Learn C Programming from Scratch

A step-by-step methodology with problem solving approach

Mohammad Saleem Mir

www.bpbonline.com

Copyright © 2024 BPB Online

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor BPB Online or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

BPB Online has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, BPB Online cannot guarantee the accuracy of this information.

First published: 2024

Published by BPB Online
WeWork
119 Marylebone Road
London NW1 5PU

UK | UAE | INDIA | SINGAPORE

ISBN 978-93-55516-060

www.bpbonline.com

Kup książkę

Dedicated to

In recognition of your unwavering support, boundless encouragement, and enduring belief in my dreams, this book on C programming is dedicated to the pillars of my life – my beloved parents and cherished family members. Your love and guidance have been the foundation upon which I've built my journey in the world of programming.

With heartfelt gratitude and endless love,

Mohammad Saleem Mir

Kup książkę

About the Author

Mohammad Saleem Mir is currently working as Senior Assistant Professor at Higher Education Department, J&K, India. He has more than 15 years of teaching experience. During the course of his journey as a Teacher, he has taught at Post graduate as well as under-graduate levels. He has taught in general all the subjects associated to computer Sciences but took lead when it came to guiding students through Programming subjects, with the aim of building strong analytical skills and inculcate logical and reasoning skills into the students. The author holds bachelor's degree in Sciences, Master's Degree in Computer Applications/ Sciences from Kashmir University besides being a Doctorate student. He is currently pursuing Ph. D in the fields of Machine Learning/ Artificial Intelligence/ Medical Sciences. He has published a number of research papers in national as well as international Journals, besides presenting his research in various conference. The author has delivered inspirational talks in various conferences/ Refresher courses and various Training Programmes conducted by various Universities and colleges. The author remains actively involved in technical seminars and workshops.

Kup książkę

About the Reviewer

Srividya Richard, Sri is a passionate and a dedicated educator with a Masters of Engineering in Computer Science from Anna University. She considers herself as a self-confident and an enthusiastic person with a flair to logically conclude any given task with complete precision and accuracy. She is extremely goal oriented and an excellent team player.

She has a decade long experience in teaching Engineering graduates at a reputed institution. Her research interests lie in the area of Machine learning, Data Analytics, Artificial Intelligence. She has published many research articles in her areas of interests in refereed journals and conferences. Sri's love for technology and the English language has contributed immensely in reviewing and proofreading several articles that ranges from research thesis to books. During her academic career, Sri had the opportunity to mentor and execute several real time IT projects. She has also conducted various trainings and workshops and served as a committee member in organizing conferences and research seminars. The exposure to interdisciplinary projects and industry interactions has had a significant impact in honing her skills.

As an AI enthusiast, Sri is delighted to embark on the path of equipping professionals with AI skills. She looks forward to leveraging the expertise gained over the years to empower individuals for both personal and professional development.

Kup książkę

Acknowledgement

This book is my first experience to pen down something for my esteemed readers. I sincerely hope they would appreciate it and let me know an honest review of the book. I would like to acknowledge with gratitude, the support and love of my family. They all kept me going, and conceiving this book would not have been possible without their support and encouragement.

I am also grateful to BPB Publications for their guidance and expertise in bringing this book to fruition. It was a long journey of revising this book, with valuable participation and collaboration of reviewers, technical experts, and editors.

Finally, I would like to thank all the readers who have taken an interest in my book and for their support in making it a reality. Your encouragement has been invaluable.

Kup książkę

Preface

I feel grateful to share my knowledge, analyses, and conclusions to raise the level of curiosity with regards to programming among the students of computer sciences, at the end of the day, we all will be beneficiaries. When one chooses computer sciences as a career, it may be noted that programing is an indispensable part of computer sciences and one cannot be a good computer scientist unless (s)he is good (if not exceptional) programmer. Well begun is half done, as they famously say; beginning on a good note, knowing fundamentals of programming clearly and building a solid base as far taking up more advanced form of programming, later in your career, is concerned is important. For this very, reason, this book focuses on giving an insight into the fundamentals of programming beforehand, and a sincere effort has been made to cover major concepts in detail so that the readers are kept interested.

