



# LESSON 2

## The Programming Process

# Learning Objectives

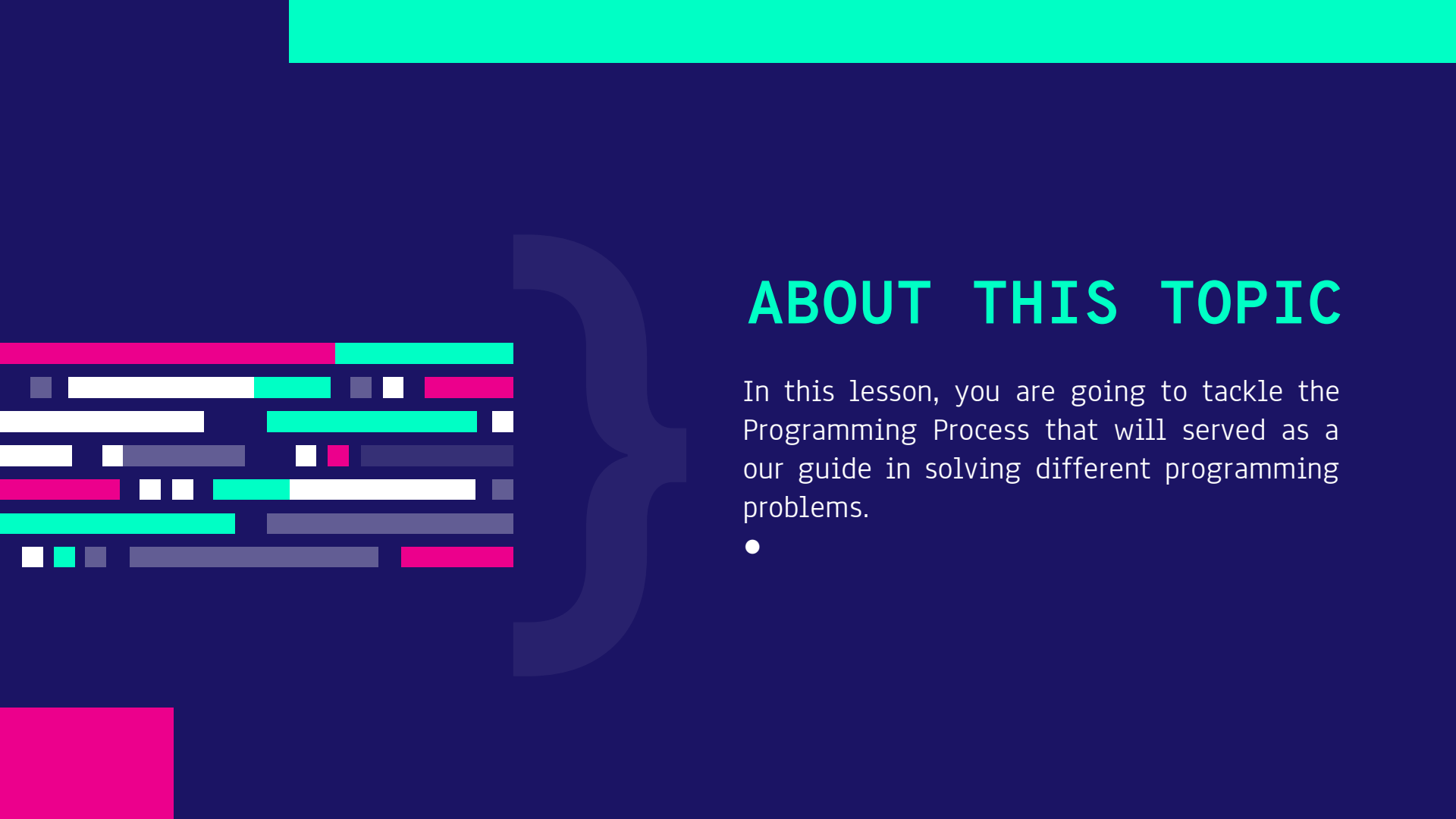
At the end of this lesson, you will be able to:

- discuss the different steps in the programming process;
- explain the purpose of Algorithm;
- differentiate the three main categories of computer instruction;
- explain the importance of desk checking;
- differentiate the two types of errors;
- differentiate the two types of program documentation;
- create a simple algorithm to solve a real-world problem;



“Perfection has to do with the  
end product, but EXCELLENCE has  
to do with the process.”

