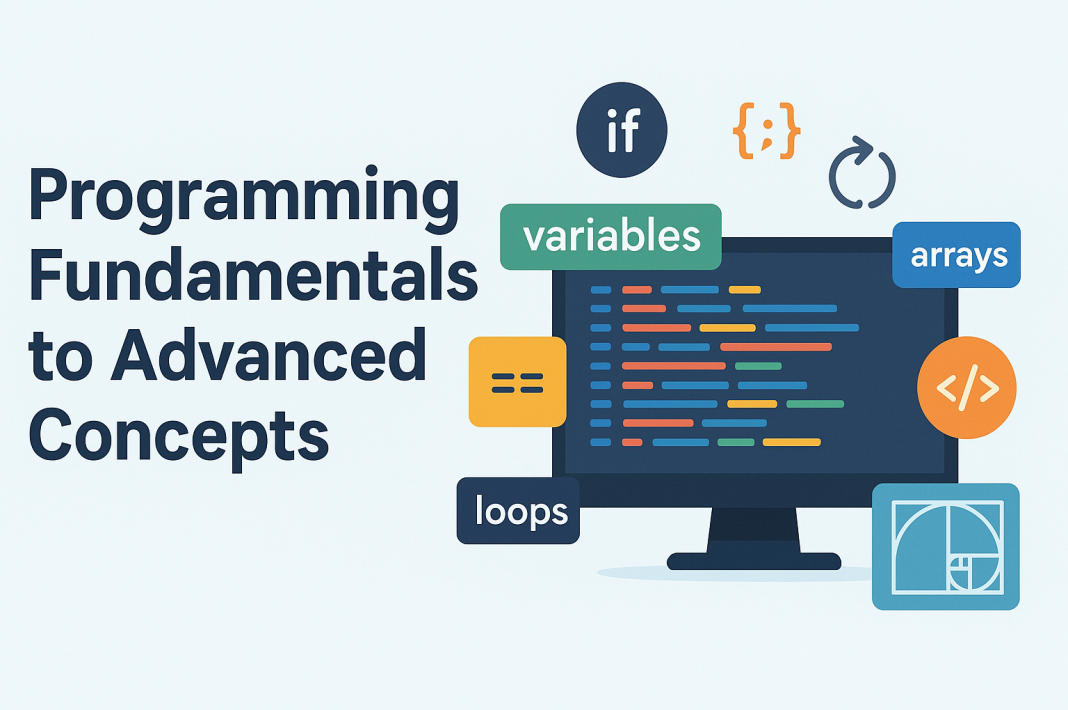
Programming Guide 2025: Variables, Loops, Arrays, Conditional Statements & Programs with Examples



Programming is the art of instructing a computer to perform specific tasks. In this detailed article by TheCubicals, we will move
step by step from the very basics to advanced programming concepts. By the end, you will not only understand theoretical concepts
but also practice with real code examples in multiple languages like C, C++, Java, and Python.

---

1. INTRODUCTION TO PROGRAMMING

Programming is the process of writing instructions that a computer can understand and execute. These instructions are written in
specific programming languages such as C, C++, Java, or Python. A program is nothing but a set of statements written in a logical
sequence.

WHY LEARN PROGRAMMING?

* It builds problem-solving skills.
* It opens doors to careers in software development, AI, data science, and more.
* It helps you understand how technology around us works.

---

2. VARIABLES

A variable is a storage location in memory with a name that stores data which can change during program execution. Think of it as
a container to hold values.

EXAMPLE: DECLARING AND USING VARIABLES

Select Language: C C++ Java Python

```
// C Example: Variables
#include
int main() {
```

```
int age = 20;
float marks = 88.5;
char grade = 'A';
printf("Age: %d, Marks: %.2f, Grade: %c", age, marks, grade);
return 0;
}
```

```
// C++ Example: Variables
#include
using namespace std;
int main() {
int age = 20;
float marks = 88.5;
char grade = 'A';
cout << "Age: " << age << ", Marks: " << marks << ", Grade: " << grade;
return 0;
}
```

```
// Java Example: Variables
public class Main {
public static void main(String[] args) {
int age = 20;
float marks = 88.5f;
char grade = 'A';
System.out.println("Age: " + age + ", Marks: " + marks + ", Grade: " + grade);
}
}
```

```
# Python Example: Variables
age = 20
marks = 88.5
grade = 'A'
print("Age:", age, ", Marks:", marks, ", Grade:", grade)
```

---

3. IDENTIFIERS

An identifier is the name given to variables, functions, classes, etc. They must follow certain rules:

* They cannot be keywords.
* They must begin with a letter or underscore.
* They can contain letters, digits, and underscores.
* They are case-sensitive.