C is a foundational programming language and ought to be learned systematically. Getting measure of the features of C programming is very important. This book covers all the fundamental concepts, rest assured the readers would have a great time going through the book contents and would benefit from this book. The book is written in unpretentious manner. The target audience being programming beginners, who would be able to understand the concepts explained in the book quite easily. Each concept in the book is synchronously supplemented by coding examples to enhance clarity. The book has the following 10 chapters:

Chapter 1: Programming Methodology – In this Introduces you to the world of Programming Methodology, Problem Solving, Program Design etc.

Chapter 2: C Programming Fundamentals – This chapter would acquaint you to the basic structure of the C Programming language, coding environment, and the syntax and references to write C Programs.

Chapter 3: Control Statements – This chapter throws light on various control statements used in C programming.

Chapter 4: Functions – This chapter gives an account of perhaps the most important topic in programming languages that is, functions.

Chapter 5: Arrays – In this chapter, we will introduce arrays, multi-dimensional arrays, strings and more how we can work with them.

Kup książkę

Chapter 6: Pointers – In this chapter, we will learn about pointers, a concept that is found difficult by the programming novices, but not anymore.

Chapter 7: Structures and Unions – In this chapter, we will learn to work with user defined data types like Structures and Unions.

Chapter 8 File Handling – In this chapter, we will learn about how to work with hard disk files and different associated operations with programming examples.

Chapter 9: C Preprocessors – In this chapter we will learn how to define and use Preprocessors in C and their importance.

Chapter 10: C Graphics – In this chapter we will see the other side of the C Output, first one being the Text mode. Graphics programming in C used to drawing various geometrical shapes using inbuilt C functions.

Kup książkę

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/9m5lv1u

The code bundle for the book is also hosted on GitHub at
https://github.com/bpbpublications/Learn-C-Programming-from-Scratch.
In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Did you know that BPB offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.bpbonline.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at :

business@bpbonline.com for more details.

At **www.bpbonline.com**, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on BPB books and eBooks.

Kup książkę

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Kup książkę

Table of Contents

1. Programming Methodology 1
Introduction 1
Structure 1
Objectives 2
Programming methodology 2
Problem-solving methodology and techniques 3
Features of a good program 4
Computer as a problem-solving tool 5
Computers and computing systems 6
Computer system 8
Hardware 9
Software 10
Compiler 11
Assembler 12
Debugger 12
Interpreter 13
Language translators 13
Programming styles 13
Procedural programming 14
Modular programming 14
Top-down modular programming 15
Bottom-up modular programming 16
Structured programming 18
Object Oriented Programming 19
Algorithms 20
Features of an algorithm 20
Writing an algorithm 20

Kup książkę

Algorithm efficiency 25
Importance of algorithmic efficiency 25
Algorithm time complexity cases 25
Implementation challenges 26
Flowcharts 27
History of flowcharts 27
Flowchart symbols 27
Meaning of a flowchart 28
Flowchart examples 29
Conclusion 36
Exercise 37

2. C Programming Fundamentals 39
Introduction 39
Structure 40
Objectives 40
Facts about C 40
Uses of C programming language 41
Life cycle of a C program 41
Integrated Development Environment 43
GCC 43
Visual Studio Code 44
Turbo C++ for Windows 10+ 45
Eclipse 47
Dev-C++ 48
Creating a source code 48
Save file 49
Compile code 49
Execute/ Run code 50
C program components 50
Tokens in C 52

Kup książkę

Semicolons ... 53
Comments ... 53
Keywords ... 53
Whitespace in C ... 53
Operators ... 54
Arithmetic operators ... 54
Relational operators ... 55
Logical operators ... 56
Bitwise operators ... 56
Assignment operators ... 58
Operator precedence in C ... 60
Data types ... 62
Variables ... 64
Variable declaration ... 64
Guidelines for declaring variables ... 65
Initialization ... 66
Identifiers ... 67
Difference between variables and identifiers ... 68
Constants ... 68
Integer data type ... 69
Integer expressions ... 70
Precedence of operators ... 70
Input-output in C ... 71
Output formatting ... 72
printf () ... 72
Control characters ... 73
Read data supplied by a user ... 74
Read data into a float or double variable ... 76
putchar () ... 78
fgtes () and fputs () ... 79