—JERRY MORAN

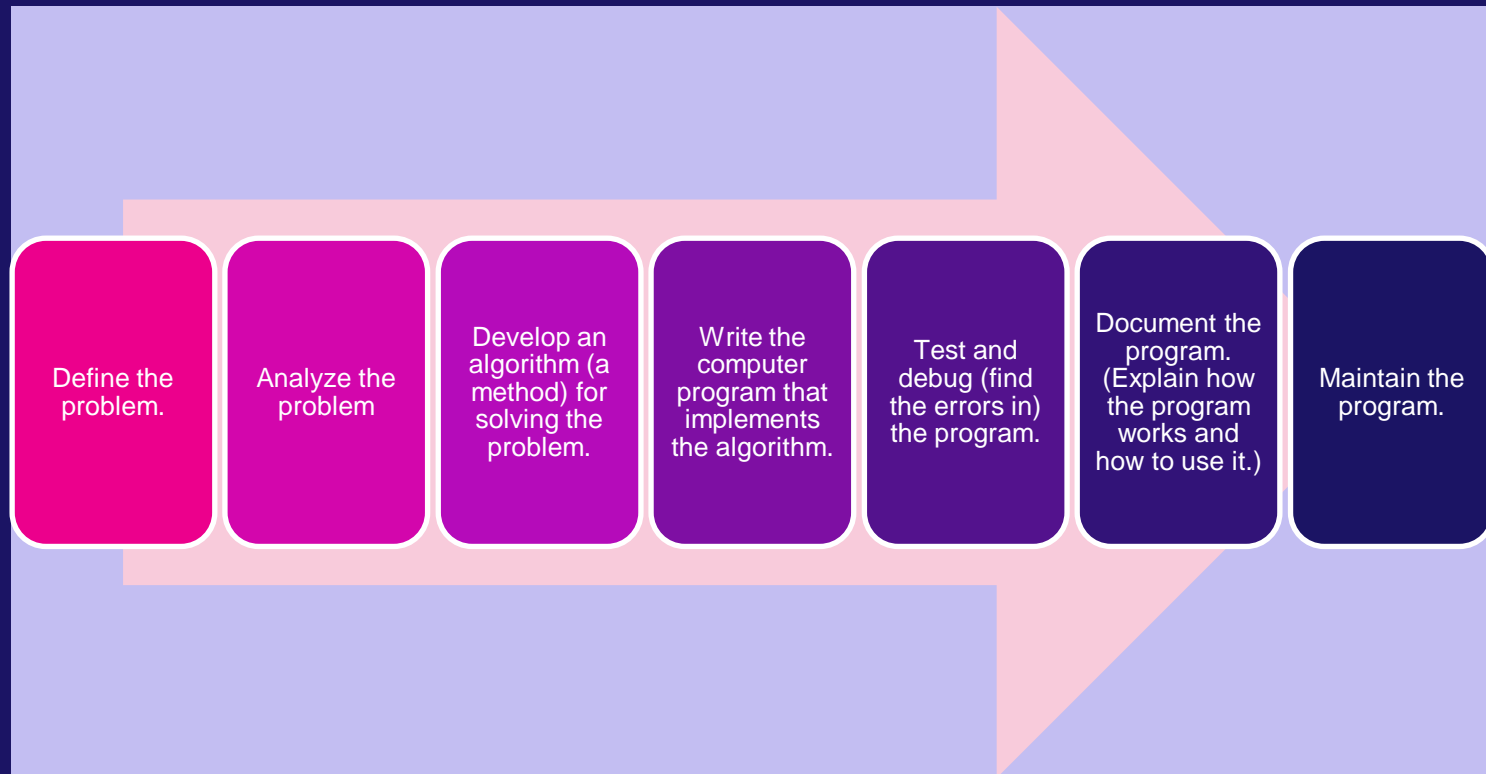


## ABOUT THIS TOPIC

In this lesson, you are going to tackle the Programming Process that will served as a our guide in solving different programming problems.

-

# The Programming Process



# The Programming Process

- The Programming process is a series of activities used by developers to create a quality program from start to finish.
- There is normally some overlap of these activities.
- For example, with a large program, a portion may be written and tested before another portion is written. Also, documentation should be done at the same time as all the other activities; each activity produces its own items of documentation that will be part of the final program documentation.

# 1. Define the Problem

- It is important that you have a complete picture of the problem so that confusion will not be a problem later on.
- Majority of the problem that arises in the middle of the programming process is due to the lack of complete information or the problem is not well established.

