Paper 2 Theory

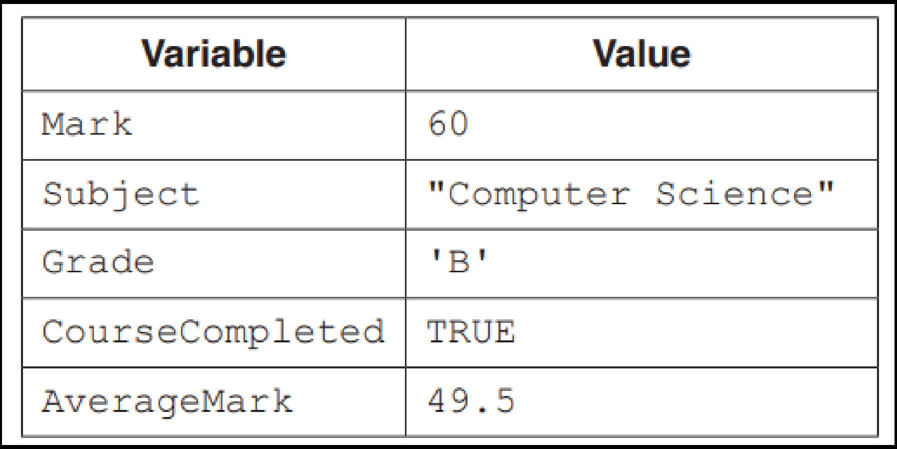
Papers Dock

COMPUTER SCIENCE 9618 PAPER 2

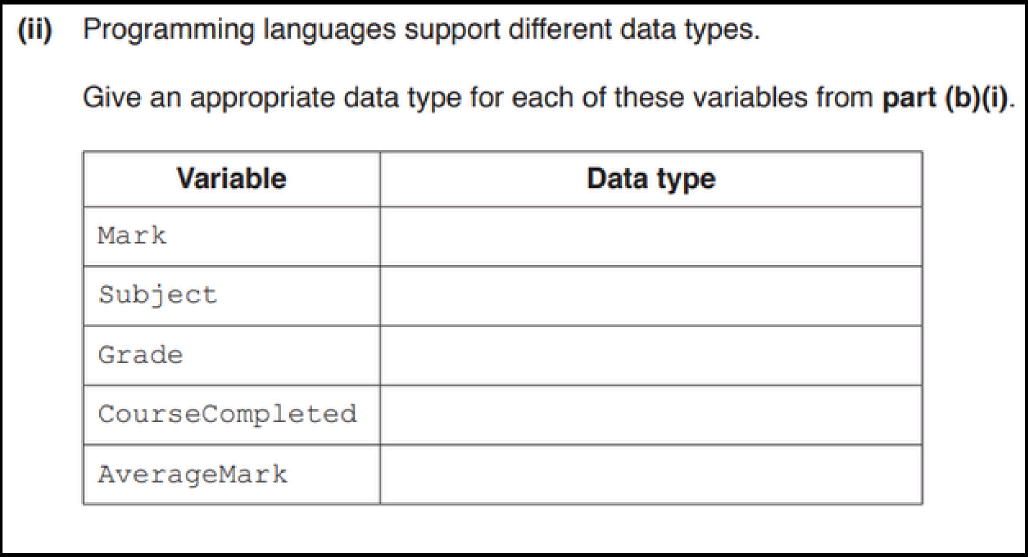
Paper 2 Theory

DATATYPES

1. INTEGER: (0-9) without decimal e.g. 95, 94, 93, 21
2. REAL: (0-9) with decimal e.g. 95.1, 101.0, 9.3, 1.5
3. STRING: Alphabets, Numbers, Symbols
4. CHAR: Single character e.g. 'P', 'p', 'A'
5. BOOLEAN: 2 comparison, True, False



Variable	Value
`Mark`	`60`
`Subject`	`"Computer Science"`
`Grade`	`'B'`
`CourseCompleted`	`TRUE`
`AverageMark`	`49.5`



(ii) Programming languages support different data types.

Give an appropriate data type for each of these variables from **part (b)(i)**.

Variable	Data type
`Mark`	
`Subject`	
`Grade`	
`CourseCompleted`	
`AverageMark`	

Data Identifier Table

List Of Variable	Description	Datatype
ProjectCompleted	It will tell the status of the project (True or False)	Boolean

Question : What an "Identifier table" contains?