VALID IDENTIFIERS

* age, student_name, totalMarks, _count

INVALID IDENTIFIERS

* 123name, my-name, class (reserved keyword)

---

4. OPERATORS

Operators are special symbols that perform operations on variables and values. They are the backbone of logical and mathematical
expressions in programming.

TYPES OF OPERATORS:

* Arithmetic Operators: +, -, *, /, %
* Relational Operators: ==, !=, >, <, >=, <=
* Logical Operators: &&, ||, !
* Assignment Operators: =, +=, -=, *=, /=
* Increment/Decrement: ++, --

EXAMPLE: USING OPERATORS

Select Language: C C++ Java Python

```
// C Example: Operators
#include
int main() {
int a = 10, b = 3;
printf("Sum: %d\n", a + b);
printf("Difference: %d\n", a - b);
printf("Product: %d\n", a * b);
```

```
printf("Division: %d\n", a / b);
printf("Modulus: %d\n", a % b);
return 0;
}
```

```
// C++ Example: Operators
#include
using namespace std;
int main() {
int a = 10, b = 3;
cout << "Sum: " << a + b << endl;
cout << "Difference: " << a - b << endl;
cout << "Product: " << a * b << endl;
cout << "Division: " << a / b << endl;
cout << "Modulus: " << a % b << endl;
return 0;
}
```

```
// Java Example: Operators
public class Main {
public static void main(String[] args) {
int a = 10, b = 3;
System.out.println("Sum: " + (a + b));
System.out.println("Difference: " + (a - b));
System.out.println("Product: " + (a * b));
System.out.println("Division: " + (a / b));
System.out.println("Modulus: " + (a % b));
```

```
}
}
```

```
# Python Example: Operators
a, b = 10, 3
print("Sum:", a + b)
print("Difference:", a - b)
print("Product:", a * b)
print("Division:", a / b)
print("Modulus:", a % b)
```

---

5. CONDITIONAL STATEMENTS

Conditional statements allow decision-making in programs. They execute different code blocks based on conditions.

TYPES:

* if statement
* if-else statement
* if-else if ladder
* switch statement (in some languages)

EXAMPLE: IF-ELSE STATEMENT

Select Language: C C++ Java Python

```c
// C Example: Conditional Statements
#include
int main(){
int number = 10;
if (number % 2 == 0)
printf("Even Number");
else
printf("Odd Number");
return 0;
}
```

```cpp
// C++ Example: Conditional Statements
#include
using namespace std;
int main(){
int number = 10;
if (number % 2 == 0)
cout << "Even Number";
else
cout << "Odd Number";
return 0;
}
```

```
// Java Example: Conditional Statements
public class Main {
public static void main(String[] args) {
int number = 10;
if (number % 2 == 0)
System.out.println("Even Number");
else
System.out.println("Odd Number");
}
}
```

```
# Python Example: Conditional Statements
number = 10
if number % 2 == 0:
print("Even Number")
else:
print("Odd Number")
```

6. LOOPS

Loops are used to execute a block of code repeatedly until a specific condition is met. They help reduce repetition and make code
cleaner and efficient.

TYPES OF LOOPS:

* for loop – executes a block of code a specific number of times.

* while loop – executes as long as the condition is true.
* do-while loop – similar to while, but executes at least once.

EXAMPLE: PRINTING NUMBERS FROM 1 TO 5 USING A LOOP

Select Language: C C++ Java Python

```
// C Example: for loop
#include
int main() {
for (int i = 1; i <= 5; i++) {
printf("%d ", i);
}
return 0;
}
```

```
// C++ Example: for loop
#include
using namespace std;
int main() {
for (int i = 1; i <= 5; i++) {
cout << i << " ";
}
return 0;
}
```

```
// Java Example: for loop
public class Main {
public static void main(String[] args) {
for (int i = 1; i <= 5; i++) {
System.out.print(i + " ");
}
}
}
```

```
# Python Example: for loop
for i in range(1, 6):
print(i, end=" ")
```

---

7. ARRAYS

An array is a collection of elements of the same type stored in contiguous memory locations. Arrays make it easy to store multiple
values under a single name.