Kup książkę

Print an integer using a field width ... 79
Floating point types ... 81
Print double and float variables ... 82
Floating-point expressions ... 83
Assigning double/float to int ... 83
Char data type ... 83
void type ... 84
Lvalues and Rvalues in C ... 85
Literals ... 86
Integer literals ... 86
Floating-point literals ... 87
Character constants ... 88
String literals ... 88
Defining constants ... 89
Differences between constants and literals ... 90
Storage class ... 91
Auto ... 91
Register ... 91
Extern ... 92
Static ... 92
Extern storage class ... 94
Writing basic programs in C ... 94
Sequential logic ... 95
Conclusion ... 98
Exercise ... 98

3. Control Statements ... 101
Introduction ... 101
Structure ... 101
Objectives ... 102
Control statements ... 102

Kup książkę

Decision making statements *103*
if statement *103*
if...else statement *104*
The if-else-if ladder *106*
Nested if statements *109*
Nested conditions *114*
Switch statements *119*
Loop statements *126*
Constituents of a loop *126*
Different loops used in C *127*
While loop *127*
do...while loop *131*
For loop *133*
Loop control statements *136*
Break *136*
Continue *137*
Example: *137*
Goto *138*
Infinite loop *139*
Nested loop *140*
Conclusion 144
Exercise 145

4. Functions 149
Introduction 149
Structure 149
Objectives 150
Functions 150
Key features of functions in C *151*
Types of functions in C *153*
Defining a function *154*

Kup książkę

Calling a function *155*
Formal arguments *156*
Return statement *156*
Various 'return' scenarios in C *157*
Function signature *158*
Mismatch *161*
Function variable scope *162*
Local variables *163*
Function parameters or variables *163*
Global variables *164*
Initializing local variable and global variables *165*
Callback function *165*
Function as an argument *166*
Recursion *168*
Memory usage in recursion *168*
Conclusion 173
Exercise 174

5. Arrays 177
Introduction 177
Structure 177
Objectives 178
Arrays: One-dimensional array 178
Declaring and initializing arrays 178
Declaring arrays *178*
Initializing arrays *179*
Bounds and array size 180
Base address 180
Array manipulation 181
Accessing array elements 181
Modifying array elements 182

Kup książkę

Passing arrays to functions 182
Some array-based programs 184
Character arrays - strings 197
Accessing characters in a string *198*
Iterate through characters of a string *198*
String input and output *199*
Read a line of text *200*
String manipulation *201*
String concatenation *201*
Substring extraction *202*
Searching and replacing *203*
Parsing and tokenization 205
Case conversion 205
String formatting *206*
String length and manipulation functions *206*
2-D strings 217
Declaration and initialization of 2-D strings *218*
Reading input into a 2D string *218*
Accessing characters in a 2-D string *219*
Matrices 220
Declaring 2-D array (Matrix) *221*
Initializing two-dimensional arrays *222*
2-dimensional array elements access *224*
Passing 2-D matrix to a function in C *225*
Conclusion 237
Exercise 237

6. Pointers 241
Introduction 241
Structure 241
Objectives 242

Kup książkę

Pointers 242
Using pointers 242
Declaring a pointer 242
Accessing the value of a pointer 243
Application of pointers 246
Direct memory manipulation 246
Dynamic memory allocation 246
Resizing and releasing memory 249
String pointers 252
Passing parameters to functions 254
Effective array manipulation 256
String manipulation 259
Pointer arithmetic 260
Pointer comparisons 263
Return pointer from functions 265
Pointer to pointer 267
Types of pointers 269
NULL pointers 269
Dangling pointers 270
Void pointers 270
Wild pointers 271
Array of pointers 272
Conclusion 273
Exercise 273

7. Structures and Unions 277
Introduction 277
Structure 277
Objectives 278
Structures and unions 278
User defined data types 279