- **List of variable names**
- **Explanation of what that variable does**
- **Datatype of that variable**

Algorithm

- **Sequence of steps / instructions**
- **To implement a task / solution to a problem**

Stages Of Algorithm

Input	**INPUT N1, N2**
Process	Answer <-- N1 + N2
Output	OUTPUT Answer

Features Of Programming Languages

A software developer is writing a program and includes several features to make it easier to read and understand.

- Indentation
- Blank Lines
- Capitalization of keywords
- Sensible Variable Name
- Comments
- Pretty print

Source Code	It represents a solution/design/algorithm expressed in a High-level language
Object Code	• It is produced by the compiler during the translation stage. • The object code is produced by translating the source code. • Not produced by an interpreter.
Pseudocode	It is a way of using keywords and identifiers to describe an algorithm without following the syntax of a particular programming language
Flowchart	Pictorial or graphical representation of a program in a flow

Four Basic Types Of Constructs

1) Assignment

- A value is given a name (identifier) or the value associated with a given identifier is changed.

Explanation

2) Sequence

- A number of steps are performed one after the other.

Explanation

3) Selection

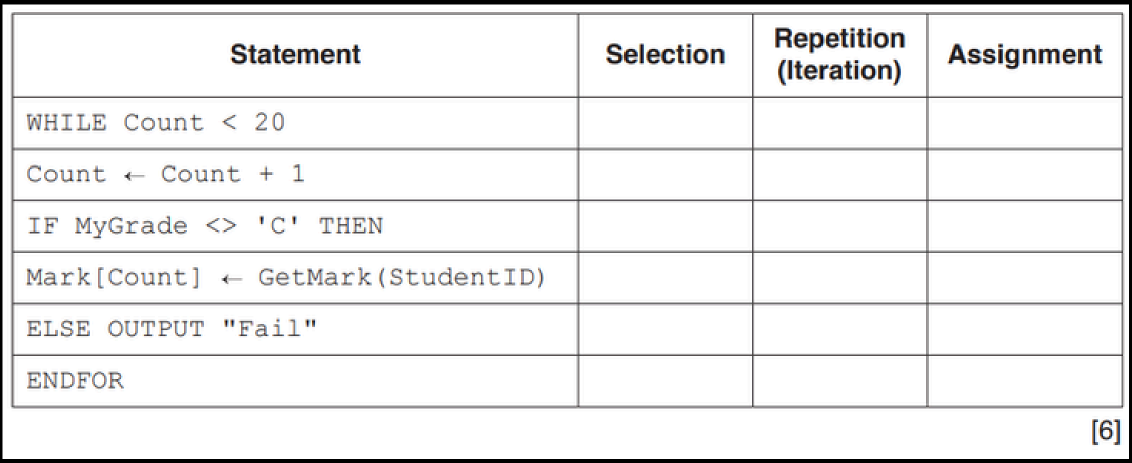
- Under certain conditions, some of the steps are performed, otherwise no or different steps are performed.

- IF ... THEN ... ELSE ENDIF
- CASE OF Otherwise ENDCASE

4) Repetitions / Iterations

- Under certain conditions, some of the steps are performed, otherwise no or different steps are performed.

- FOR … TO … NEXT (count-controlled loop)
- WHILE … DO … ENDWHILE (pre-conditional loop)
- REPEAT… UNTIL (post-conditional loop)



Statement	Selection	Repetition (Iteration)	Assignment
`WHILE Count < 20`			
`Count ← Count + 1`			
`IF MyGrade <> 'C' THEN`			
`Mark[Count] ← GetMark(StudentID)`			
`ELSE OUTPUT "Fail"`			
`ENDFOR`			

