Programming In C

Part II

Constants, Variables, and Data types

The C Character Set

C uses:

- Uppercase letters (A→Z)
- Lowercase letters (a→z)
- Digits (0→9)
- Some special symbols (+, *, %, &, #, {, }, …etc).

Variables and Constants

- Variables and constants are named memory locations in which data of a certain type can be stored.
- Constants refer to data that remains unchanged throughout the program execution.
- Variables store data that may vary during program exe
- Variables and constants must be declared before they can be used in a program.

Identifiers and Keywords
1. Identifiers

- Identifiers are names given to various program elements such as variables, functions, and arrays.
- Rules for naming:
  - Only alphabets, digits, and underscores are allowed.
  - An identifier must start with a letter or an underscore, and may be followed by any combination of characters, underscores, or the digits 0-9.
  - The name cannot start with a digit.
  - Uppercase and lowercase letters are treated as different.
  - <u>Optional</u>: always choose a meaningful names for identifiers in order to make programs easier to read.

Identifiers and Keywords
1. Identifiers

- Examples of valid identifiers:

  x, y21, sum, sum_1, taxe_rate, _temperture.

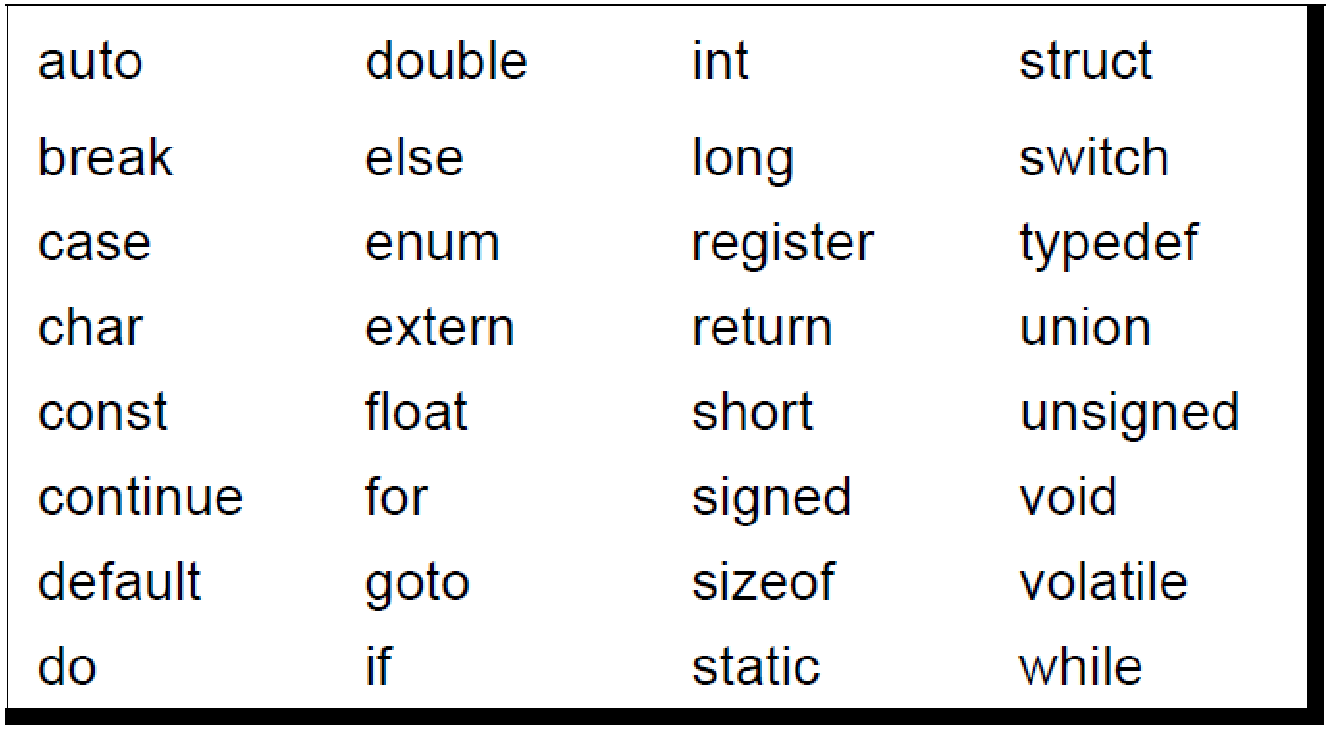
- Examples of non valid identifiers:

  4th, “x”, order-no, error flag

- C makes a difference between uppercase and lowercase characters. Example: “Age” and “age” are different identifiers.

