PSDD
TESTING, ERROR DETECTION AND CORRECTION

Contents

Testing

Testing and evaluating is the fourth stage of the Software Development Cycle, but testing is not just undertaken then but is undertaken **throughout** the development of a program, including during the algorithm development and coding stages. The more time spent designing a solution the less time that needs to be spent testing and debugging (removing errors).

Before the algorithm is coded into a programming language (e.g. PHP, JAVA) it should be tested and checked to see if it works. There are two common methods used to test an algorithm: -

- Desk-checking
- Structured walk-through

It is best to get someone else to check your algorithm, this is called **peer checking.** (If you wrote the algorithm you are less likely to spot the mistakes).

Desk checking

Desk Checking involves drawing up a table of the variables and outputs from the algorithm and then methodically working through the algorithm, with sets of test data, recording the changing values of the variables. The final results can be then be compared to the expected results to see if the algorithm works as it should.

Desk Checking Example
The following algorithm is meant to calculate the average of 5 numbers. So for the test data, of 5, 4, 6, 2, 8 the Expect Output would be the average of 5 (25/5 = **5)**

```
BEGIN
        Count = 0
        Total = 0
        Average = 0
        REPEAT
                count = count + 1
                Get number
                Total = Total + number
        UNTIL count = 5
        Average = Total/count
        Display average
END
```

Count	Total	Number	Average	Output
0	0		0	
1	5	5		
2	9	4		
3	15	6		
4	17	2		
5	25	8	5	5

Activity 1

The following algorithm should work out the total weekly pay for an employee, given the payrate and number of hours they have worked each day.

```
BEGIN
    count = 0
    Totalhours = 0
    Enter Payrate
    WHILE count < 7
            count = count + 1
            Enter hours worked
            Totalhours = Totalhours + hours
    END WHILE
    Pay = payrate * Totalhours
    Display Pay, Totalhours
END
```

1. Complete a desk check, using the following test data:-
   Payrate = $9.30, hours = 4, 2, 0, 8, 0, 0, 4
   Expected output is Total hours = 18 Pay = $167.40

Variables					Outputs
count	Hours	Totalhours	Payrate	Pay	

2. Outline whether the algorithm produced the required output.

Structured walk through and Peer Checking

A walk-through is used to quickly test the logic of an algorithm to find errors or to find out what an algorithm does. It is usually done by a peer (**peer checking**) who works through the algorithm or code to check for errors. This is an efficient way to improve the quality of program code and documents that describe the program requirements.

Activity 2

The following algorithm should print out the numbers 1 to 10

```
BEGIN
    count = 0
    WHILE count > 10
            count = count + 1
            PRINT count
    END WHILE
END
```

1. Use a structured walk through to identify the error in the algorithm

2. Outline examples of errors you might expect to find using a structured walk-through.

3. Why do you think peer checking is effective in finding errors?

Test data

The design of the test data is very important and should be produced from the program specification during the defining and understanding stage before you start developing the algorithm or code. Test data should check that the algorithm and code performs what it was intended to do and should be thorough enough to check all aspects of the problem.

For small to medium sized problems, the test data created or selected should include:-

- Data where the expected answers is known
- Data that tests **all logical pathways** in the algorithm, i.e. every option in a casewhere selection is tested
- Data that tests the conditions in loops and selection control structures where decisions are made, including having expected value
  - At **Boundary values**, which is a value on a limit where a condition is different either side of the value. E.g. for the condition age > 18 the boundary value is 18 as the value below, 17 is false and the value above, 19 is true
  - Middle values, which are values between limits
    - **Above the boundary** value
    - **Below the boundary** value
- test data that tests **outside the expected values** including
  - Unexpected values e.g. incorrect data types, very large values
  - Non-valid values e.g. 0, negative values

Activity 3

1. For the following section of algorithm

```
Enter Age
CASEWHERE Age
        < 15, too young to work
        <= 65, you can work
        Otherwise, too old
END CASE
```

   a) List the boundary values

   b) How many pathways are there through the algorithm?

   c) What are 2 non-valid age value ranges?

d) Create a set of test data that would fully test this algorithm.

Test Data	Expected Output	What it tests
15	You can work	Boundary value

2. For each of the following problems
   a. work out the boundary value(s)
   b. create a set of test data that would fully test the problem.

Problem	Boundary value(s)	Sample set of test data
Check the Voting Age		
Check the number of passengers in a normal Taxi		
Check the drink driving alcohol levels		

3. Explain what a boundary value is, using an example to illustrate the answer.

4. Outline where boundary values are normally found in control structures

5. Explain what a logical pathway is, use an example to illustrate the answer

6. Outline some of the reasons why it is not always possible to test all logical pathways in an algorithm.

Types of Errors

There are 3 main types of errors in code, Syntax, Logic and Run-Time.

Syntax Errors

A syntax error occurs when the code does not match the syntax of the language. Syntax errors will be detected when you try to compile the program, the program will not compile if you have syntax errors. Examples of syntax errors, including missing brackets, miss spelt or missing keywords.

Activity 4

For the following sections of PHP code find and fix the Syntax errors:

CODE	Error	Fixed code
ehco "Hello there";	Incorrect spelling of function	echo "Hello there";

if ($age < 13){
echo "Child"
}
	1 error	

ifs (age < 13){
echo "Child";
}
	2 errors	

$count = 0; while count < 10 $count = $count + 1; echo "count "; }	2 errors	
if ($age == 13){ echo Just a teenager;	2 errors	

Logic Errors

A logic error is where the algorithm or code does not produce the required action. These errors are normally detected during the test by using test data to check that the program works as it should, checking the actual output matches the expected output. Code with a logic error will run, but will not produce the required output.

