



Diploma in

# Computer Science



Error Handling





Identify the types of errors in programming <

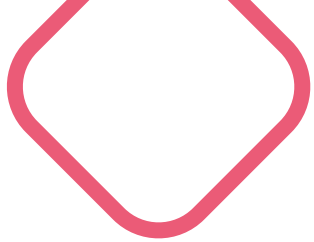
Explore error handling techniques <

Describe built-in ways of dealing with errors <

Design a new solution <



# Objectives



# Programming errors





# Errors, bugs and exceptions

- Not all errors are bugs, not all bugs show up as errors
- Referred to in other programming languages as exceptions
- Exception: a unique event that prevents the code from working as intended





# Errors, bugs and exceptions

**Bug:** a result of bad programming – program is the problem

**Error:** a result of external factors





# Errors, bugs and exceptions

- Program should be able to handle foreseeable errors
- Computers shouldn't present users with cryptic error messages





# Error handling





# Did you know?

Error handling is separated into two distinct groups: error detection and recovery.







# Error handling methods

- Prevention
- Termination
- Global indication error
- Local error indicator
- Return values
- Assertions
- Signals
- Goto chains
- Non-local jumps
- Error call-backs



# Error handling methods: Prevention

- Attempt to write the code so that errors don't happen

**Example:** program asking user for a pin

- Other instances that make prevention impractical

**Example:** the  $\tan(x)$  function can overflow for values of  $x$  near  $n \cdot \pi\pi/2$ , and  $n$  here can be any odd integer



# Error handling methods: Termination

Method of ascertaining the reason why a program exited.



Function	Closes open streams	Flushes stream buffers	Removes temporary files	Calls exit handlers	Terminates program
Abort()	Implementation defined	Implementation defined	Implementation defined	No	Abnormal
_exit()	Implementation defined	Implementation defined except in POSIX	Implementation defined	No	Normal
Quick_exit()	Implementation defined	Implementation defined	Implementation defined	At quick_exit()	Normal
Exit()	yes	yes	yes	Atexit()	Normal
Return from main()	yes	yes	yes	Atexit()	Normal





# Error handling methods: Global error indicator

- Static variable or data structure that indicates the error status of a function call
- Three types:
  - `errno`
  - `feraiseexcept()`
  - `getlasterror()`
- All fail-soft



# Error handling methods: Global error indicator

- Both a good and bad feature

**Example:** in files, if the program kept going, writing junk to your files with no way of recovering the data



# Rules for using errno

- Set to zero before calling a function, and use it only after the function returns a value indicating failure.
- Value is zero at program start up but must never be set to zero by any library function.
- The value may be set to non-zero by a library function call even if not part of the C standard.
- Only meaningful after library function has returned error code.





# Errno error codes

Errno value	Error
1	Operation not permitted
2	No such file or directory
3	No such process
4	Interrupted system call
5	I/O error
6	No such device or address
7	Argument list too long
8	Exec format error
9	Bad file number
10	No child processes
11	Try again
12	Out of memory
13	Permission denied





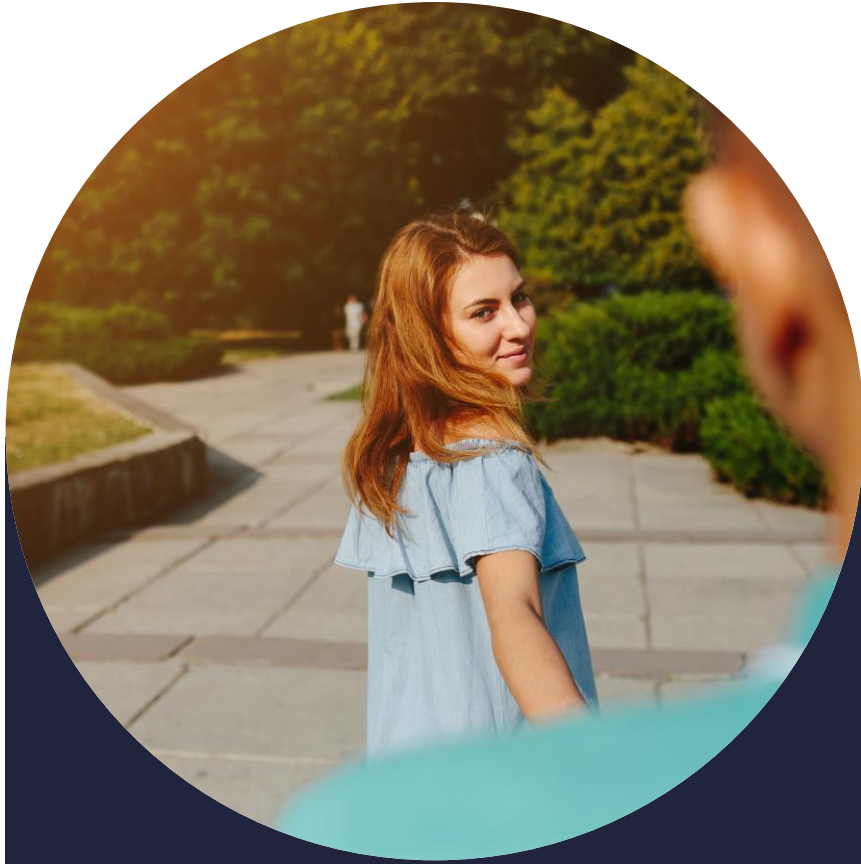
# Error handling methods: Non-global error indicators