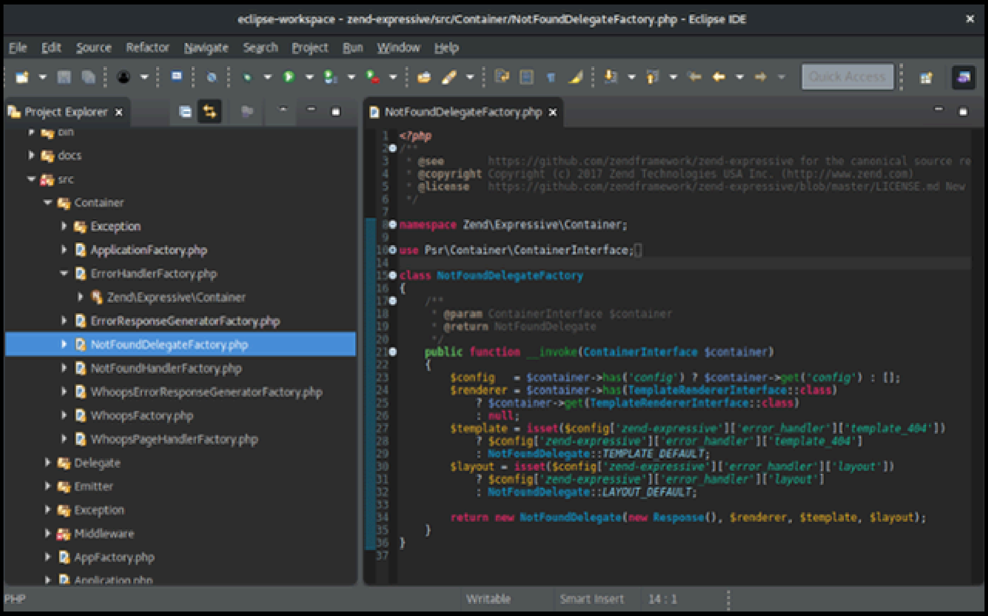
[6]

Identifier

Constant	Variable	Arrays
Fix Not Changeable	Temporary Memory	Temporary Memory

Constant	The value cannot be changed accidentally during the execution of the program.
Variable	Stores a value that can change during execution.
Arrays	A list of items of the same data type stored under a single name.

Integrated Development Environment (IDE)


eclipse-workspace - zend-expressive/src/Container/NotFoundDelegateFactory.php - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Project Explorer
bin
docs
src
Container
Exception
ApplicationFactory.php
ErrorHandlerFactory.php
Zend\Expressive\Container
ErrorResponseGeneratorFactory.php
NotFoundDelegateFactory.php
NotFoundHandlerFactory.php
WhoopsErrorResponseGeneratorFactory.php
WhoopsFactory.php
WhoopsPageHandlerFactory.php
Delegate
Emitter
Exception
Middleware
AppFactory.php
Application.php
NotFoundDelegateFactory.php
PHP
Writable
Smart Insert
14 : 1

```php
<?php
/**
 * @see       https://github.com/zendframework/zend-expressive for the canonical source re
 * @copyright Copyright (c) 2017 Zend Technologies USA Inc. (http://www.zend.com)
 * @license   https://github.com/zendframework/zend-expressive/blob/master/LICENSE.md New
 */

namespace Zend\Expressive\Container;

use Psr\Container\ContainerInterface;

class NotFoundDelegateFactory
{
    /**
     * @param ContainerInterface $container
     * @return NotFoundDelegate
     */
    public function __invoke(ContainerInterface $container)
    {
        $config   = $container->has('config') ? $container->get('config') : [];
        $renderer = $container->has(TemplateRendererInterface::class)
            ? $container->get(TemplateRendererInterface::class)
            : null;
        $template = isset($config['zend-expressive']['error_handler']['template_404'])
            ? $config['zend-expressive']['error_handler']['template_404']
            : NotFoundDelegate::TEMPLATE_DEFAULT;
        $layout = isset($config['zend-expressive']['error_handler']['layout'])
            ? $config['zend-expressive']['error_handler']['layout']
            : NotFoundDelegate::LAYOUT_DEFAULT;

        return new NotFoundDelegate(new Response(), $renderer, $template, $layout);
    }
}
```

Integrated development Environment is a software application that combines all of the features and tools needed by a software developer

Features Of An IDE

1) Pretty Printing

The pretty print feature in an IDE enhances code readability by applying color-coding which helps in identifying key terms

2) Context Sensitive Prompts

Displays predictions of the code being entered and helps to complete statements

3) Expand and Collapse Code Blocks

The Expand and Collapse Code Blocks feature in an IDE allows developers to manage code visibility by hiding or showing specific sections.

