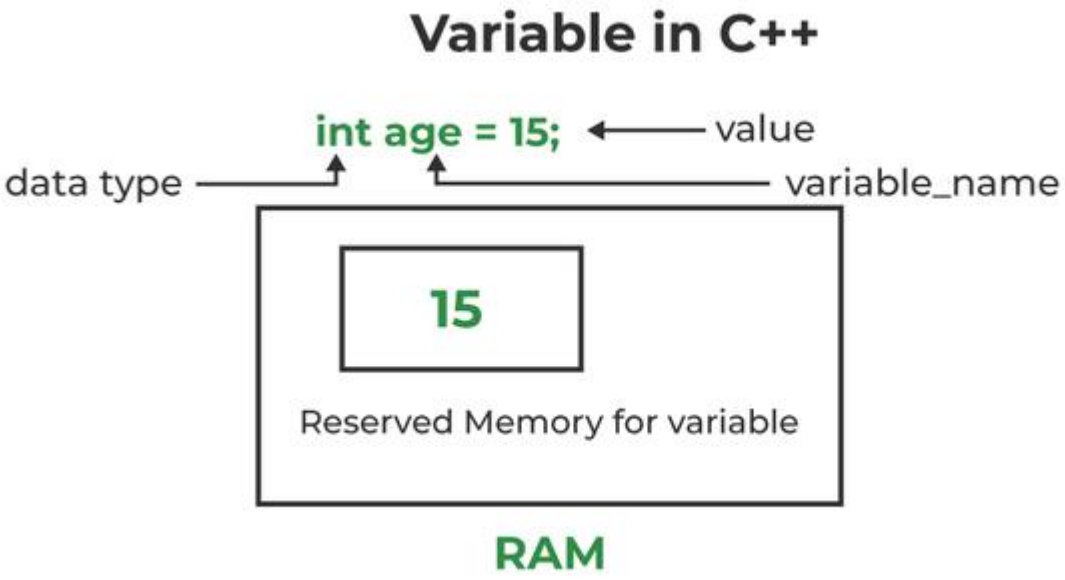
```
10
```

Scope of Variables

Scope of variable is the region inside the program where the variable can be referred to by using its name. Basically, it is the part of the program where the variable exists. Proper understanding of this concept requires the understanding of other concepts such as functions, blocks, etc.

Memory Management of Variables

When we create or declare a variable, a fixed-size memory block is assigned to the variable, and its initial value is a garbage value. Initialization assigns a meaningful value using the assignment operator. Variables essentially manipulate specific memory locations, and their stored data is accessed via their names.



Moreover, different variables may be stored in different section of memory according to its storage class.