System SW

Lecture 1 – Basics of C programming language – Part 1

Jarno Tuominen / Reviewed by Sanna Määttä

Foreword on literature

- There are tons and tons of excellent study/training material available in the web
  - MOOCs
  - Online compilers
  - Free, online books
  - Tutorials
    - https://www.tutorialspoint.com/cprogramming/index.htm
    - https://www.geeksforgeeks.org/c-language-set-1-introduction/

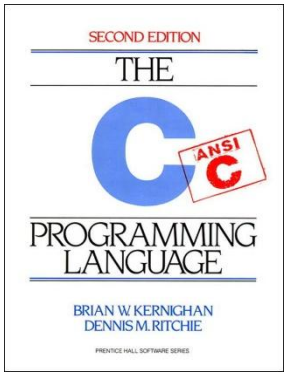
- Feel free to use any such resources in your studies
  - This means: No specific C-programming book in this course

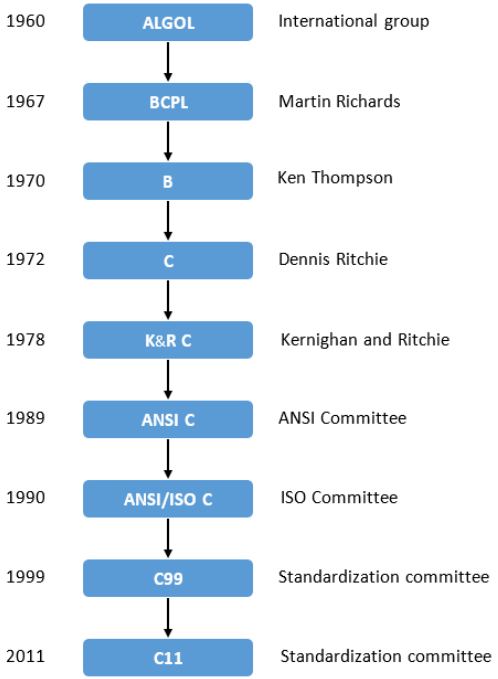
Lecture 1 – Basics of C programming language – Part 1

- History of C
- C basic syntax

The C programming language

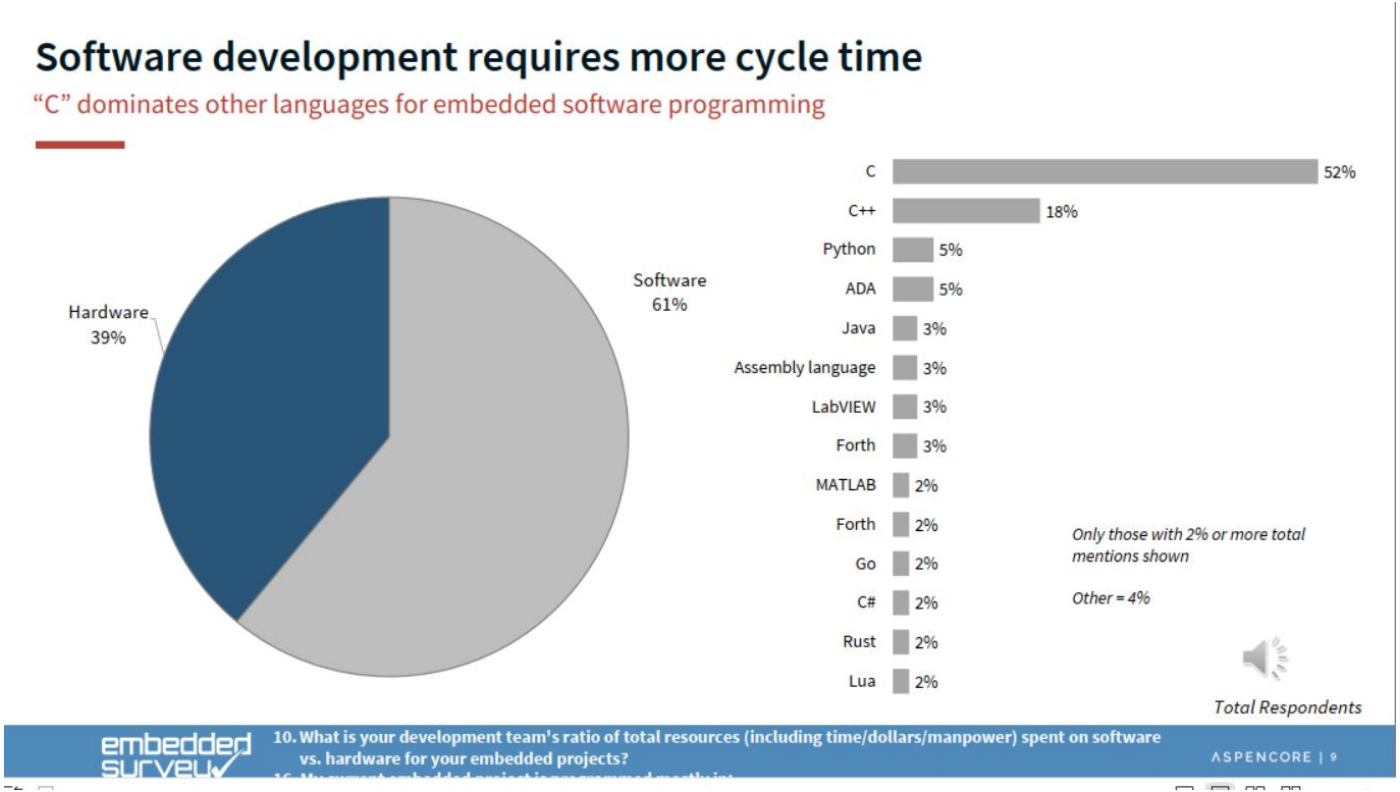
- C language was originally developed in 1972 by computer science scientist Dennis M. Ritchie
- C was invented to write an operating system called UNIX
- C is a general-purpose programming language
  - Not targeted to any specific purpose or computer architecture
- In this course: ANSI C (C89) unless otherwise mentioned





Also C17/C18 and C23 revisions exist.