4) Dynamic Syntax checking

Underlines or highlights statements that do not meet the rules of the language

5) Single Stepping

Executes one line of the program and then stops and runs the code line by line

6) Breakpoints

To stop the code at set points to check values

7) Report Window

Outputs the contents of the variables and data structure

8) Variable Watch

Checks the content of variables at specific points

Identify features in writing of the program?

- pretty printing
- Auto-complete
- Auto-correct
- Context sensitive prompts
- Expand and collapse code blocks

Identify debugging tools that IDE can provide?

- Breakpoints
- Single stepping
- Report windows

Explain how a programmer can make use of a typical Integrated Development Environment (IDE) when writing and testing a program.?

Writing e.g.

- Enter code into an editor
- Pretty printing to identify key terms
- Context-sensitive prompts to help complete statements
- Expand and collapse code blocks
- Auto-complete to suggest what to type next
- Auto-formatting to indent code blocks
- Dynamic syntax checking

Testing e.g.

- Single stepping to run the code line by line
- Breakpoints to stop the code at set points to check values
- Report window to see how variables change

What are the features provided by an IDE that assist in "Initial Error Detection"

Dynamic Syntax Checking
Identification of Unused Variables
Type Checking

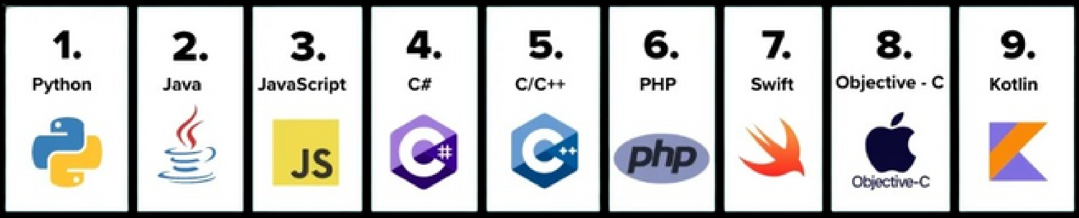
Features Of A Program

What are the features of a program?

- Function/Procedure
- Sequence
- Iteration
- Selection
- Parameters
- Variable
- Logic Operation

Transferrable Skills

There are certain skills that a person has in a certain language. Those can be used in another language



You should be able to recognize/understand in another language:

- Declaration
- Assignment
- Sequence
- Selection
- Repetition/Iteration
- Subroutines
- Parameters passed between modules
- Input/output

Exam Style Question

An algorithm is implemented in a high-level language. Changes are required, and the program is given to Albert, who is an experienced programmer, but he is not familiar with the language that has been used.

Explain why Albert would be able to understand the program?

- He would use his transferable skills to understand the new program.
- He could recognize basic control structures in the language, such as loops, selection statements, and declarations.
- He could read the comments/meaningful variable names.