EXAMPLE: DECLARING AND ACCESSING ARRAYS

Select Language: C C++ Java Python

```
// C Example: Arrays
#include
int main() {
int arr[5] = {10, 20, 30, 40, 50};
for (int i = 0; i < 5; i++) {
printf("%d ", arr[i]);
}
return 0;
}
```

```
// C++ Example: Arrays
#include
using namespace std;
int main() {
int arr[5] = {10, 20, 30, 40, 50};
for (int i = 0; i < 5; i++) {
cout << arr[i] << " ";
}
return 0;
}
```

```
// Java Example: Arrays
public class Main {
public static void main(String[] args) {
int[] arr = {10, 20, 30, 40, 50};
for (int i = 0; i < arr.length; i++) {
System.out.print(arr[i] + " ");
```

```
}
}
}
```

```
# Python Example: Arrays (using list)
arr = [10, 20, 30, 40, 50]
for i in arr:
print(i, end=" ")
```

WHY ARE ARRAYS IMPORTANT?

* They allow storing large amounts of data under one name.
* Efficient access using index numbers.
* Useful in implementing algorithms like searching and sorting.

8. SEARCHING IN ARRAYS

Searching is the process of locating a specific element in a collection of data. It is one of the most common tasks in
programming. There are multiple ways to search:

* Linear Search: Check each element one by one until the required element is found. Works for unsorted arrays but can be slow for
large datasets.

* Binary Search: Much faster but requires the array to be sorted. It repeatedly divides the array into halves until the element
is found.

Time Complexity:
- Linear Search → O(n) in worst case (checking each element).
- Binary Search → O(log n) in worst case (splitting the array repeatedly).

USE CASE IN REAL LIFE:

- Searching for a student roll number in a class list.
- Searching for a product name in an e-commerce database.
- Searching for a contact in your mobile phone directory.

---

9. SORTING AN ARRAY

Sorting means arranging data in a particular order (ascending or descending). It is extremely important because most advanced
searching algorithms, like Binary Search, only work efficiently on sorted data.

TYPES OF SORTING ALGORITHMS:

* Bubble Sort: Repeatedly swaps adjacent elements if they are in the wrong order.
* Selection Sort: Selects the smallest (or largest) element and places it at the correct position.
* Insertion Sort: Inserts elements into their correct position in a growing sorted part of the array.

* Merge Sort: Uses divide-and-conquer to split the array and then merge back in sorted order.
* Quick Sort: Uses a pivot element to partition the array into smaller subarrays and sort them recursively.

Time Complexity Overview:
- Bubble, Selection, Insertion → $O(n^2)$
- Merge Sort, Quick Sort → $O(n \log n)$ (much faster for larger datasets)

REAL LIFE APPLICATIONS:

- Sorting marks of students from highest to lowest.
- Arranging names in alphabetical order.
- Ranking players based on their scores in a game.
- Organizing files by date or size in a computer system.

---

10. FINDING THE GREATEST NUMBER

Finding the greatest number in a dataset is a basic but essential task. Instead of showing repetitive code, let's understand how
computers solve this logically:

1. Start by assuming the first element is the largest.
2. Compare it with every other element one by one.
3. If a bigger number is found, replace the current maximum.
4. At the end, the maximum variable will hold the greatest value.

REAL WORLD EXAMPLE:

- A teacher finding the highest marks scored in an exam.
- A shopping site showing the most expensive product in a list.
- Weather app displaying the highest temperature recorded in a week.

---

11. FIBONACCI SERIES

The Fibonacci sequence is a famous series where each number is the sum of the two preceding ones, starting with 0 and 1. Example
sequence: 0, 1, 1, 2, 3, 5, 8, 13...

It appears not only in programming practice but also in nature and mathematics. For example:

* The arrangement of leaves on a stem often follows the Fibonacci sequence.
* The pattern of sunflower seeds and pinecones reflects Fibonacci spirals.
* It is also used in computer algorithms like dynamic programming and recursive problem solving.

WHY FIBONACCI IS IMPORTANT IN PROGRAMMING?

- Teaches the concept of recursion.
- Helps understand dynamic programming (avoiding repeated calculations).
- Forms the basis of many mathematical algorithms used in optimization.

While it’s a simple sequence, learning Fibonacci helps you understand how to break down bigger problems into smaller subproblems,
which is a key concept in advanced programming.

