

Producing robust programs

Input sanitisation and validation

Assume the user will input data incorrectly

PLANNING FOR CONTINGENCIES

What happens if it goes wrong? What error message do you want to appear?

ANTICIPATING MISUSE

Put yourself into the mind of your users and make the system as simple to use as possible. Use menus or short codes rather than entire words to cut down on potential spelling errors. If possible, make sure it is not case sensitive.

AUTHENTICATION

Ensuring only users who have the right to use that program can have access to it. Use email checks, passwords etc.



VALIDATION CHECKS

Validation Type	Description
Range check	the input must fall within a specified range. This is usually applied to numbers and dates but can apply to characters. For example, when making a payment to someone the amount to be entered might be set to be greater than zero and not greater than the funds available.
Length check	the input must not be too long or too short. For example, a surname will require at least one letter, but is unlikely to require more than 40.
Presence check	a data value must be entered. For example, entering a quantity when placing an order.
Format check	the data must be in the correct format, such as entering a date in the format DD/MM/YYYY.
Type check	the data must be of a specified data type, such as an integer when specifying a quantity.

Validation does not ensure that the data entered is correct, just that it is possible and sensible

TESTING

A programmer must thoroughly test their program to spot syntax and logic errors and correct them. This is usually done in two stages:

Iterative testing

Carried out while a program is being developed. The programmer writes a section of code then tests it.

Final/terminal testing

Carried out when all sections of the program is complete. The program is tested by running through it several times to test out different scenarios.

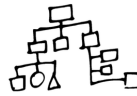
Code within sub programs or iterations should be **indented**. Some programs will not run unless correct indentation is used.



Comments provide information about what the different parts of the program do

Maintainability

Ensuring other programmers can make sense of the program you have written



You should select **variable names** that reflect the purpose or contents rather than short difficult to understand abbreviations

Syntax errors

A **syntax error** occurs when code does not follow the rules of the programming language, this includes:

- misspelling a word eg writing "iptu" instead of "input"
- using a variable in a statement before it has been created or declared
- missing brackets, speech marks to another grammatical error that code will be expecting etc.

Selecting and using suitable test data

Test data should cover a range of possible and impossible inputs

- **Valid data** - sensible, possible data that the program should accept and be able to process
- **Extreme data** and **range check** - valid data that falls at the boundary of any possible ranges
- **Invalid (erroneous) data** - data that the program cannot process and should not accept

Logic errors

A **logic error** is a problem with the way a program works and the route it will follow through a program. Logic errors can have many causes, such as:

- Incorrectly using logical operators, eg expecting a program to stop when the value of a variable reaches 5, but using <5 instead of <=5
- Unintentionally creating a situation where a loop will keep repeating and it cannot break out of it; this is known as an infinite loop
- Incorrectly using brackets in calculations
- Unintentionally using the same variable name at different points in the program for different purposes

Nº	Data to test	How it is tested	Expected outcome	Actual outcome	Pass/Fail
1	num	Valid data – (15)	Counts from 1 to 15	Counts from 1 to 15	Pass
2	num	Presence check – leave num blank	"Invalid entry"	"Invalid entry"	Pass
3	num	Invalid data – enter a string ("Hello")	"Invalid entry"	"Invalid entry"	Pass
4	num	Range check – enter a number outside range (2)	"Invalid entry"	"Invalid entry"	Pass
5	num	Extreme data – data on the edge of the range (10)	Counts from 1 to 15	"Invalid entry"	Fail

A test table is used to record the results of the tests that are performed