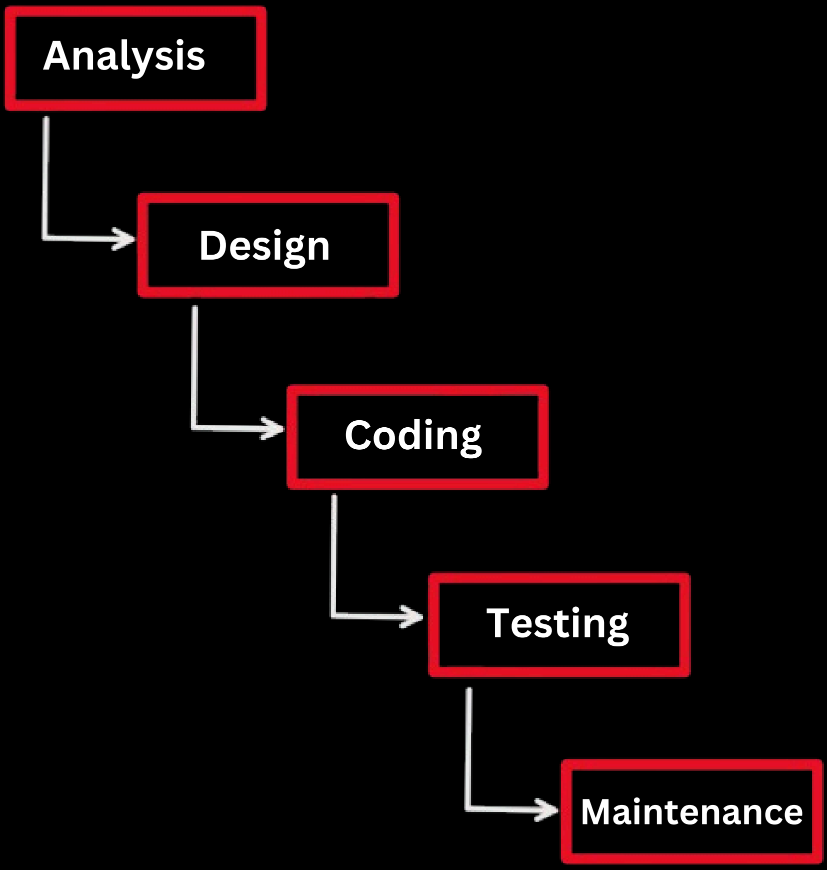
5 Stages In Program Development

Analysis Design Coding Testing Maintenance

Types Of Program Development Cycles

Waterfall Model Iterative Model Rapid Application Development

Waterfall Model



Principles

- Linear, as each stage is completed before the next is begun
- Well documented
- Low customer involvement
- (only at the start and end of the process)

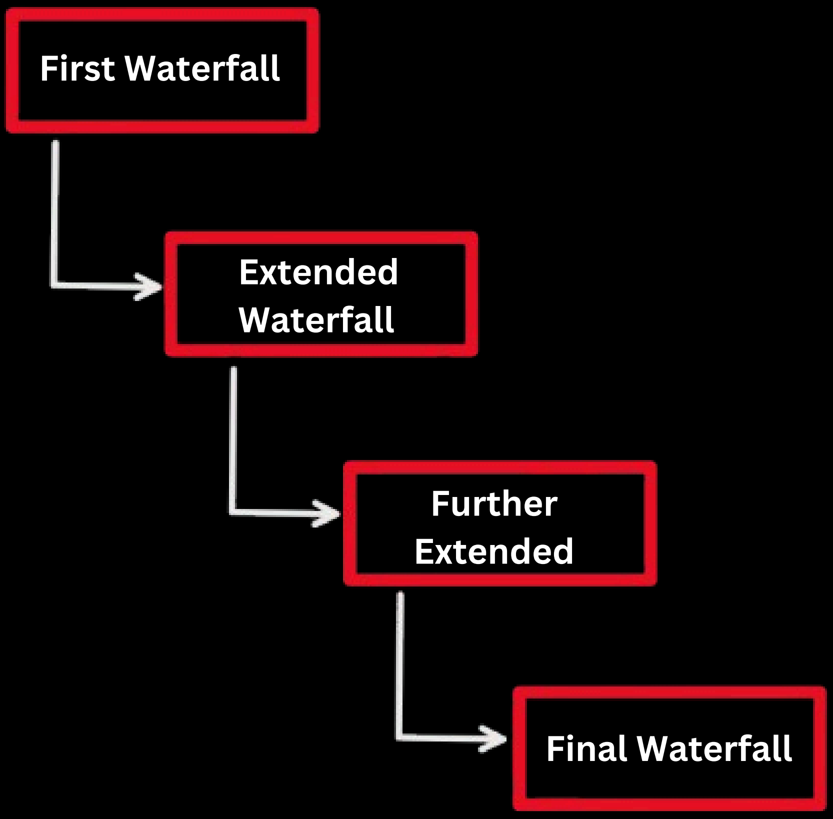
Benefits

- Easy to manage, understand, and use
- Stages do not overlap and are completed one at a time
- Works well for smaller programs

Drawbacks

- Difficult to change the requirement at a later stage
- Subject to change programs are not suitable
- Working program is produced late in the life cycle
- Not suitable for complex projects

Iterative Model



Principles

- Incremental development as the program development lifecycle is repeated.
- High customer involvement as part of the system can be shown to the customer after every iteration.

