CSD101: Introduction to computing and programming (ICP)

Values, names and types

- All programs manipulate values to achieve their purpose.
- Within programs these values are accessed via names so all programming languages have some way to bind names to values. Many languages call this binding operation **assignment**.
- To ensure that values are manipulated correctly most programming languages require values to have **types**. A type (some times called data-type) denotes a set or class of values along with a set of operations on those values.
- Many languages (**C** among them) require that before using a name a type be declared for the value that can be bound to the name. This is called the **declaration before use** principle. This allows compilers to check whether programs are binding the right kinds of values to names and whether they are being manipulated correctly.
- Values can be **primitive** (or equivalently **atomic**) or they can be **composite**. Composite values have structure (e.g. a sequence is a composite value - called a 1-Dimensional array in **C**).

Rule for legal names in **C**

- A legal name in **C** must start with an alphabetic character (‘A’ to ‘Z’ or 'a' to 'z') or underscore (‘_’) and can contain alphabetic characters, underscores or digits after the first character.
- While no length limit is defined most compilers distinguish the first 31 characters. So, if two names agree on the first 31 characters but differ after that the compiler will treat it as the same name.

Statements

Definition 2

A statement is the basic consituent unit of a program. A program consists of a sequence of statements.
Statements can be simple or compound.
Statements can also be declarative or commands/action statements.

- In C a simple statement is always terminated by a semi-colon - `';'`.
- A compound statement has one or more embedded simple/compound statements. A typical compound statement is a sequence of simple (or compound statements) demarcated by curly brackets - `{s1; s2; ... sn}`, where `s1` to `sn` are themselves statements. A compound statement is often called a **block**.
- The execution sequence of a C program follows the execution sequence of statments in the program starting with `main`. The normal sequence can be interupted/changed by control statements and function calls.

Kinds of statements in **C**

- Declaration. This describes the types of variables and is not an action or command but is still considered a statement.
- Expression (simple).
- Compound or block.
- Labelled (simple/compound).
- Control. There are three kinds of control statements.
  - Conditional (compound).
  - Unconditional: `break`, `continue`, `goto` (all simple).
  - Iteration or looping (compound).
- Function calls (simple). Input/Output is done using function calls.

Two important control statements

- Conditional execution and looping are the two most important constructs that are present in all languages.
- **C** has the following conditional constructs:

  ```
  if <cond> <statement>;
  if <cond> <statement_1>; else <statement_2>;
  switch (discussed later)
  ```

- The looping constructs are:

  ```
  for (initializer; terminator; step) <statement>
  while <cond> do <statement>
  do <statement> while <cond>
  ```