Kup książkę

Structures 279
Enumerations 279
Structures 281
Declaring a structure 282
Accessing structure members 283
Access structure members through a pointer 284
Using arrow (->) operator to access structure members 285
Array of structures 286
Sort array of structures 288
Structures as function arguments 290
Nested structures 292
Accessing nested structure 292
Embedded structure 293
Unions in C 294
Syntax of unions 294
Usage of unions 295
Considerations and limitations 295
Examples of union usage 296
Defining a union 296
Accessing union members 298
Error handling and validation in structures/unions 300
Handling memory allocation errors 300
Member initialization 300
Input validation 301
Union discriminant handling 301
Error reporting and exiting 302
Graceful decline 302
Graceful recovery 302
Conclusion 303
Exercises 303

Kup książkę

8. File Handling 307
Introduction 307
Structure 308
Objectives 308
Introducing file handling 308
Data hierarchy - layers of data 309
Bit 309
Byte 310
Field 310
Record 310
File 310
Database 311
Data warehouses 311
Data mart 311
Files 311
Importance of files 312
File modes 312
File operations 313
Opening a file 314
Closing a file 314
Writing to a file 315
Reading from a file 317
File positioning 319
Error handling 321
File truncation 321
File deletion 321
File locking 321
Type of files 322
Working with text files in C 323
Opening a text file 323

Kup książkę

Reading text from a file 323
Writing text to a file 323
Appending text to an existing file 324
Closing a text file 324
Error handling 324
Reading and writing line-by-line 324
Text file manipulation 324
Working with CSV files 325
Binary file handling in C 325
Opening and closing binary files 326
Reading binary data 326
Writing binary data 326
Moving the file pointer 327
Binary file size 327
Binary file error handling 328
Working with complex data structures 328
Considerations for binary file handling 328
File access 328
Sequential file access 329
Opening a file for sequential access 329
Reading from a file 330
Writing to a file 330
Moving the file pointer 330
Closing a file 330
Advantages 330
Limitations 331
Program examples 331
Random access files 337
File position indicator 338
Opening random access files 338

Kup książkę

Writing data 338
Reading data 338
Advantages of random-access files 339
Examples of random-access files in C 339
Programs on random access files in C 340
Direct file access 351
Programs on direct access files in C 351
File descriptors 355
Opening files for direct access 355
Reading from files 355
Writing to files 356
File positioning 356
Closing files 357
Some program examples on file descriptors 357
Stream file access 359
Benefits 359
Error handling in files 360
Global variable errno 360
Errno example 360
Conclusion 361
Exercise 362

9. C Preprocessors 365
Introduction 365
Structure 365
Objectives 366
Introduction to preprocessors 366
 # define directive 367
Defining a constant macro 368
Defining a function-like macro 368
Defining a conditional macro 368

Kup książkę

Redefining a macro *368*
 # ifdef preprocessor *369*
 # ifndef preprocessor *370*
 # include preprocessor *370*
 # undef preprocessor *372*
 # if # else # elif # endif macros in C *374*
Control structure macros *376*
Types of pre-processor directives *378*
File inclusion directives *378*
Macro definition directives *378*
Applications of macros *380*
Conditional compilation *383*
Preprocessor directive # Pragma *385*
ANSI C predefined macros *390*
Preprocessor operators *392*
Macro continuation (\) operator *392*
Stringize (#) operator *392*
Token pasting (##) operator *393*
Defined () operator *393*
C-header files *393*
Include syntax *394*
Include operation *394*
Once-only headers *395*
Using # ifdef for different computer types (Computed includes) *395*
Conclusion 396
Exercise 396