2. Keywords

- Keywords are reserved words that have standard and predefined meaning in C.
- C has 32 keywords:



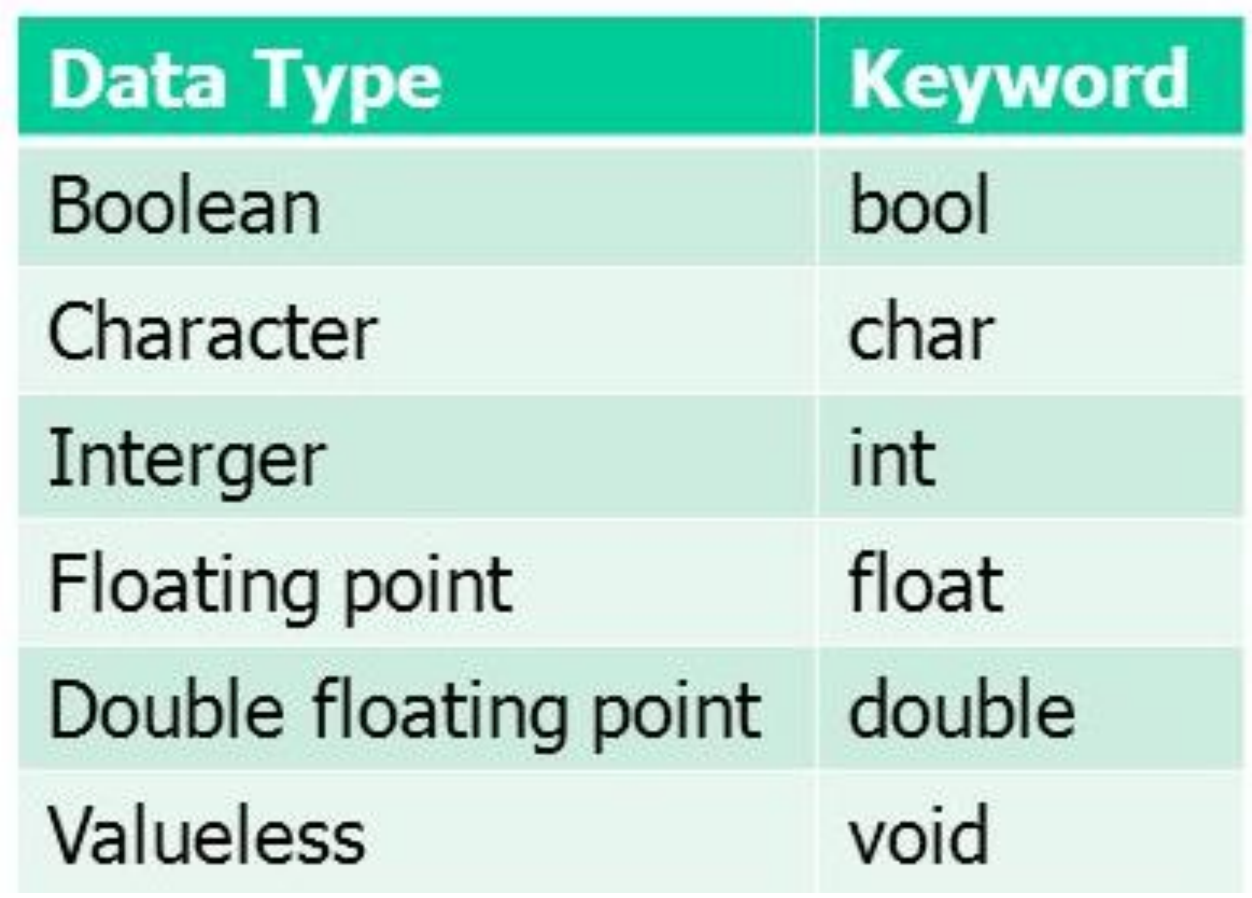
auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

- *Remark*: keywords cannot be used as user-defined identifiers.

Data types

- Each variable in C has a specific type, which determines the size and layout of the variable's memory.
- Data type specifies the range of values that can be stored within that memory; and the set of operations that can be applied to the variable.
- The C compiler must be informed about the type of data you intend to store in a variable.
- Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the reserved memory.

Fundamental Data Type



Data Type	Keyword	
Boolean	bool	1 byte (8 bits)
Character	char	1 byte (8 bits)
Interger	int	4 bytes (32 bits)
Floating point	float	4 bytes (32 bits)
Double floating point	double	8 bytes (64 bits)
Valueless	void	

Data type qualifiers

- The fundamental data types can be associated with some qualifiers for their **size** and **sign**.
- For size: short/long
- For sign: signed/unsigned
- Examples:
  - **short int** (2 bytes)
  - **unsigned int** (4 bytes positive numbers)
  - **unsigned short int** (2 bytes positive numbers)
  - **long double** (10 bytes)
- If we do not specify either signed or unsigned, the C compiler assume the type to be **signed**.
- If short, long, or unsigned are used without data type specifier, it is taken as an **int** by the C compiler.

Declaring variables in C

- In general variables are declared at the beginning of a block of a function. A block of a function is denoted by { }. Variables inside of a block are local to that block.

```
int main( ) {
int a,b;
float c;
char d;
}
```

- Variables may be initialized when declared with the = operator like:

```
int main( ) {
int a=1;
float b=1.2;
char c='a';
}
```

Declaring constants inside a function

- Variables can be declared using constant qualifiers to indicate that its value does not change during the program execution.

  Example:

```
int main( ) {
int a,b;
const float c=6.022
float d;
}
```