C Programming Essentials

Unit 1: Sequential Programming

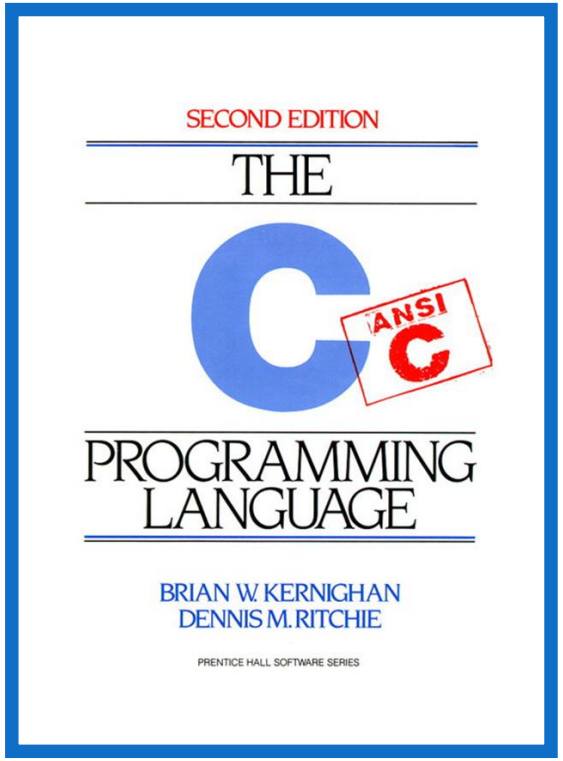
CHAPTER 1: INTRODUCTION

DR. ERIC CHOU

IEEE SENIOR MEMBER

LECTURE 1

Introduction



About This Course

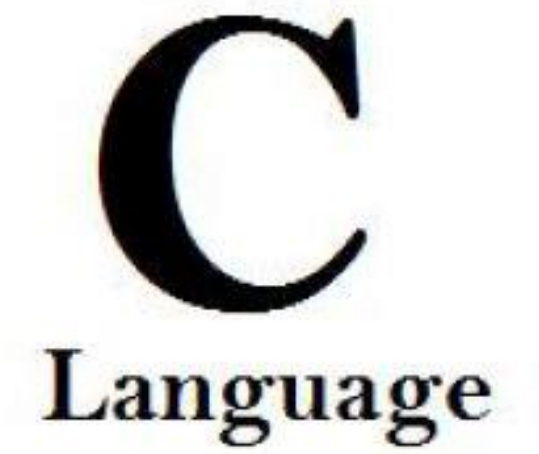
Instructor: Dr. Eric Chou

Textbook:

The C Programming Language

by Kernighan, Ritchie. Second Edition

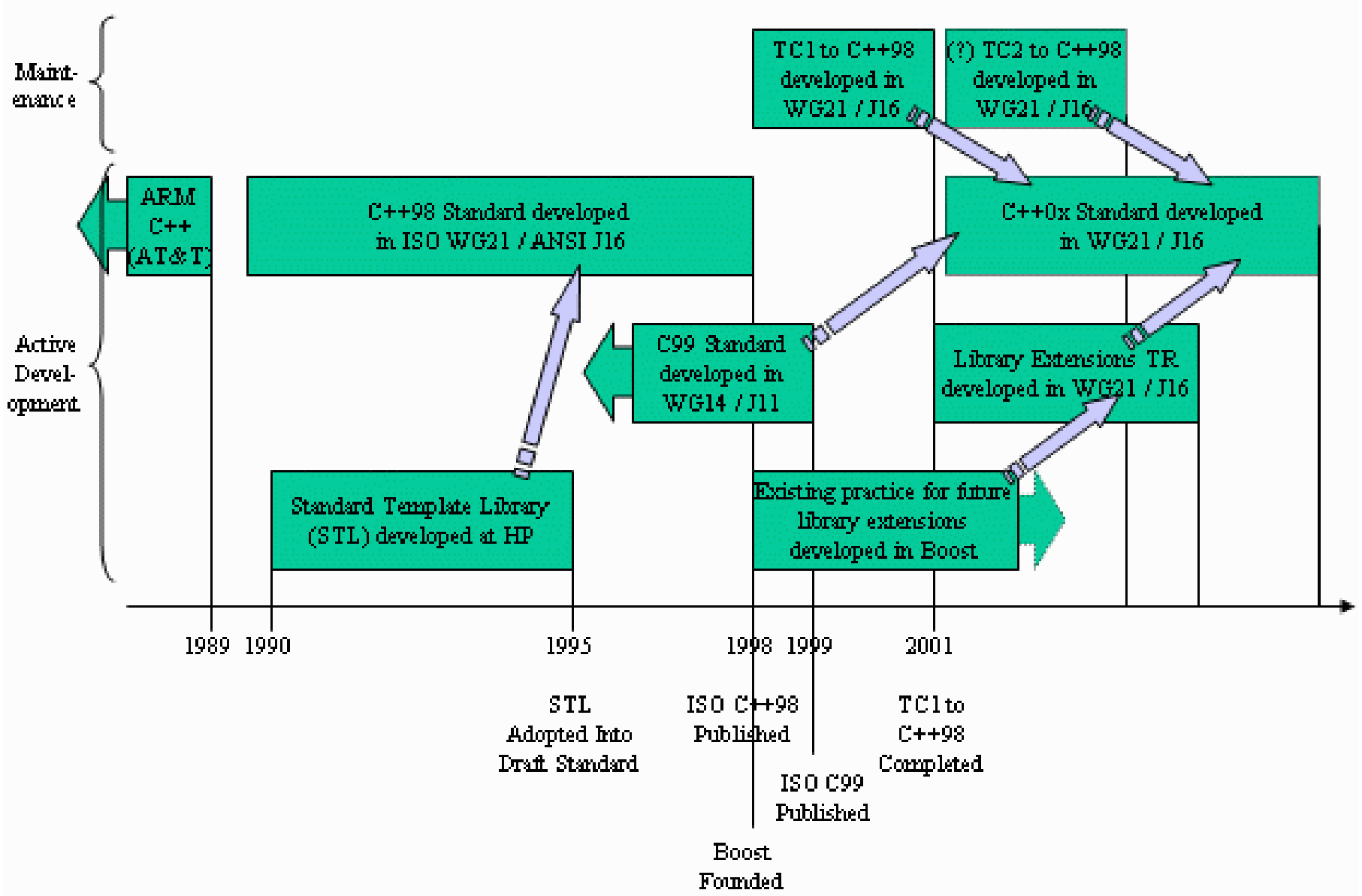
Topics

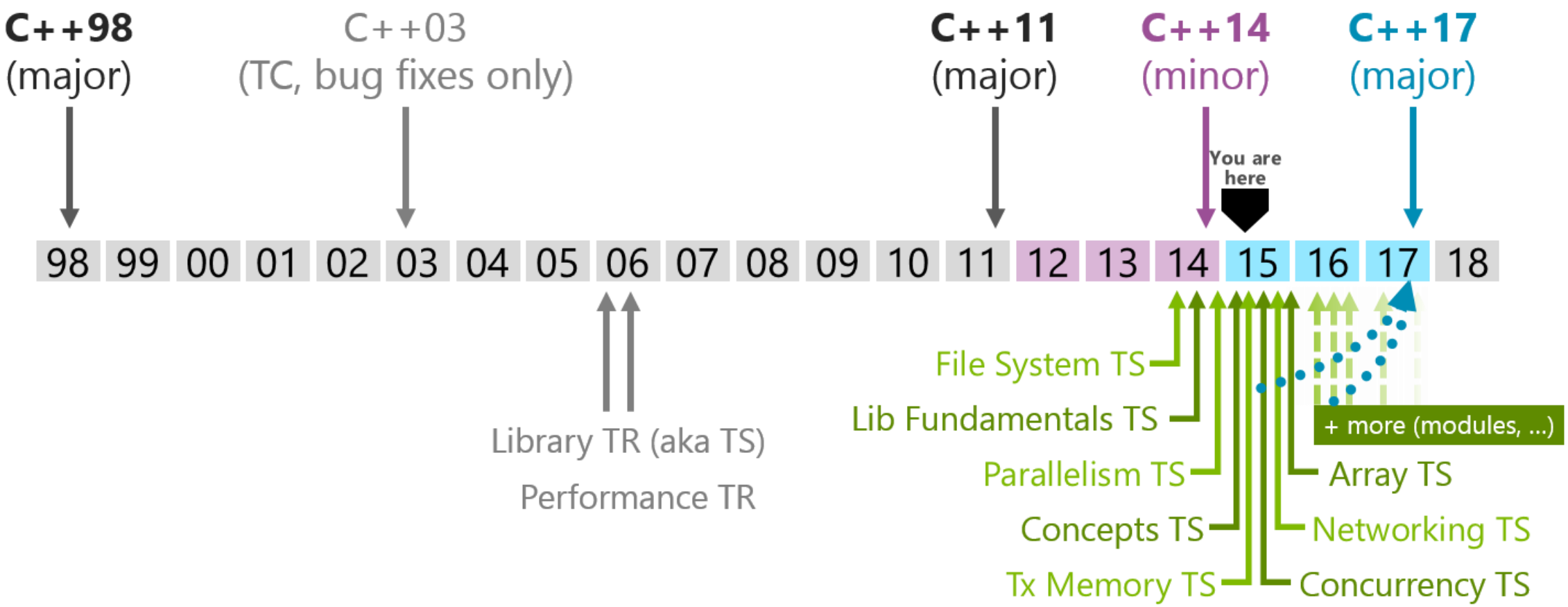


- Introduction to Programming in C
- Data Type, Operators, Basic I/O
- Conditional Expressions, Control Flow
- Loops
- Functions, Program Structure and Recursion
- Pointer and Arrays
- Structure, Union and Enum
- Dynamic allocation
- Storage Classes
- Pre-processor, File Handling, Exceptions, Math library
- Basic Algorithms: Searching, Sorting

Standard C

- Standardized in 1989 by ANSI (American National Standards Institute) known as ANSI C
- International standard (ISO) in 1990 which was adopted by ANSI and is known as *C89*
- As part of the normal evolution process the standard was updated in 1995 (*C95*) and 1999 (*C99*)
- C++ and C
  - C++ extends C to include support for Object Oriented Programming and other features that facilitate large software development projects
  - C is not strictly a subset of C++, but it is possible to write "*Clean C*" that conforms to both the C++ and C standards.


Maint-
enance
Active
Devel-
opment
ARM
C++
(AT&T)
C++98 Standard developed
in ISO WG21 / ANSI J16
TC1 to C++98
developed in
WG21 / J16
(?) TC2 to C++98
developed in
WG21 / J16
C++0x Standard developed
in WG21 / J16
C99 Standard
developed in
WG14 / J11
Library Extensions TR
developed in WG21 / J16
Standard Template Library
(STL) developed at HP
Existing practice for future
library extensions
developed in Boost
1989
1990