- Variables that indicate an error state
- Typically used with files
- Three notable functions for this purpose:
  - `feof()`
  - `ferror()`
  - `clearerr()`
- Work the same way as `errno`



## Error handling methods: Jumps

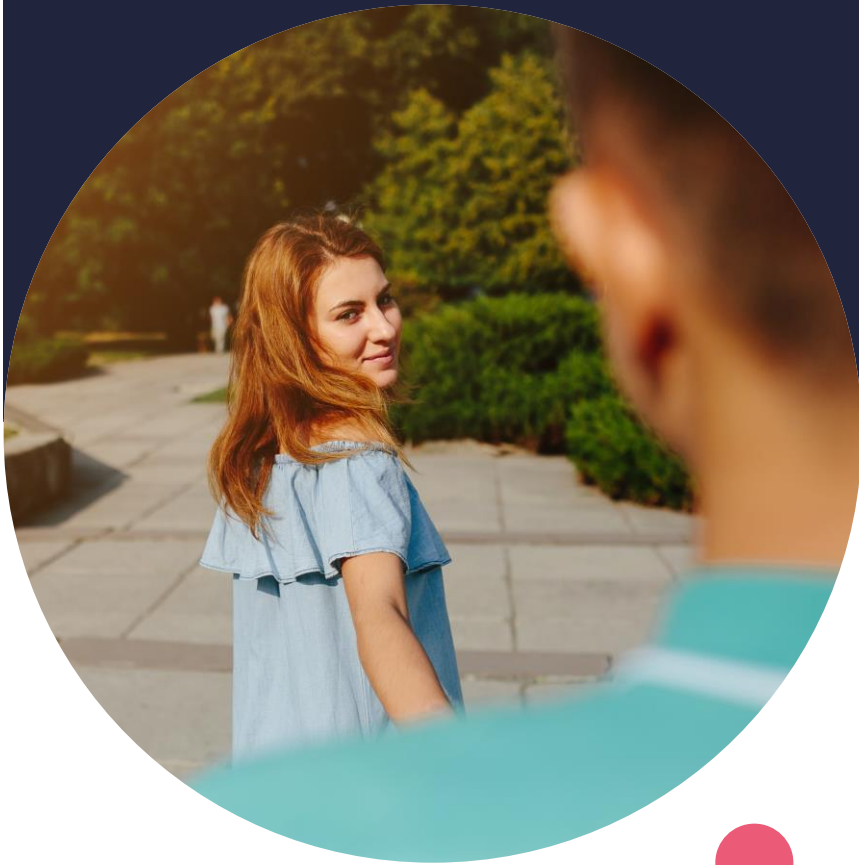
- If condition that causes an error is encountered, code jumps to a predetermined line and resumes execution
- Not a good way of writing code
- Notoriously difficult to debug
- Better to use structured loops