10. C Graphics 399
Introduction 399
Structure 399
Objectives 400

Kup książkę

C Graphics ... 400
C libraries for graphics ... *401*
Choosing the right graphics library ... *401*
Configuring the graphics environment ... *402*
Initialization of the graphics mode ... *402*
Setting up Borland Graphics Interface for Windows ... *402*
Get and set up WinBGIm ... *402*
Setting up your compiler ... *402*
Add the necessary header file ... *402*
Initialize the graphics mode ... *402*
Drawing on the screen ... *403*
Closing the graphics mode ... *404*
Setting up graphics mode using SDL ... *404*
Understanding coordinates and resolution ... *404*
Basics of color ... *405*
Operations for simple drawings ... *405*
Colors in C graphics programming ... 409
C Program on graphics colors ... *410*
Input handling ... 411
Animations and delay ... 411
Double buffering ... *412*
Text mode graphics functions ... 412
C graphics program to draw shapes ... 414
C graphics programs ... 417
Conclusion ... 420
Exercise ... 421

Index ... 423-432

Kup książkę

Chapter 1
Programming Methodology

Introduction

In this chapter, we will explore the fascinating world of computers and computing, gaining a comprehensive understanding of their principles and capabilities. Our primary focus will be on programming as we delve into various programming styles and methodologies. We will also explore the inner workings of computer systems, understanding their components and how they function together. Along the way, we will develop a strong foundation in algorithms, learning how to design and implement them effectively. To aid our understanding, we will utilize flowcharts as visual tools for representing and analyzing algorithms. By the end of this chapter, you will possess a deep knowledge of programming principles, problem-solving techniques, and the powerful role that computers play in solving real-world challenges. Let us embark on this exciting journey of exploration and discovery.

Structure

The chapter discusses the following topics:

- Programming methodology
- Computer as a problem-solving tool
- Computers and computing

Kup książkę

- Computer system
- Programming styles
- Algorithms
- Flowcharts

Objectives

The objective of this chapter is to provide a comprehensive understanding of programming methodology, emphasizing the use of computers as problem-solving tools. It explores the concepts of computers and computing, delves into the components and functioning of computer systems, and explores various programming styles. The chapter aims to develop a solid foundation in algorithms, their design, and implementation. Additionally, it covers the use of flowcharts as visual tools for algorithm representation and analysis. By the end of the chapter, readers will have gained a deep knowledge of programming principles, problem-solving techniques, and effective use of computers in solving real-world challenges.

Programming methodology

Programming methodology refers to the systematic approach and techniques used in software development. It provides a framework for software engineers to follow while designing, coding, testing, and maintaining software applications.

A computer program is a set of related commands or instructions to solve a given computer problem. Programming is the process of conceptualizing and subsequently writing a computer program that is executed to produce the desired result. Programming involves conceptualizing and writing a program, compiling the program, executing the program, and debugging the program. A computer program aims to solve real-life problems ranging from simple to more complex ones. Programming methodology is the method to analyze real-life complex problems, provide software solutions, and control the associated activities of the software development process. Some popular programming methodologies include:

- **Waterfall methodology**: This is a sequential software development model that follows a set of steps in a linear fashion, each step building upon the previous one.
- **Agile methodology**: This iterative approach to software development emphasizes collaboration between developers and stakeholders, flexible planning, and adaptive delivery.
- **Scrum methodology**: This agile methodology is designed specifically for software development projects. It focuses on iterative, incremental delivery of working software, emphasizing teamwork, flexibility, and continuous improvement.

Kup książkę

- **DevOps methodology**: This software development approach emphasizes collaboration between development and operations teams to ensure that applications are delivered quickly, reliably, and securely.

Each of these methodologies has its own advantages and disadvantages, and the choice of methodology depends on the specific needs and constraints of the software development project. The choice of the most appropriate methodology helps the software development team deliver a high-quality product that meets the needs of its users.

Problem-solving methodology and techniques

Computer problem-solving methodology is a systematic approach to identify, analyze, and resolve computer-related issues.

The steps involved in this methodology are:

1. **Define the problem**: Identify the issue and gather information about it, including error messages, symptoms, and the context in which the problem occurred.
2. **Analyze the problem:** Use critical thinking and troubleshooting techniques to understand the root cause of the issue.
3. **Formulate a hypothesis**: Develop a potential solution based on the information gathered in the previous step.
4. **Test the hypothesis**: Implement the proposed solution and check if it resolves the problem.
5. **Evaluate the results**: If the solution worked, document it and move on to the next problem. If it did not work, go back to *Step 3* and formulate a new hypothesis.
6. **Implement a permanent solution**: If a solution is found, implement it permanently to prevent future problems.
7. **Document the solution**: Document the solution, including the steps taken, the outcome, and any lessons learned, to help with future troubleshooting.