## 2. Analyze the Problem

- We analyze the problem to ensure that we have the clearest possible understanding of it.
- Determine the general requirement such as but not limited to, main inputs to the program and the main outputs from the program.
- If there are different ways to solving the problem, consider the alternatives and choose the best or most appropriate one.



### 3. Develop an Algorithm to Solve the Problem

- An **algorithm** is a set of instructions that, if faithfully followed, will produce a solution to a given problem or perform some specified task.
- A problem can be solved in different ways like how you solve mathematical problems. You need to decide for the most suitable solution to solve a problem and that is where you will base your algorithm.

### 3. Develop an Algorithm to Solve the Problem

- A programmer must write the instructions in the algorithm in such a way that they can be easily converted into a form that the computer can follow.
- Computer instructions fall into three main categories:
  - Input instructions
  - Processing instructions
  - Output instructions

# Three Categories of Computer Instruction

- **Input instructions** - used for supplying data from the “Outside World” to a program; this is usually done through the keyboard, mouse clicks, a file, and others.
- **Processing instructions** - used for manipulating data inside the computer. These instructions allow us to add, subtract, multiply, and divide; they also allow us to compare two values, and act according to the result of the comparison. Also, we can move data from one location in the computer’s memory to another location.
- **Output instructions** - used for getting information out of the computer to the outside world.

# Developing the Algorithm

Using the notion of an algorithm and the concept of a variable, we develop the following algorithm for calculating the area of a square, given one side: Algorithm for calculating area of square, given one side:


1. Start
2. Ask the user for the length of a side.
3. Store the value in the box *s*.
4. Calculate the area of the square ( $s \times s$ ).
5. Store the area in the box *a*.
6. Print the value in box *a*, appropriately labeled.
7. Stop.

# Dry running/Desk checking

- It is the process of checking the algorithm to ensure that it works accordingly. We test the algorithm by “playing computer”, that is, we execute the instructions by hand, using appropriate data values.
- It is important to do desk checking to pinpoint logical errors before starting the actual coding.
- We should never start coding unless we are confident that the algorithm is correct.

## 4. Write the computer program that implements the Algorithm

You already have an algorithm using English statements. However, these statements are sufficiently “computer-oriented” for a computer program to be written directly from them. Before we do this, let us see how we expect the program to work from the user’s point of view.



## 4. Write the computer program that implements the Algorithm

In order to write the computer program from the algorithm, a suitable programming language must be chosen.

# Algorithm vs Program

The difference between an algorithm and a program is that the algorithm can be written using informal language without having to follow any special rules, whereas, a program is written in a programming language and must follow all the rules of the language.