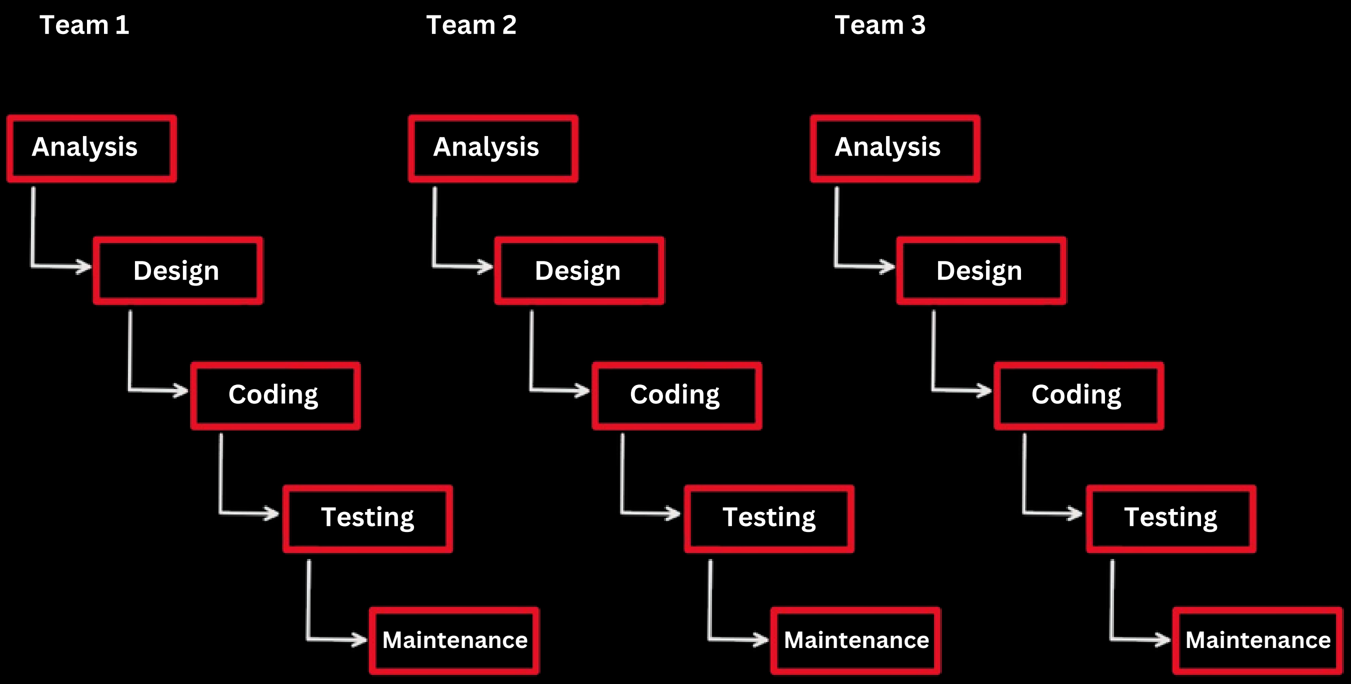
Benefits

- Easier to test and debug
- More flexible as easier to alter requirements
- Customers involved at each iteration, therefore no surprises when the final system is delivered.

Drawbacks

- Whole system needs to be defined at start
- Needs a good planning overall and at every stage.
- Not suitable for short simple projects

Rapid Application Development



- Program is divided into modules and each module has one team and its own waterfall.

Principles

- Minimal planning
- Uses previously written code where possible
- High customer involvement.

Benefits

- Reduced overall development time
- Rapid feedback informs the development.
- Very flexible as requirements evolve from feedback.
- Modification is easier because each part must work independently.

Drawbacks

- System under development needs to be modular
- Need strong team of skilled developers
- Not suitable for short-simple projects

Types Of Maintenance

Corrective	Perfective	Adaptive

Corrective Maintenance

- Maintenance occurs when testing reveals a fault (or error) in the program and this is corrected.
- Corrective maintenance is when errors are found and fixed.

What are the reason for using corrective maintenance?

- To amend the algorithms to eliminate errors.

Adaptive Maintenance

- It is a term used for changes that made to a program in response to a specification changes.
- As a result of changes to requirements or available technology.

What are the reason for using adaptive maintenance?

- In response to specification change arising from changes to business rules or environment.

Perfective Maintenance

- Perfective Maintenance mainly deals with implementing new or changed user requirements.
- Perfective Maintenance involves making functional enhancement.
- This includes enhancing both the function and efficiency of the code and changing the functionalities as per the user's changing needs.

