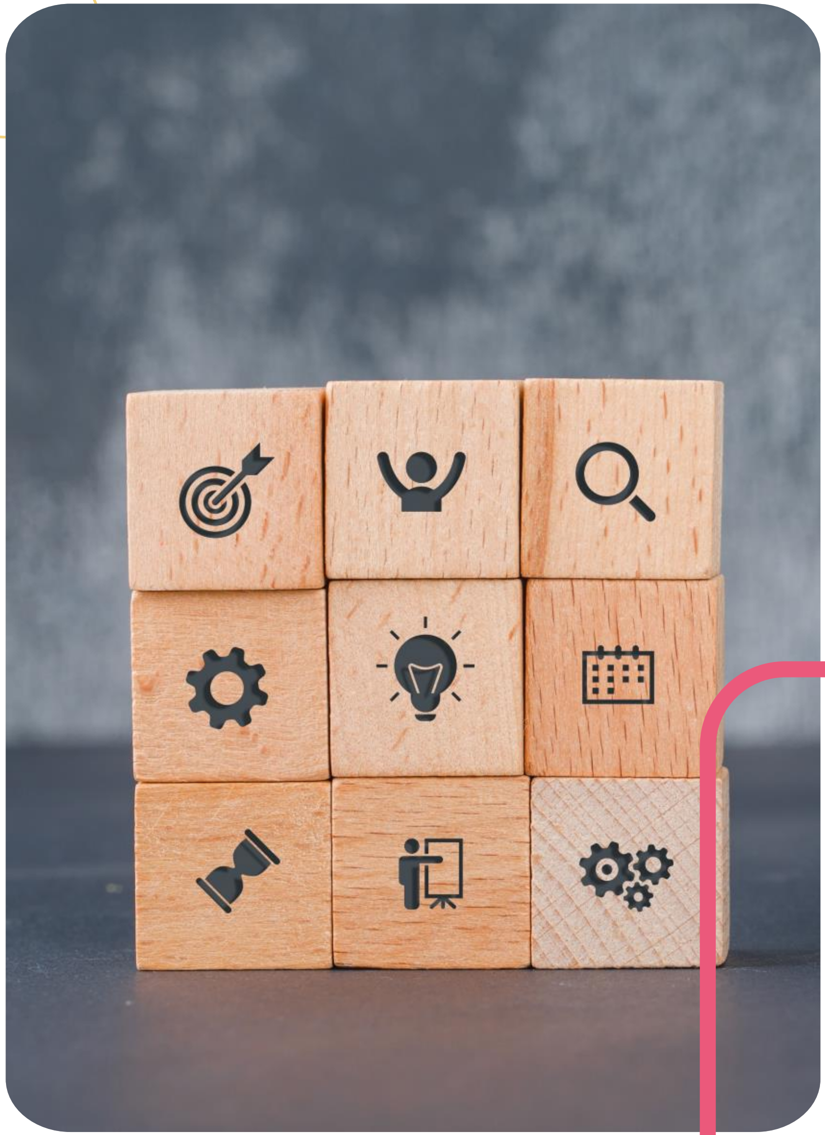
Diploma in

Computer Science

Storage Classes



Objectives

- Appreciate the importance of scope in software integrity
- Recall the importance of variables as a tool for data manipulation
- Recognise how close C is to hardware

Challenge

Create a program that demonstrates scope using the mayor and prime minister example used in this lesson.

Scope and software integrity

Variable scope

- Visibility of variables to other parts of the program.
- Global variable: visible anywhere in the program.
- Local variable: only visible in area where it is declared.

Variable scope

- Determined by the space in which the variable is contained.
- Demarcated by relevant tokens.
- Space between the open curly bracket {and the closing curly bracket} is a scope.

Variable scope example

The mayor (local variable) can make changes and name roads, build new structures in a town.

The prime minister (global variable) can do the same in any town and does not need to work with the mayors to make things happen.

Global variables carry their visibility among different source files

Function scope

Functions also have scope

Functions are above the main function - global scope

Variables declared within it will be active from the beginning to the end of the function

Did you know?

In C programming language, only the goto label has a function scope.

Formal parameters

Also a scope as they are only visible to the function

A bit more about scope...

- Variables should never encroach into each other's space.
- Parametric values are passed from one function to another as input:

If passed by reference: generally global and accessible outside the scope of the function

If passed by value: two copies exist

- Scope allows us to use memory as efficiently as possible.
- A local variable can be destroyed when we no longer need it.
- This frees up memory for other things.

Storage classes

Four types of storage classes

- Auto
- Register
- Static
- Extern

Linkage

The ability of an identifier to be used in other scopes.

- No linkage
- Internal linkage
- External linkage

Linkage

No linkage

Variable can only be used in the scope it is in.

Linkage

Internal linkage

Variable can be used in any other translation units in the entire program.

Linkage

External linkage

- Variable can only be used in the scope it is in.

Four types of storage classes

Auto specifier

Auto storage class is default for all variables.

Variables take on whatever is in the memory location.

Storage allocated when block is entered and de-allocated when block is exited.

Only exception is variable length arrays.

Register specifier

Specifier is borrowed from computer hardware.

Storing variables in registers may help speed up your program.

Ideal for when your variable is heavily used.