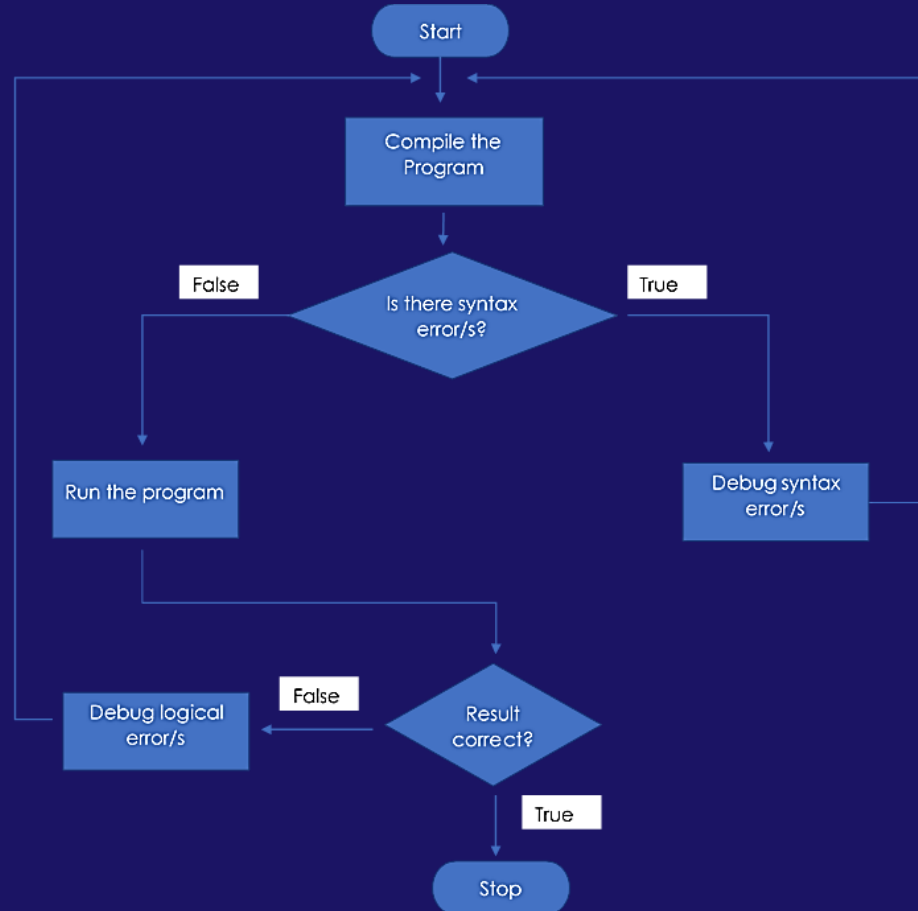




## 5. Test and Debug the Program

Testing and debugging are a crucial part of the programming process as it is where you will assess if the program is doing its intended job.

# Testing and Debugging Flowchart





## 5. Test and Debug the Program

When you compile the program remember that the computer can execute instructions written in machine language only. The source code (C program codes) must be converted to object code or machine code. The program that does this job is called the compiler.

## 5. Test and Debug the Program

Two types of errors:

- **Syntax errors** - errors that arise from breaking the rules in writing statements in the language.
- **Logical errors** - errors that causes a program to give incorrect results for valid data. This may cause program to crash.

## 5. Test and Debug the Program

- When you run the program, test data are provided to check if the result of the program matches the expected value or output.
- If the program contains logical errors, you need to debug the program. Find and correct any errors that are causing the program to produce incorrect results.



## 6. Document the Program

The final job is to complete the documentation of the program. One purpose of the program documentation is it will be a guide for the maintenance and future upgrade of the program.

## 6. Document the Program

The documentation includes the following:

- The statement of the problem
- The algorithm for solving the problem
- The program listing
- Test data and the results produced by the program

## 6. Document the Program

### Types of Program Documentation

- **Technical Documentation** - this is the documentation useful for a programmer, perhaps for future modification or upgrade of the program.
- **User Documentation** - this enables a non-technical person to use the program without needing to know about the internal workings of the program.



## 7. Maintain the Program

Programs are normally meant to be used over a long period of time. During this time, errors may be discovered that previously went unnoticed. Errors may also surface because of conditions or data that never arose before. Whatever the reason, such errors must be corrected.

## 7. Maintain the Program

A program may need to be modified to adapt to the changes needed by the user. Assumptions where made during the development of the program and have now changed due to different reasons such as change in company policy or even due to change in government regulations, or perhaps the company is changing its computer system and the program needs to be “migrated” to the new system.

## 7. Maintain the Program

Whether the maintenance is easy or not depends on how the original program was written. If it was well-designed and properly documented, then the job of the maintenance programmer would be easier.

# How a Computer Executes a Program

- Recall that a computer can execute a program written in machine language only. For the computer to execute the instructions of such a program, those instructions must be loaded into the computer's memory (also called primary storage).

memory
instruction 1
instruction 2
instruction 3
etc.

# How a Computer Executes a Program

- You can think of memory as a series of storage locations, numbered consecutively starting at 0. Thus, you can speak of memory location 27 or memory location 31548. The number associated with a memory location is called its **address**.
- A computer runs a program by executing its first instruction, then the second, then the third, and so on. It is possible that one instruction might say to jump over several instructions to a particular one and continue executing from there. Another might say to go back to a previous instruction and execute it again.



Wrap-up



# Programming Process

is a series of activities used by developers to create a quality program from start to finish.

# 7-steps of the Programming Process

1. Define the problem.
2. Analyze the problem.
3. Develop an algorithm (a method) for solving the problem.
4. Write the computer program that implements the algorithm.
5. Test and debug (find the errors in) the program.
6. Document the program. (Explain how the program works and how to use it.)
7. Maintain the program.