PROGRAMS IN MULTIPLE LANGUAGES

In this section, we will implement some commonly asked programs in programming. You can choose your preferred programming language
from the dropdown menu and see the implementation accordingly. This helps learners compare how the same logic is written
differently across C++, Java, Python, and JavaScript.

Choose Language: C++ Java Python JavaScript

1. SEARCHING A NUMBER IN ARRAY

```
// C++ Program
#include
using namespace std;
int main() {
int arr[] = {10, 20, 30, 40, 50};
int n = 5, key = 30;
bool found = false;
for(int i=0; i < n; i++) {
if(arr[i] == key) {
cout << "Element found at index " << i;
found = true;
break;
}
```

```
}
if(!found) cout << "Element not found.";
return 0;
}
```

```
// Java Program
public class Search {
public static void main(String[] args) {
int[] arr = {10, 20, 30, 40, 50};
int key = 30;
boolean found = false;
for (int i = 0; i < arr.length; i++) {
if (arr[i] == key) {
System.out.println("Element found at index " + i);
found = true;
break;
}
}
if (!found) {
System.out.println("Element not found.");
}
}
}
```

```
# Python Program
arr = [10, 20, 30, 40, 50]
key = 30
```

```
if key in arr:
print("Element found at index", arr.index(key))
else:
print("Element not found.")
```

```
// JavaScript Program
let arr = [10, 20, 30, 40, 50];
let key = 30;
let index = arr.indexOf(key);
if (index !== -1) {
console.log("Element found at index " + index);
} else {
console.log("Element not found.");
}
```

2. SORTING AN ARRAY

```
// C++ Program
#include
#include
using namespace std;
int main() {
int arr[] = {50, 20, 40, 10, 30};
int n = 5;
sort(arr, arr+n);
for(int i=0; i < n; i++)
cout << arr[i] << " ";
```

```
return 0;
}
```

```
// Java Program
import java.util.Arrays;
public class SortArray {
public static void main(String[] args) {
int[] arr = {50, 20, 40, 10, 30};
Arrays.sort(arr);
for (int num : arr) {
System.out.print(num + " ");
}
}
}
```

```
# Python Program
arr = [50, 20, 40, 10, 30]
arr.sort()
print(*arr)  # prints values space-separated
```

```
// JavaScript Program
let arr = [50, 20, 40, 10, 30];
arr.sort((a, b) => a - b);
console.log(arr.join(" "));
```

3. FIND THE GREATEST NUMBER

```
// C++ Program
#include
using namespace std;
int main() {
int a = 15, b = 25, c = 10;
if(a >= b && a >= c)
cout << "Greatest is " << a;
else if(b >= a && b >= c)
cout << "Greatest is " << b;
else
cout << "Greatest is " << c;
return 0;
}
```

```
// Java Program
public class Greatest {
public static void main(String[] args) {
int a = 15, b = 25, c = 10;
if(a >= b && a >= c)
System.out.println("Greatest is " + a);
else if(b >= a && b >= c)
System.out.println("Greatest is " + b);
else
System.out.println("Greatest is " + c);
}
```

```
}
```

```
# Python Program
a, b, c = 15, 25, 10
print("Greatest is", max(a, b, c))
```

```
// JavaScript Program
let a = 15, b = 25, c = 10;
console.log("Greatest is " + Math.max(a, b, c));
```

4. FIBONACCI SERIES

```
// C++ Program
#include
using namespace std;
int main() {
int n = 10, t1 = 0, t2 = 1, next;
cout << "Fibonacci: ";
for(int i=1; i<=n; i++) {
cout << t1 << " ";
next = t1 + t2;
t1 = t2;
t2 = next;
}
return 0;
```

```
}
```

```
// Java Program
public class Fibonacci {
public static void main(String[] args) {
int n = 10, t1 = 0, t2 = 1;
System.out.print("Fibonacci: ");
for(int i=1; i<=n; i++) {
System.out.print(t1 + " ");
int next = t1 + t2;
t1 = t2;
t2 = next;
}
}
}
```

```
# Python Program
n = 10
a, b = 0, 1
print("Fibonacci:", end=" ")
for _ in range(n):
print(a, end=" ")
a, b = b, a+b
```

```
// JavaScript Program
```

```
let n = 10, a = 0, b = 1;
let result = "Fibonacci: ";
for(let i=0; i
```