What are the reason for using perfective maintenance?

- To enhance the functionality and efficiency of the program to meet the evolving needs of users or improve performance.

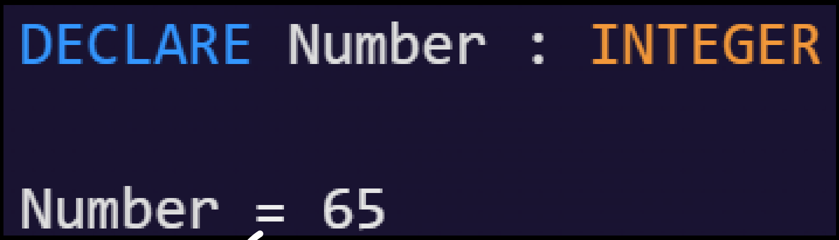
Types Of Errors

Syntax	Logical	Run-Time

Syntax Error

- Are small grammatical mistakes, sometimes limited to a single character.

For e.g.: A missing semi-colon at the end of a line or an extra bracket at the end of a function may produce a syntax error.



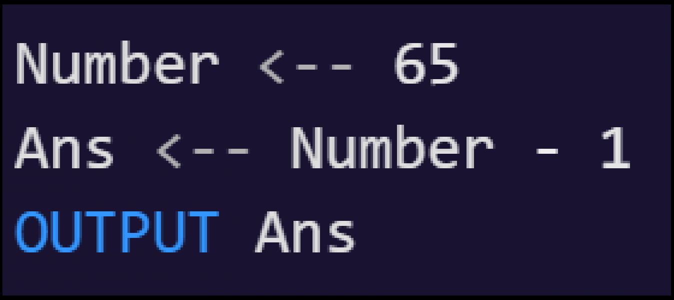
```
DECLARE Number : INTEGER

Number = 65
```

According to syntax arrow should be used rather than = so syntax error

Logic Error

- Program does not perform as expected
- Errors in the logic of a program

Question : Add 1 in 65

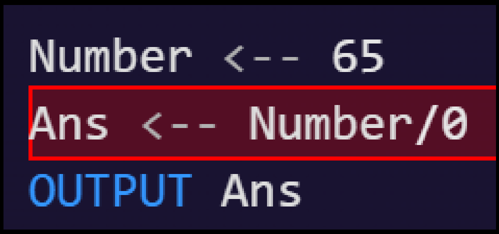


```
Number <-- 65
Ans <-- Number - 1
OUTPUT Ans
```

There is no error but the answer is not as expected

Run-Time Error

- Program executes an invalid instruction (out of bounds error)
- Attempts to divide by zero
- Program crashes.



```
Number <-- 65
Ans <-- Number/0
OUTPUT Ans
```

Types Of Test Data

Normal	Abnormal	Extreme	Boundary

1️⃣ Normal Data

- Within range (accepted)

2️⃣ Abnormal Data

- Out of the range (rejected)

3️⃣ Extreme

- In range, largest and smallest value which is accepted.

4️⃣ Boundary

- At edge, Accepted, Rejected.

Types Of Testing

Dry Run	Walkth rough	White -Box	Black- Box	Integr ation	Alpha	Beta	Accep tance	Stub

Black-Box Testing

- Match expected ans to actual ans
- A person does not need to know the structure of the code.
- A person chooses normal, boundary and erroneous data.

White-Box Testing

- Test every possible "Logic path" through the code
- A trace table is often used during the program testing.

Alpha Testing

- In house testing
- Testing is done by developers in-house other than the developer who actually built the software.

Beta Testing

- Out of house testing
- Performed by group of people those are assigned to find out bugs (errors) in the code.
- Testing performed by external users in a real-world environment to identify issues before the final release.

Integration Testing

- Testing combined modules or components to ensure they work together as expected.

Stub Testing

- Simple / dummy module written to stimulate the model.
- Test carried out all the modules (subroutines) have been written.

Acceptance Testing

- Is used for the completed program to prove to the customer / client that it works as required.
- End-user testing.

Dry Run

- Manually reviews code or logic without execution.
- Identifies logical errors or issues.
- Often uses a trace table to follow variable changes.

Walkthrough

- Informal review of design or code.
- Conducted with peers or team members.
- Focuses on feedback and collaborative improvement.