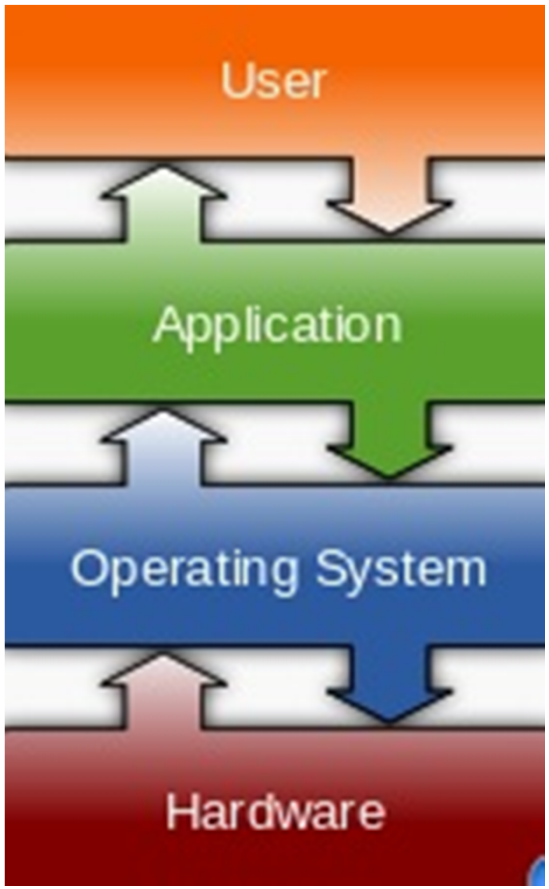
Computers Programming
Course 4

Iulian Năstac

Recap from previous course **Operating Systems**

- The operating system is an essential component in a computer.



Recap
OS Classification

- **batch processing**;
- **multiprogramming**;
- **time sharing**;
- **multiprocessing**.

Recap
Other classification considers the following operating systems:

- Real-time
- Multi-user
- Multi-tasking / single-tasking
- Distributed
- Embedded

Recap
Writing information on mass memory (HDD)

- A hard disk drive (HDD) can be divided into multiple logical storage units (partitions)
- A separate file systems can be used on each partition
- Most used file system architectures:
  - File Allocation Table (FAT)
  - High Performance File System (HPFS)
  - New Technology File System (NTFS)

Software's classification in a PC

- **Operating system(s)**

- **A various collection of electronic files and directories**, which includes: user files, programs, data, etc.

- **Programming languages**

Programming Languages

Definition:

A programming language is a formal language based on instructions, which is designed to implement a specified task.

Kinds of programming languages

- **Low level** (assembling languages)

- **High level**
  - **Based on interpreters** (BASIC, MATLAB, JAVA, etc).
  - **Based on compilers** (FORTRAN, PASCAL, ADA, C, etc.)

Interpreter

- An interpreter translates the source code into some efficient intermediate representation and immediately execute this.

Compiler

- A compiler transforms the source code written in a programming language (the source) into an object code or further in an executable program.

An extended classification of the programming languages

- **High level** (Ada, Pascal, Fortran, etc.)
  - programming languages with strong abstraction from the details of particular computer
- **Medium level** (C, C++, FORTH, etc.)
- **Low level** (assembly languages)
  - programming languages that provide little or no abstraction from a computer's instruction set architecture

C programming language

- 1966 Martin Richards (University of Cambridge) developed **BCPL** (Basic Combined Programming Language)

- 1969 **Ken Thomson** with contributions from Dennis Ritchie – **B programming language**

- 1969-1973 **Dennis Ritchie** – **C programming language**

Development of C programming language

- Beginning of ’70 – UNIX code was rewritten in C
  - Since then there is always a C compiler (**Unix's C shell**) embedded in every UNIX (even in some UNIX-like) operating system.

- 1978 **Dennis Ritchie** and **Brian Kernighan** had elaborated a famous book, "The C Programming Language".

Languages based on C

- C#, C++, Objective-C
- D
- Go
- Rust
- Java, JavaScript
- Limbo,
- LPC
- Perl
- PHP
- Python
- Verilog

A standard was needed ...

