Qualifiers and storage classes

By: Yehia M. Abu Eita

Outlines

- Introduction
- Type qualifiers
- Size qualifiers
- Sign qualifiers
- Storage classes

Introduction

- **Qualifiers** are **keywords** that are **applied to a data type** resulting in a qualified **type, size, or sign** data type.
- **Storage classes** are **keywords** that are **applied to a variable or a function** that defines their **scope (visibility) and life-time**.
- In C there are **four storage classes**, **auto**, **static**, **extern**, and **register**.

Type qualifiers

- Are **keywords** that are **applied to a type** resulting in a qualified **type.**
- The **const** qualifier:
  - When it is added to a variable it qualifies this variable to be **constant**, **can not be modified during execution.**
  - `const int x  = 10;`
- The **volatile** qualifier:
  - When it is added to a variable it qualifies this variable to become **volatile** and **can not be optimized** by the **compiler** because this variable **may** be **changed without** any **action taken by your code**.
  - `volatile int x = 15;`

Size qualifiers

- Are **keywords** that are **applied to a type** resulting in a qualified **size type.**
- The **short** qualifier:
  - It is added only to **integers**.
  - When it is added it qualifies **integer size** to become **2 bytes instead of 4 bytes**.
  - `short int x  = 10;`
- The **long** qualifier:
  - It is added only to **integers**.
  - When it is added it qualifies **integer size** to become **4 bytes instead of 2 bytes**.
  - `long int x  = 10;`

Sign qualifiers

- Are **keywords** that are **applied to a type** resulting in a qualified **sign type.**
- The **unsigned** qualifier:
  - It is added to **integers and characters** only.
  - When it is added it qualifies **integers or characters** to become **unsigned.**
  - `unsigned int x  = 10;`
- The **signed** qualifier:
  - It is added to **integers and characters** only.
  - When it is added it qualifies **integers or characters** to become **signed.**
  - `signed int x  = -10;`

Storage classes

- The **auto** storage class:
  - It is the **default** storage class for all **local variables**.
  - It makes the variables stored in the **stack memory** section.
- The **extern** storage class:
  - It can be added to **global variables** and **functions**.
  - It makes the global variables and functions **shared** between **all project files** and **stores** the **global variables** into **data segment**.
  - `extern int x = 50;`

Storage classes

- The **static** storage class:
  - It can be added to **global**, **local variables**, and **functions**.
  - When it is added to a **global variable** or a **function**, it **hides** this variable from being seen in other files, **private**.
  - When it is added to a **local variable**, it **prevents re-initialization** of this variable and **retain the last value** stored in this variable.
- The **register** storage class:
  - The register storage class is used to define **local variables** that **should** be **stored** in a **register** instead of RAM, **this decision is taken by the compiler**.
  - The register variable will **have a maximum size equal** to the **register size** and **can't get its address**.
  - `register int x = 50;`

Summary

- Now you are familiar with qualifiers and storage classes in C.
- Remember, qualifiers control type, size, and sign of a variable.
- Storage classes control the visibility and lifetime of the variables or functions.