# Error handling methods: Function return values

- Values that are used by functions to indicate errors by returning a value

**Example:** malloc function returns a null pointer if there's an error in the allocation process

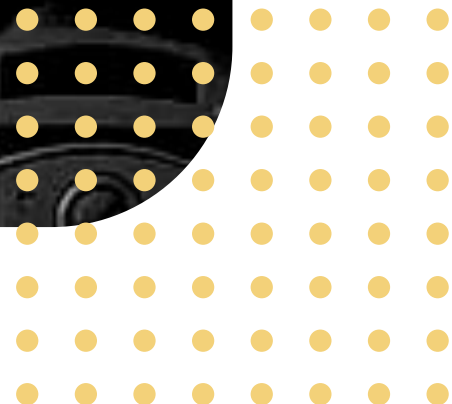
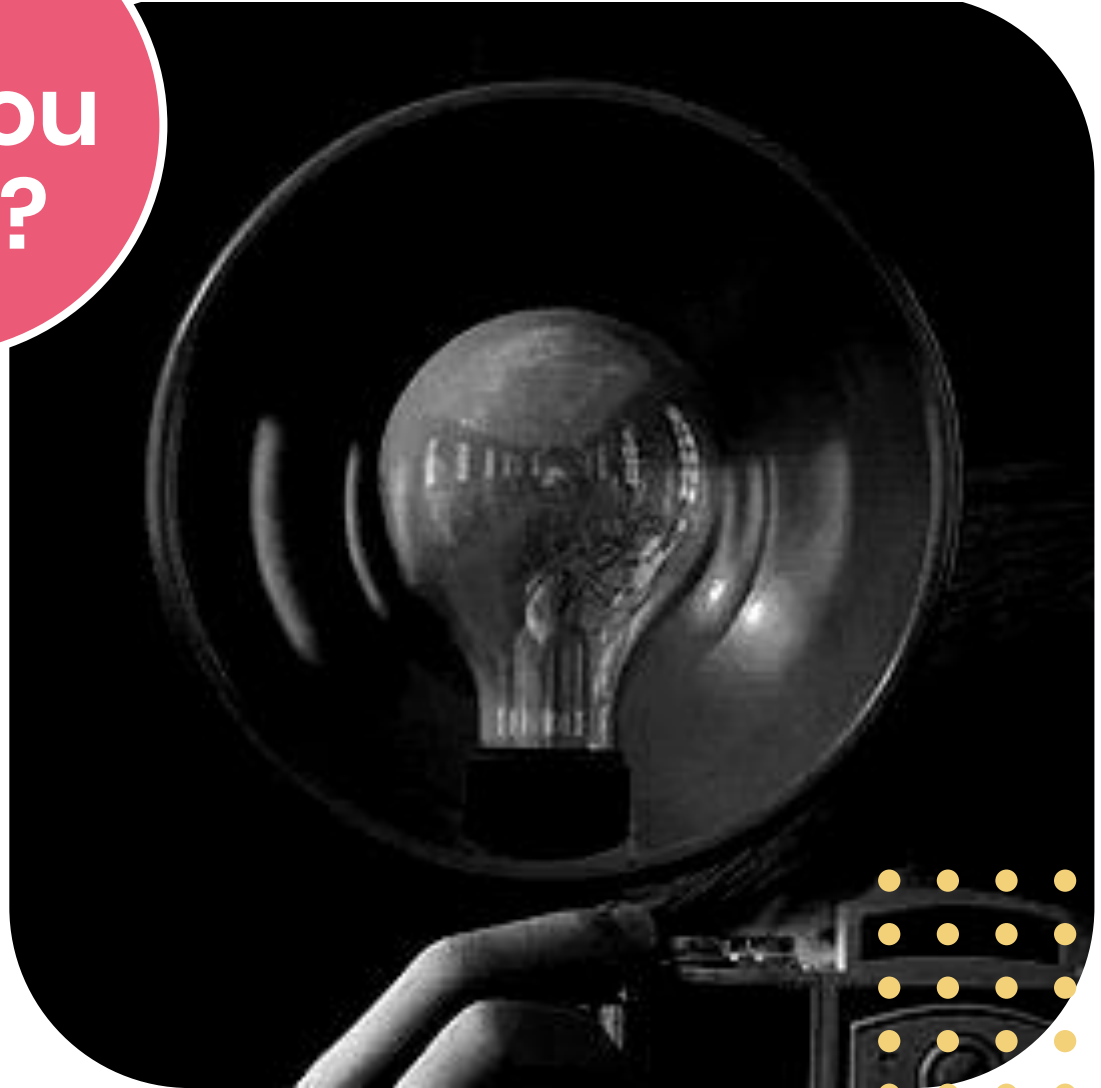


# Error handling methods: Function return values

- The caller function is responsible for checking for error conditions
- Return value cannot be used for other purposes
- Can make code significantly longer
- Easy to ignore return value so can create code security concerns

**Did you  
know?**

**A vulnerability arose in  
Adobe Flash Player  
partially because it  
ignored the return  
value of calloc().**



# How to ascertain which error handling method to use...