- Before the end of '80, many users relied on an informal specification contained in the book of Dennis Ritchie and Brian Kernighan (version is generally referred to as "K&R" C)
- 1989 the American National Standards Institute published a standard for C (generally called "**ANSI C**" or "**C89**")
- 1990 - ISO approved an international standard (called "C90").
- 1995 - ISO released an extension of C standard
- 1999 - a revised standard (known as "**C99**")
- December 2011– another revised version of the standard (**C11**)

- 2017-2018 - a new version of the standard ("**C18**")

C++ programming language standard

- 1998 C++ standard was ratified as ISO/IEC 14882:1998.
- 2003 - the standard was amended by the technical corrigendum, ISO/IEC 14882:2003.
- 2011 - extending C++ with new features was ratified as ISO/IEC 14882:2011 (informally known as C++11)
- 2014 - C++14 standard supersedes C++11 with new features and an enlarged standard library
- 2017 - The C++17 specification reached the Draft International Standard (DIS) stage in March 2017.
- C++20 standard is expected to be published before the end of 2020.

Portability

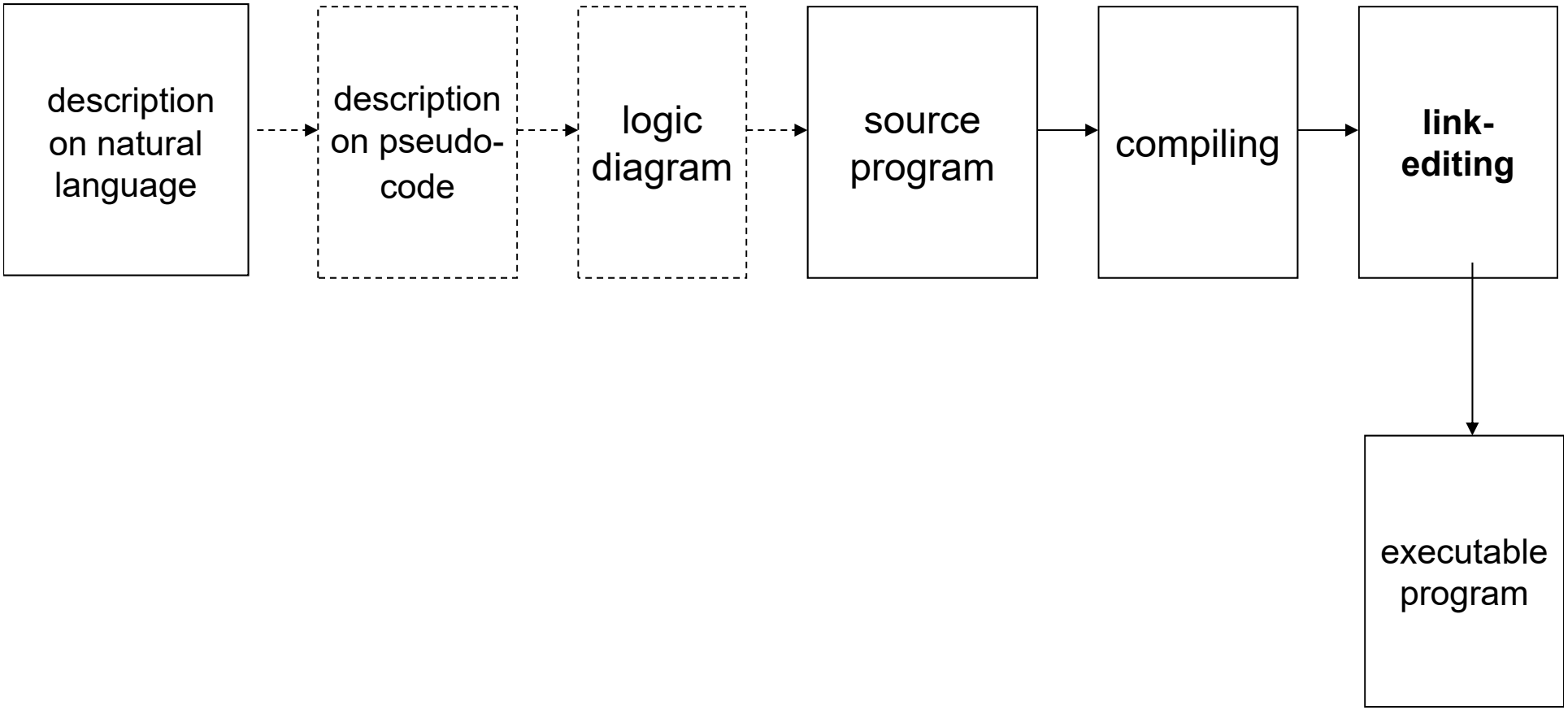
- **Portability is the property of a software to work properly on an changed environment.**