By following this methodology, computer problems can be solved efficiently and effectively.

Writing a good program is an expertise; hence, it needs to follow certain practices:

- **Meaningful names for identifiers**: Writing understandable, legible, and maintainable code in programming requires giving identifiers (such as variables, functions, classes, and more) meaningful names. Here are some suggestions for selecting identifiers with meaningful names:
    - **Assigning identifiers meaningful names:** We should assign identifiers meaningful names that appropriately reflect their function and intended use. Use descriptive names like **`num_students`** or **`current_date`** instead of ambiguous or general ones like **`x`** or **`temp`**.

Kup książkę

- **Consistency in naming**: We should maintain consistency by using the same naming patterns across our source code.
- **Steer clear of acronyms**: They might make your code more difficult to read and comprehend. If an acronym is not well-known and frequently used in your programming community, do not use it.
- **Name classes and objects with nouns**: Classes and objects should be named with nouns since they reflect actual objects. For instance, `Student` should be the class name for a class representing a student.
- **Name functions and methods using verbs**: Verbs describe activities that may be taken; hence, they should be used to name functions and methods. For instance, the function `Sum` should compute the sum of two values.
- **Clarity of expression:** Expression represents a specified operation. Hence, they need to be clearly understood and depicted with no concession on clarity of expression.
- **Comments and indentations:** Comments are used for documentation purposes in a program. Comments (denoted by `//` for a single line comment) tend to guide a programmer through debugging; indentation is the ideal technique for writing a program in such a way that helps to understand the flow of a program.
- **Use of blank lines or blank spaces:** Blank lines are used to separate blocks of code that are long. The standard for the use of spaces in programming languages matches what we follow in normal English rules, meaning that symbols in C, like `=`, `+`, and more, should precede and follow with at least one space.
- **Depiction of statements**: Statements represent meaningful commands through which we perform operations in C. They should be put on separate lines. In case of a block of statements (explained later), which is denoted by `{`and`}`, the opening brace, `{`should appear on the line after the block-level statement, and the closing brace`}`, should appear after the last statement of the block-level statement. A block of statements is indented with relation to the braces.

Features of a good program

A good computer program should have the following features:

- **Usability**: The program should be easy to use and understand, with a clear and intuitive user interface.
- **Reliability**: The program should function as expected and not cause unintended consequences or errors.

Kup książkę

- **Performance**: The program should be fast and responsive without significant lag or delay.
- **Scalability**: The program should be able to handle increasing demands and data processing requirements as the user base grows.
- **Security**: The program should have robust security measures to protect against hacking, data breaches, and other security threats.
- **Flexibility**: The program should allow users to customize and modify their settings and preferences to fit their needs.
- **Compatibility**: The program should work seamlessly with other software and hardware components without compatibility issues.
- **Support**: The program should have adequate documentation, training, and technical support to help users resolve any issues.

By having these features, a good computer program will provide a positive user experience, increase productivity, and provide long-term value to its users.

Reasons for using a structured, systematic approach to problem-solving:

- To ensure consistency
- To make everyone aware of how others are approaching the project.
- To keep the procedure objective and resistant to personal prejudices and impressions.
- To facilitate group decision-making
- To help a group follow the model rather than using different ways all at once by focusing on the six phases in the problem-solving model, which also helps define the agenda. A group can more easily agree by employing a process and facts to guide their judgments.
- To successfully resolve issues.

Computer as a problem-solving tool

Computers are information processing devices that transform information/data according to prescribed rules, organized in an *algorithm* implemented as a *program***.**

Algorithm: A precise, step-by-step method of doing a task in a finite amount of time.

How to use the computer as a problem-solving tool?

We need to:

- Analyze and describe the initial state and target stage of events with respect to input data.

Kup książkę