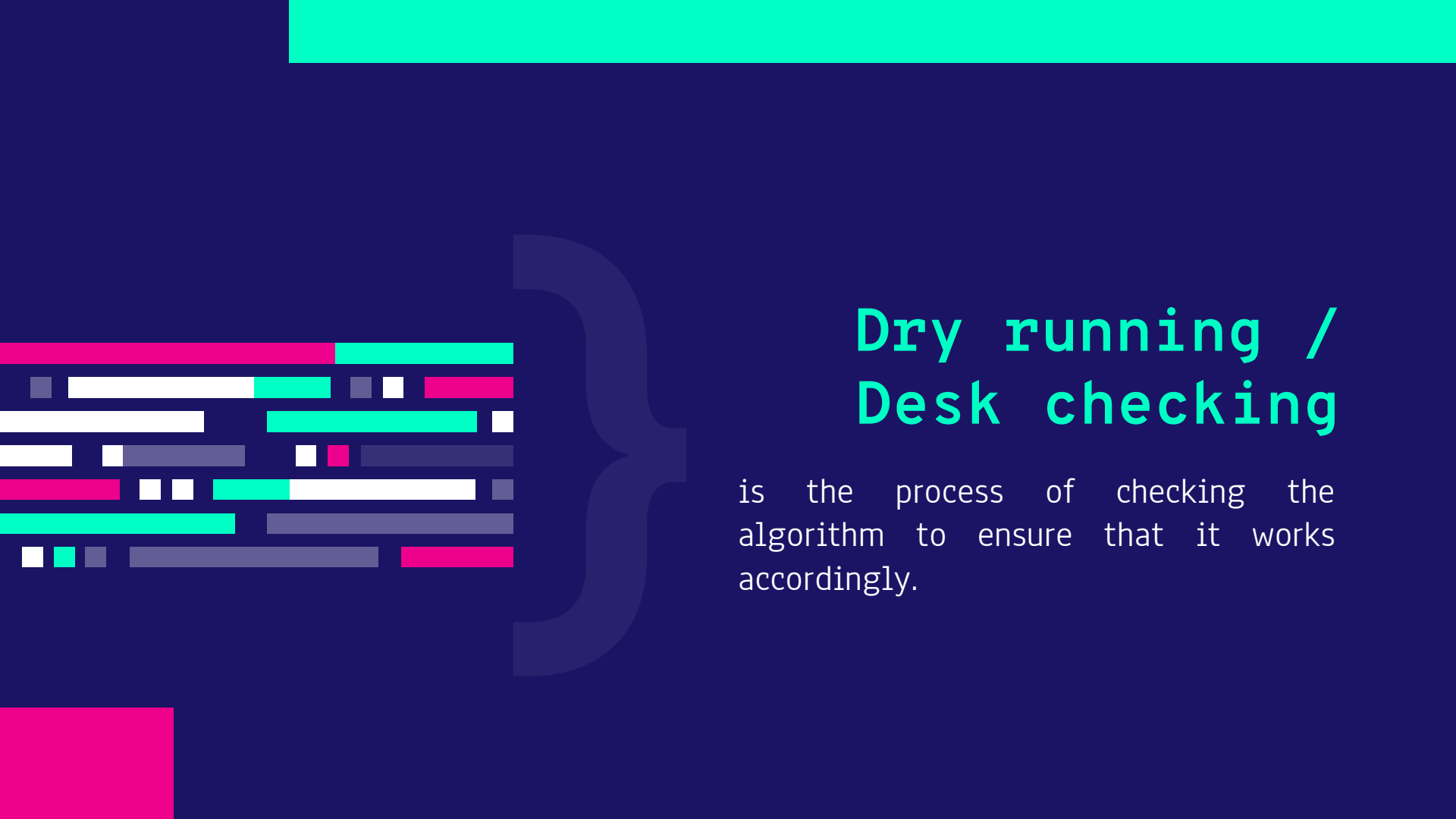


# Algorithm

It is a set of instructions that, if faithfully followed, will produce a solution to a given problem or perform some specified task.

# Three Main Categories of Computer Instructions

- Input instructions
- Processing instructions
- Output instructions



## Dry running / Desk checking

is the process of checking the algorithm to ensure that it works accordingly.

# Two Types of Errors

- Syntax errors - errors that arise from breaking the rules in writing statements in the language.
- Logical errors - errors that causes a program to give incorrect results for valid data. This may cause a program to crash (come to abrupt halt).

# The documentation includes

- The statement of the problem
- The algorithm for solving the problem
- The program listing
- Test data and the results produced by the program

# Two Types of Program Documentation

- **Technical documentation** – this is the documentation useful for a programmer, perhaps for future modification or upgrade of the program.
- **User documentation** – this enables a non-technical person to use the program without needing to know about the internal workings of the program. The user needs to know how to load the program, and how to use the various features of the program. Also, the user needs to know how to handle unusual situations that may arise while the program is being used.



# Activities

# THANKS!



CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, and infographics & images by Freepik.



# RESOURCES

Kalicharan, N. (2015). *Learn to Program with C*. New York: Apress Media.