Porting

- **Porting** is the process of adapting software.
- This way an executable program can be used for a computing environment that is different from the one for which it was originally designed.
- The lower the cost of porting software, relative to its implementation cost, the more portable it is said to be.

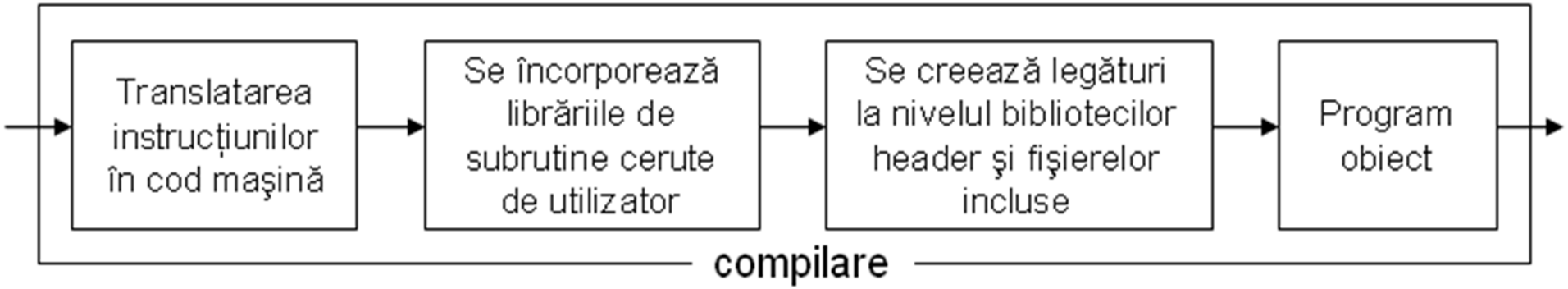
Obs.

- The portability concept can be established at different levels:
  - description on natural language
  - description on pseudocode
  - logic diagram
  - source program
  - compiling
  - link-editing
  - executable program