Why C? Because it is the #1



Lecture 1 – Basics of C programming language – Part 1

- Review
- History of C
- C basic syntax

Just 32 keywords in C

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

- Keywords are those words whose meaning is already defined by **compiler**
- Cannot be used as **variable name**
- C Keywords are also called as **reserved words**

Hello class! – your first C-program

```
/*
* hello.c
* Created on: 27.8.2018
* Author: jmtuom
*/

#include <stdio.h>

int main(void) {
	printf("Hello DTEK2041 class!\n"); //This outputs formatted text
	return 0;
}
```

/* This is a block comment */

Preprocessor directives start with #

Blocks of code are enclosed in curly braces { }

// Starts a single line comment

Each statement must end with a semicolon ;

Hello class! – your first C-program

```
/*
* hello.c
* Created on: 27.8.2018
* Author: jmtuom
*/

#include <stdio.h>

int main(void) {
    printf("Hello DTEK2041 class!\n"); //This outputs formatted text
    return 0;
}
```

Basic I/O facilities like printf() is defined in stdio library, so we include it here

A C program must have one and only one main()-function, which is the entry point

Return integer, as ”promised” in function header

This function returns a value of type int, and it takes no arguments (void)

Keyword bingo

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

Variables

- Must be declared before use

```
int n; // int = integer data type
float pi; // float = floating point data type
```

- Variables are uninitialized – init happens via assignment operator ”=”

```
n = 5;
pi = 3.14159;
```

- Can also be initialized at declaration

```
int n = 5;
```

- Can declare/initialize multiple variables at once

```
int a,b,c = 0, d = 4;
```

Naming of identifiers

- Identifier = the name of a variable, constant or a function

- Identifier can contain alphanumberic (a-z,A-Z,0-9) and underscore (_) characters only
- Below are some valid identifier names.
  - number
  - _money
  - _student_
  - car1234
  - home321_
- It can't start with digit - i.e. '9number' is invalid identifier
- Identifiers are case sensitive.
  - i.e. 'number' and 'Number' are two different identifiers
- A good identifier name should be descriptive (e.g. int sum instead of int s)

Scope of variables

- Scope is a region in a programming language
  - Variable scope is the region in the program where you can define and use the variables and it can be used only in that scope
- There are 2 types of variables:
  - Local variables
  - Global variables
- **Local variables** – The scope of local variables is within the function only, the variables which are defined inside the function can't be accessed outside the function.
- **Global variables** – Scope of the global variables is throughout the program. These variables can be accessed anywhere in the program.
  - Mostly global variables are declared after the header file and before the main() function
  - In case of lot of global variables, they can be defined in a separate header file (like globals.h)

Scope of variables - example

```
#include <stdio.h>
int a=12,b=22;

void fun();

int main()
{
        printf("Global vars can be accessed anywhere in the program %d %d",a,b);
        fun();
}

void fun()
{
        int m=111,n=222;
        printf("Local vars can only be accessed inside this fun function %d %d",m,n);
}
```

Global variables

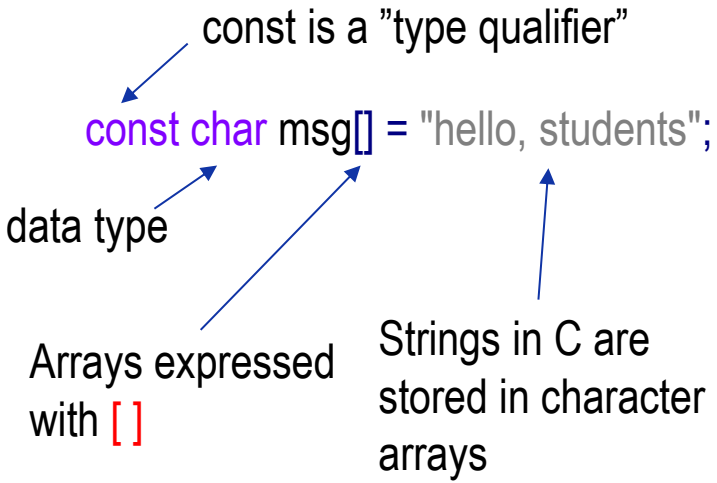
For compiler: Prototype/declaration of a function **fun**. If definition would be before main(), no declaration needed.

Local variables

Definition (body) of the function **fun**

Constants

- Like variables, but their value cannot be changed during code execution
- Also called **literals**
- Constants are stored in code memory space (ROM) instead of data memory space (RAM)
- **const** keyword: qualifies variable as constant
- **char**: a data type containing a single character (1 byte)
- Here we created a constant array of characters – a string constant



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

Questions?