Activity 5

1. For the following lines of php code **circle and explain the logic error(s)**, rewrite the code once without the logical error(s):-
   The following algorithms should print out the numbers 1 to 5, e.g. 1, 2, 3, 4, 5

Code	Output from code	Explain and fix the error

$count = 0;
while ($count < 3) {
$count = $count + 1;
Echo " $count";
}		

$index = 0;
while ($index < 5) {
$count = $count + 1;
echo " $count";
}		

$count = 0;
while ($count < 5) {
echo " $count";
$count = $count + 1;
}		

$count = 1;
while ($count < 5) {
$count = $count + 1;
echo " $count";
}		

2. The following code prints out if someone is able to legally vote

Test Data

Age	**Expected Output**
14	Not legal
25	Legal
18	Legal

Code	Output from code	Explain and fix the error

if ($age > 18) {
Echo "Legal";
}else{
Echo "Not legal";
}		

3. The following code calculates the average of 2 numbers
Test data, 6, 4 Expected output is 5

Code	Output from code	Explain and fix the error
$ave = $num1 + $num2 / 2; echo "The average is"; $ave";		

Run-Time Errors

A run-time error occurs when the program is running. It causes the program to terminate or stop. It may be the result of a logic error. Examples include,

- division by zero error (this error does not occur in php),
- an overflow error which is were data to be stored in a variable is larger than the declared variable can hold
- an infinite loop.

Activity 6:

1. The following algorithm should output the numbers 1, 2, 3, 4, 5

Code	Output from code	Explain and fix the error

$count = 0;
while ($count < 5) {
echo " $count";
}		

2. Outline why a division by zero error would cause a problem.

3. Outline how an overflow error causes a problem. Give an example of data and a variable that would cause an overflow error.

Error Detection and Correction Techniques

There are number of tools and techniques that can be used to help test and debug code to find logic and runtime errors, including stubs, flags and debugging output statements.

Stubs

Stubs are used to check

- The flow or connection between the modules (sub programs) are working
- If a subroutine is causing an error by replacing it with a sub (to check if the error is disappears, if it does then the error would be in the subroutine)
- A section of code, before all sub programs required by that code are finished.

Example
In the first example below the sub program has not yet been created, so some (arbitrary) values are return from the sub program, so that the rest of the program can be tested (e.g. the surface area formula for a cylinder)

```
REM the following program works out the surface area of a cylinder
DECLARE SUB circ (radius)
DIM SHARED area AS SINGLE
DIM SHARED perimeter AS SINGLE
DIM radius AS SINGLE
DO
        INPUT "Please enter the radius of the cylinder"; radius
        INPUT "Please enter the length of the cylinder"; length
        CALL circ(radius)
        surfarea = 2 * area + perimeter * length
        PRINT "The surface area is "; surfarea
        INPUT "Do you want to do again? (Y/N)"; again$
LOOP UNTIL UCASE$(again$) = "N"
END

SUB circ (radius)
        area = 30
        perimeter = 10
END SUB
```

Activity 7

Outline some reasons for using a stub.

Flags

Flags are used to check that a section of code has been processed or executed. They can be used as part of a solution or as an error detection process. They are normally Boolean variables that change to true (1) if the line of code is executed, e.g. $found = true;

Example
In the example below the flag is part of the solution. The flag finish is set to 1 to end the loop.

```
DIM Team(1 TO 9) AS STRING
Finish = 0
Index = 0
DO
index = index + 1
PRINT "Enter name of player (XXX to finish)"
INPUT names$
IF names$ = "XXX" THEN
        finish = 1
END IF
LOOP UNTIL finish = 1 OR index >= 9
```

Activity 8

1. Outline what the program above does.

2. Outline the purpose of the flag in this program

Debugging Output Statements

Debugging Output Statements are additional print/output statements in the code that help in telling what part of the code has executed or for working out what is happening to the program, especially what values the variables are holding. They can also be used to check if a section of code has been executed or to interrogate the values variables held at a particular point in the program i.e. print the value of a variable echo "$count";

Example
The cylinder program from before can have some output statements put in the program to let the user know what the values of area and perimeter are, before surfarea is calculated and to show if the subprogram was executed.

```
REM the following program works out the surface area of a cylinder
DECLARE SUB circ (radius)
DIM SHARED area AS SINGLE
DIM SHARED perimeter AS SINGLE
DIM radius AS SINGLE

DO
        INPUT "Please enter the radius of the cylinder"; radius
        INPUT "Please enter the length of the cylinder"; length
        CALL circ(radius)
        surfarea = 2 * area + perimeter * length
        PRINT "The area is "; area
        PRINT "The perimeter is "; perimeter
        PRINT "The surface area is "; surfarea
        INPUT "Do you want to do again? (Y/N)"; again$
LOOP UNTIL UCASE$(again$) = "N"
END

SUB circ (radius)
        area = 3.14 * radius * radius
        perimeter = 2 * radius * 3.14
        PRINT "subprogram executed"
END SUB
```

Activity 9

1. Outline some reasons you would want to know the value of a variable.

2. Outline how you use an output statement to check which IF-THEN-ELSE path is chosen.