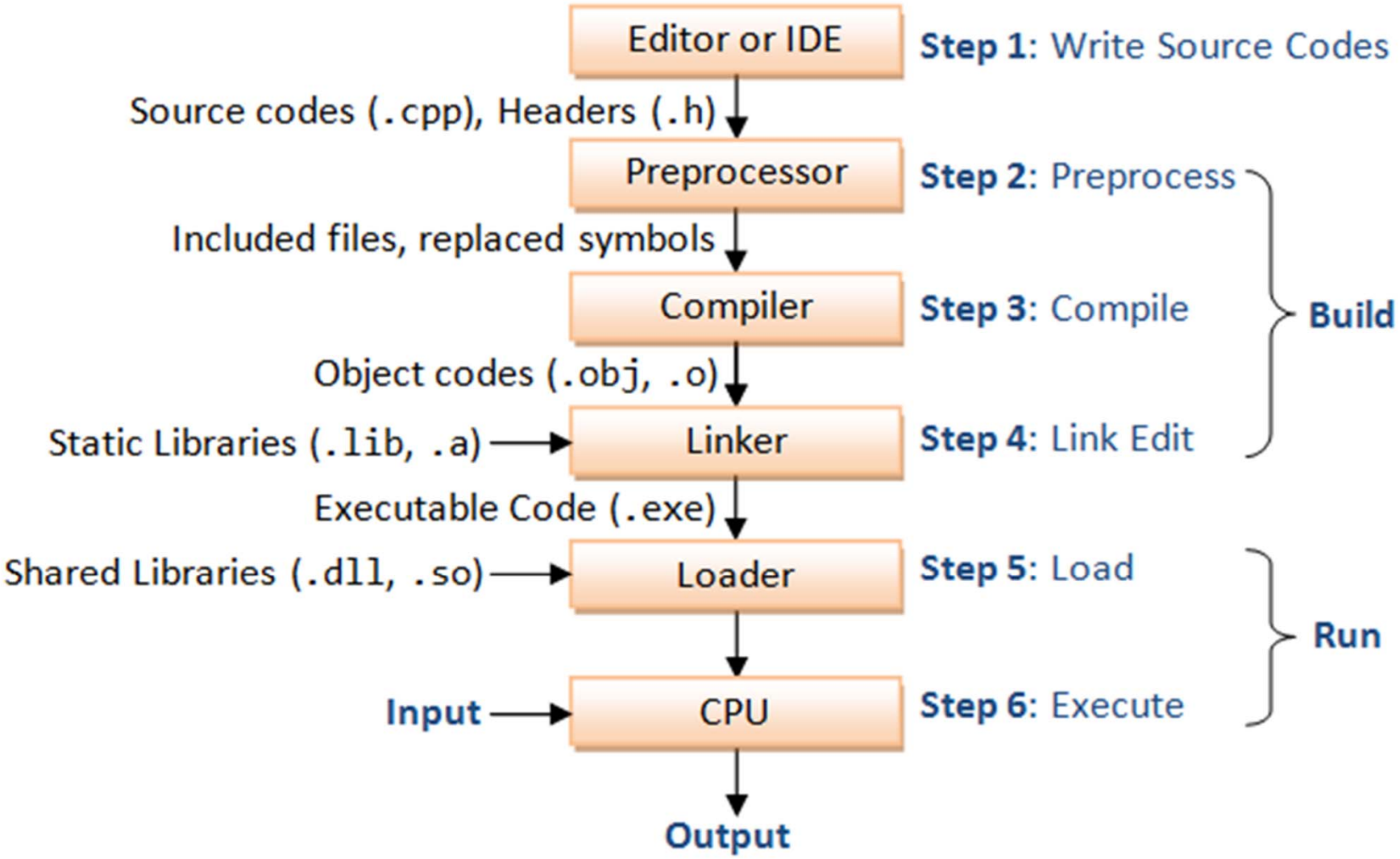
A linker or link editor is a computer program that takes one or more object files generated by a compiler and combines them into a single executable program.

The compiler is a complex program which convert the instructions from source language into machine language (assembler code).



- The result is an object program.
- If the link-editor is included in compiler then the result is an executable file.

The Process of Writing a C Program



The main properties of C programming language

1. Portability
2. Data types
3. Errors control
4. Work at assembler level
5. Few keywords
6. Structured language
7. Programmers' language

1. Portability of C programming language

- According to experienced software engineers, the C programming language seems to be the most portable support for a designed program.

2. Data types

- four basic arithmetic type specifiers:
  - char
  - int
  - float
  - double

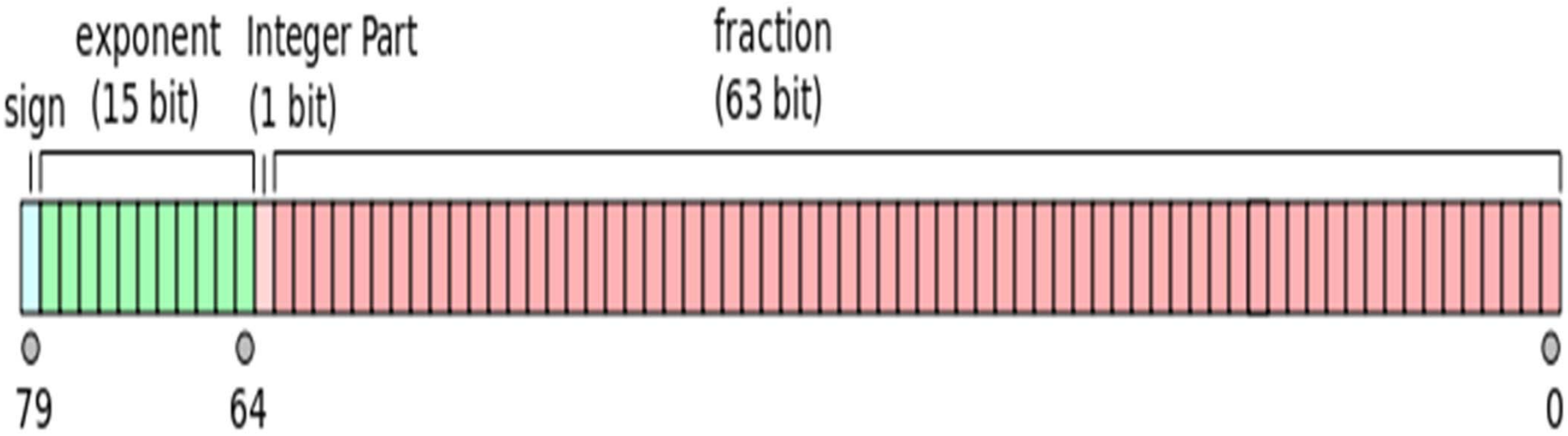
- optional specifiers:
  - signed,
  - unsigned
  - short
  - long

