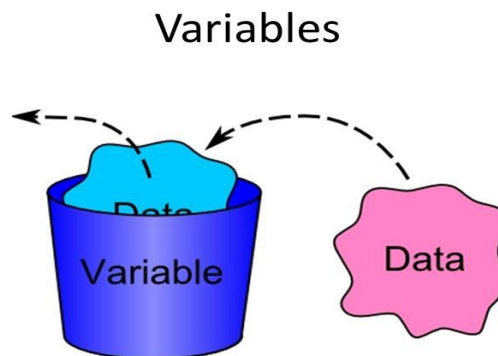




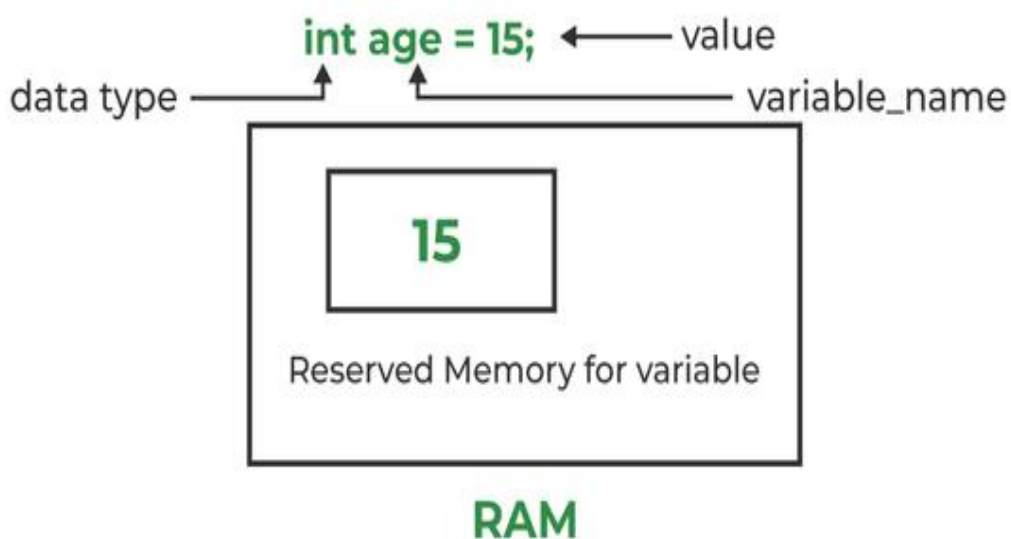
Variables

Variables are containers for storing data values.



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.

Variable in C++





There are different **types** of variables (defined with different keywords), for example:

- **int** - stores integers (whole numbers), without decimals, such as 123 or -123
- **double** - stores floating point numbers, with decimals, such as 19.99 or -19.99
- **char** - stores single characters, such as 'a' or 'B'. Char values are surrounded by single quotes
- **string** - stores text, such as "Hello World". String values are surrounded by double quotes
- **bool** - stores values with two states: true or false

Declaring (Creating) Variables

To create a variable, specify the type and assign it a value:

```
type variableName = value;
```

Where *type* is one of C++ types (such as **int**), and *variableName* is the name of the variable (such as **x** or **myName**). The **equal sign** is used to assign values to the variable. To create a variable that should store a number, look at the following code

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

int main() {
    int myNum = 15;
    cout << myNum;
    return 0;
}
```

You can also declare a variable without assigning the value, and assign the value later:

```
int myNum;
myNum = 15;
cout << myNum;
```



Note that if you assign a new value to an existing variable, it will overwrite the previous value:

```
int myNum = 15;    // myNum is 15
myNum = 10;        // Now myNum is 10
cout << myNum;     // Outputs 10
```

Display Variables

The ***cout*** object is used together with the << operator to display variables. To combine both text and a variable, separate them with the << operator:

```
int myAge = 35;
cout << "I am " << myAge << " years old.";
```

Add Variables Together

To add a variable to another variable, you can use the + operator:

```
int x = 5;
int y = 6;
int sum = x + y;
cout << sum;
```

Declare Many Variables

To declare more than one variable of the **same type**, use a comma-separated list:

```
int x = 5, y = 6, z = 50;
cout << x + y + z;
```

One Value to Multiple Variables

You can also assign the **same value** to multiple variables in one line:

```
int x, y, z;
x = y = z = 50;
cout << x + y + z;
```



Identifiers

All C++ **variables** must be **identified** with **unique names**. These unique names are called **identifiers**. Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).

```
// Good
int minutesPerHour = 60;

// OK, but not so easy to understand what m actually is
int m = 60;
```

Constants

When you do not want others (or yourself) to change existing variable values, use the `const` keyword (this will declare the variable as "constant", which means **unchangeable and read-only**):

```
const int myNum = 15;    // myNum will always be 15
myNum = 10;             // error: assignment of read-only variable 'myNum'
```

You should always declare the variable as constant when you have values that are unlikely to change:

```
const int minutesPerHour = 60;
const float PI = 3.14;
```

Not: When you declare a constant variable, it must be assigned with a value:

```
const int minutesPerHour;
minutesPerHour = 60;    // error
```

Example 1

Write a program that stores different data about a college student:

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

int main() {
```



```
// Student data
int studentID = 15;
int studentAge = 23;
float studentFee = 75.25;
char studentGrade = 'B';

// Print variables using cout
cout << "Student ID: " << studentID << "\n";
cout << "Student Age: " << studentAge << "\n";
cout << "Student Fee: " << studentFee << "\n";
cout << "Student Grade: " << studentGrade << "\n";

return 0;
}
```

Output

```
Student ID: 15
Student Age: 23
Student Fee: 75.25
Student Grade: B
```

Example 2

Write a program to calculate the area of a rectangle (by multiplying the length and width)

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

int main() {
    // Create integer variables
    int length = 4;
    int width = 6;
    int area;

    // Calculate the area of a rectangle
    area = length * width;

    // Print the variables
    cout << "Length is: " << length << "\n";
}
```



```
cout << "Width is: " << width << "\n";  
cout << "Area of the rectangle is: " << area << "\n";  
  
return 0;  
}
```

Output

```
Length is: 4  
Width is: 6  
Area of the rectangle is: 24
```

User Input

You have already learned that **cout** is used to output (print) values. Now we will use **cin** to get user input. **cin** is a predefined variable that reads data from the keyboard with the extraction operator (>>).

In the following code, the user can input a number, which is stored in the variable x. Then we print the value of x:

```
int x;  
cout << "Type a number: ";           // Type a number and press enter  
cin >> x;                             // Get user input from the keyboard  
cout << "Your number is: " << x;      // Display the input value
```

Not: **cout** is pronounced "see-out". Used for **output**, and uses the insertion operator (<<) while **cin** is pronounced "see-in". Used for **input**, and uses the extraction operator (>>).

Example 3

In this example, the user must input two numbers. Then we print the sum by calculating (adding) the two numbers:

```
#include <iostream>
```



```
using namespace std;

int main() {
    int x, y;
    int sum;
    cout << "Type a number: ";
    cin >> x;
    cout << "Type another number: ";
    cin >> y;
    sum = x + y;
    cout << "Sum is: " << sum;
    return 0;
}
```

Homework: Write a program to calculate the area of a square using user input (*cin*)