Your variable will be assigned to a register subject to availability.

Restrictions with register specifier

Can’t use pointers.

Can’t use when declaring objects in global scope.

Can’t apply address operator.

Has no linkage.

Static specifier

Major mechanism for enforcing data hiding in C.

Can be used with functions at file scope and with variables at both file and block scope, but not in function parameter lists.

Objects have static storage duration.

Variable can have static duration but local scope.

Can be used on data objects and anonymous unions but not with type declarations and function parameters.

Can be used in the declaration of an array parameter to a function.

Static specifier

Object takes on internal linkage.

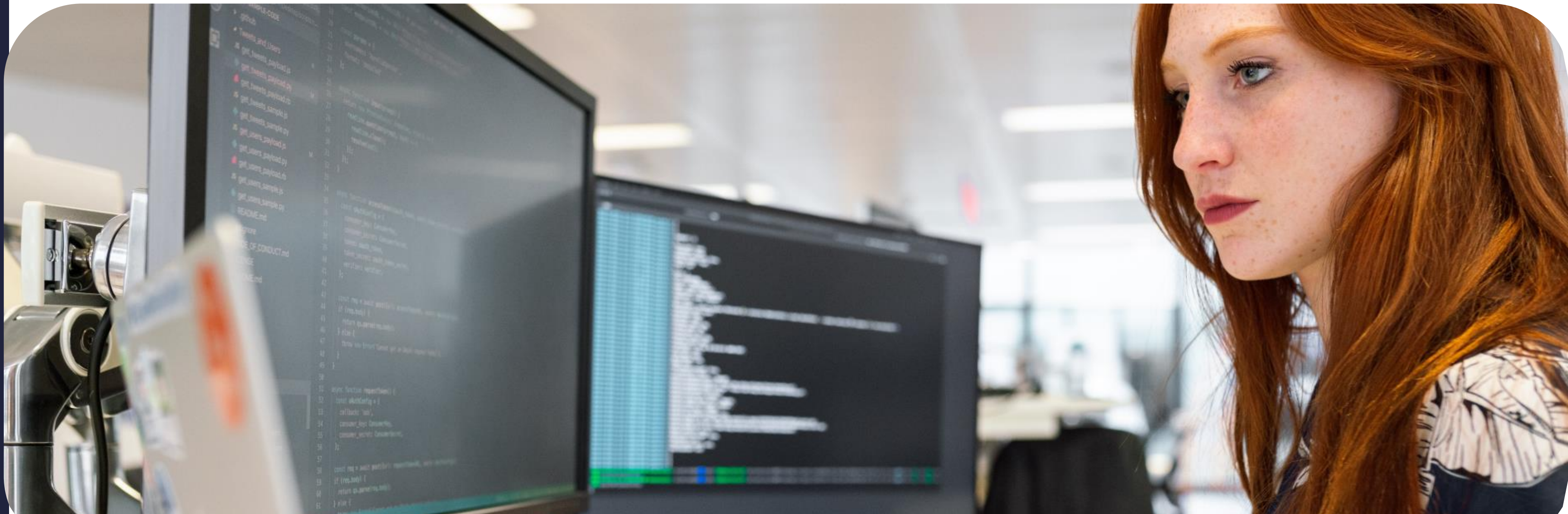
Each time the program returns to the scope of the object, the last value carried by the object is still there.

Extern specifier

Declares objects that will be used in several source files

Can be used with function declarations, object declarations. and block scope but not for function parameter lists

Assumes a static storage duration unless combined with thread local



Extern specificer

Scope depends on the location of the declaration in the program text

Can appear outside a function OR at beginning of a block

Linkage depends on location

Extern specificer

Storage Class	Declaration	Storage	Default Initial Value	Scope	Lifetime
auto	Inside a function/block	Memory	Unpredictable	Within the function/block in which they are defined	Within the function/block in which they are defined
register	Inside a function/block	CPU Registers	Garbage	Within the function/block in which they are defined	Within the function/block in which they are defined
extern	Outside all functions	Memory	Zero	Entire file and other files where the variable is declared as extern. not bound by any function, available everywhere	Program runtime
Static (local)	Inside a function/block	Memory	Zero	Within the function/block	Program runtime
Static (global)	Outside all functions	Memory	Zero	Global	Program runtime

Guidelines for when to use what...

- **Static** – for variables that need to be the same every time they are called
- **Register** – for variables that are used frequently
- **External** – for variables used by all functions in the program
- **Auto** – when variables don’t need special treatment

Code security

Why all this?

There's a feature in the way the brain interprets spatial information to help you refresh your working memory, just like degaussing old CRT monitors.

Why all this?

Environment dictates where your application will reside and how it behaves.

When using scope and storage access specifiers, ask yourself:

- Which operating system are you targeting?
- Are you targeting any specific hardware?
- Where will the application be run from?

Why all this?

Designing for a specific environment allows you to use the full potential of that environment and write extremely secure code.

Making code portable exposes it to different environments, libraries, error handling schemes.

Optimising for environment may end up being a hindrance in a different OS.

Scope interactions to find your attack surface

Before you start writing code, look at your pseudocode and flowchart.

The scope is the first port of call for determining the application's attack surface.

Security starts with understanding.