Type	Explanation
char	Smallest addressable unit (8 bits) that can contain basic character set. It is an integer type. Actual type can be either signed or unsigned depending on the implementation.
signed char	Same size as char, but guaranteed to be signed
unsigned char	Same size as char, but guaranteed to be unsigned
short **short int** **signed short** **signed short int**	*short* signed integer type. At least 16 bits in size
unsigned short **unsigned short int**	Same as short, but unsigned
int **signed int**	Basic signed integer type. At least 16 bits in size
unsigned **unsigned int**	Same as int, but unsigned

Type	Explanation
long **long int** **signed long** **signed long int**	*long* signed integer type. At least 32 bits in size
unsigned long **unsigned long int**	Same as long, but unsigned
long long **long long int** **signed long long** **signed long long int**	*long long* signed integer type. At least 64 bits in size (specified since the C99 version of the standard).
unsigned long long **unsigned long long int**	Same as long long, but unsigned (specified since the C99 version of the standard).

Type	Explanation
float	Single-precision floating-point format is a computer number format that occupies 4 bytes (32 bits) in computer memory and represents a wide dynamic range of values by using a floating point.
double	Double-precision floating-point format is a computer number format that occupies 8 bytes (64 bits) in computer memory and represents a wide dynamic range of values by using floating point.
long double	Extended precision floating-point type. Unlike types float and double, it can be either 80-bit floating point format, or IEEE 754 quadruple-precision floating-point format if a higher precision format is provided.

long double



The 80-bit floating point format was widely available by 1984 after the development of C and similar computer languages, which initially offered only the common 32- and 64-bit floating point sizes.

Notes:

- The actual size of integer types varies by implementation.
- The standard only requires size relations between the data types and minimum sizes for each data type.
- the **long long** is not smaller than **long**, which is not smaller than **int**, which is not smaller than **short**.

Notes:

- **char** size is always the minimum supported data type, all other data types can't be smaller.
- The minimum size for **char** is 8 bit, the minimum size for short and **int** is 16 bit, for **long** it is 32 bit and **long long** must contain at least 64 bit.
- Many conversions are possible in C.

3. Errors control

- Excepting syntax errors there are no other control

- There are no control over dimensions of variables, pointers, etc.

4. Work at assembler level

- There is the possibility to work directly with bits, octets, words and pointers.
- C instructions require a minimum number of processor instructions when are compiled.

5. C Language 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

C89 Standard

- There are only 32 key-words on ANSI C standard:
  - 27 from Kernighan & Ritchie book

- Other languages have at least twice more keywords.

C99 adds five more keywords:

_Bool

_Complex

_Imaginary

inline

restrict

C11 adds seven more keywords:

_Alignas

_Alignof

_Atomic

_Generic

_Noreturn

_Static_assert

_Thread_local

6. Structured language

- Structured programming is a programming paradigm aimed on improving the clarity, quality, and development time of a computer program by making extensive use of subroutines, block structures and for and while loops

- This is in contrast to using simple tests and jumps such as the **goto** statement which is both difficult to follow and to maintain.

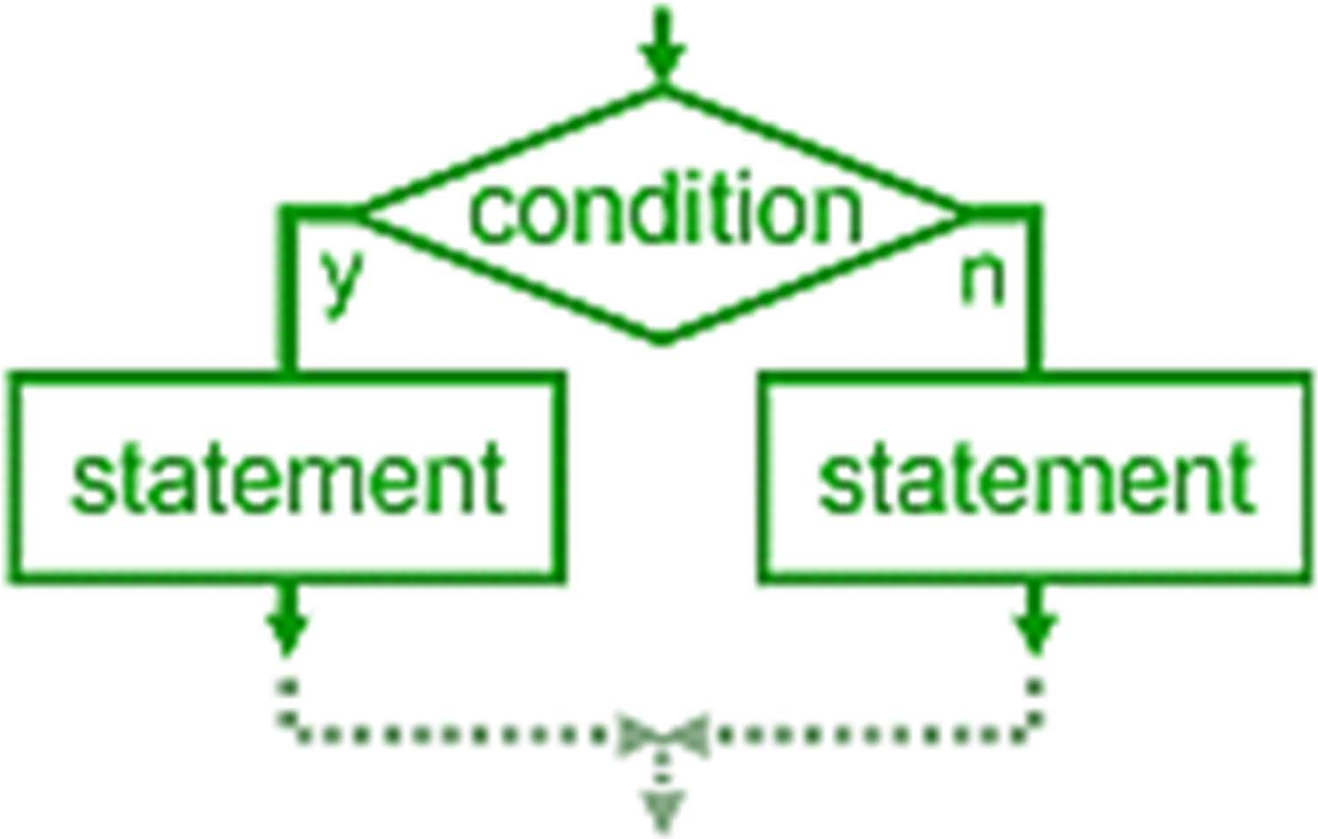
Notes:

- Structured programs are often composed of simple, hierarchical program flow structures.

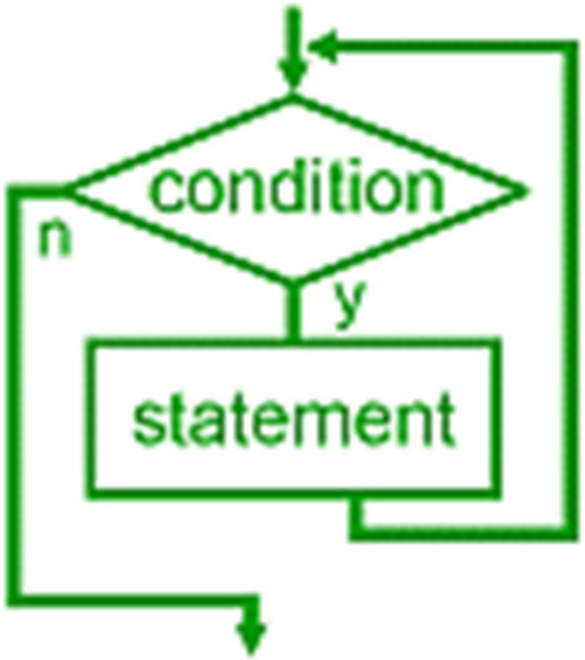
- These are sequence, selection, and repetition

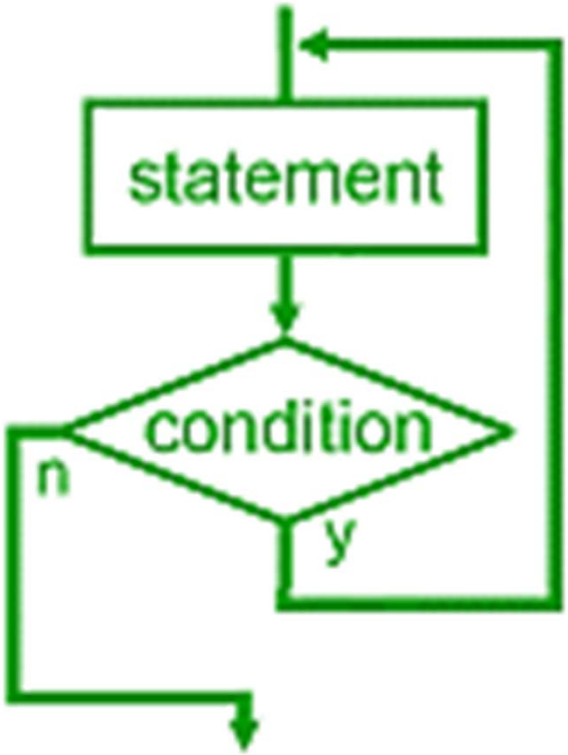
sequence

selection



repetition





Note:

- **compartmentalization** – the facility of splitting and hiding (from the rest of program) the whole information and instructions that are necessary to fulfill a particular task

- This is a characteristic of C

Note:

- The main structural component in C is the function concept.

- The main possibility to obtain the compartmentalization is to use a block of instructions grouped by special brackets (accolade)

```
{
.....
}
```

7. Programmers' language

- C is often used for "system programming", including implementing operating systems and embedded system applications.

- An active programmer needs:
  - code portability and efficiency
  - ability to access specific hardware addresses
  - ability to pun types to match externally imposed data access requirements
  - low run-time demand on system resources

- C is sometimes used as an intermediate language by implementations of other languages.

The structure of a C program

- global statements:
  - inclusions of header files
  - statements of constants and global variables
  - declarations of local functions
- function **main()**
- other functions

Notes:

- **keywords** are written with lowercases
- a C program must contain a single **main** function, and only one.
- the C standard library and C++ standard library traditionally declare their standard functions in **header files**.

Header file

- Each header file contains one or more function declarations, data type definitions, and macros.

Note: New header files were always added when a newer improved standard was released

Some basic header files

<stdio.h>	Defines core input and output functions
<stdlib.h>	Defines numeric conversion functions, pseudo-random numbers generation functions, memory allocation, process control functions
<string.h>	Defines string handling functions.
<math.h>	Defines common mathematical functions.

C preprocessor

- The preprocessor provides the ability for:
  - **inclusion of header files**
  - **macro expansions**
  - **conditional compilation**

Including files

```
#include <stdio.h>
int main(void)
{       printf("Hello, world!\n");
        return 0;
 }
```

- The preprocessor replaces the line #include <stdio.h> with the text of the file 'stdio.h', which declares the printf() function among other things

Statements and macro definition

- Define a constant:

```
#define PI 3.14159
```

- Define a macro function:

```
#define ABS(a)   (a<0) ? –a : a
```

Conditional compilation

- Conditional compilation allows the compiler to produce differences in the executable according with some parameters.

- This technique is commonly used when these differences are needed to run the software on different platforms, or with different versions of required libraries or hardware.

if-else directives

- The if-else directives:

  #if

  #ifdef

  #ifndef

  #else

  #elif

  #endif

  can be used for conditional compilation.