1995
1998
1999
2001
STL
Adopted Into
Draft Standard
ISO C++98
Published
ISO C99
Published
TC1 to
C++98
Completed
Boost
Founded


C++98
(major)
C++03
(TC, bug fixes only)
C++11
(major)
C++14
(minor)
C++17
(major)
You are here
98 99 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18
Library TR (aka TS)
Performance TR
File System TS
Lib Fundamentals TS
Parallelism TS
Concepts TS
Tx Memory TS
Array TS
Networking TS
Concurrency TS
+ more (modules, ...)

Elements of a C Program

- A C development environment includes
  - *System libraries* and *headers*: a set of standard libraries and their header files. For example see `/usr/include` and `glibc`.
  - *Application Source*: application source and header files
  - *Compiler*: converts source to object code for a specific platform
  - *Linker*: resolves external references and produces the executable module
- User program structure
  - there must be one main function where execution begins when the program is run. This function is called main
    - `int main (void) { ... },`
    - `int main (int argc, char *argv[]) { ... }`
    - UNIX Systems have a 3rd way to define main(), though it is not POSIX.1 compliant
      `int main (int argc, char *argv[], char *envp[])`
  - additional local and external functions and variables

About C

- GNU : GNU's Not Unix, We use MinGW version of gcc.
  - GNU C: gcc is a standard compiler
- C is non-portable (Suitable for machine-native code development)
  - Terms: Compiler (human -> machine [once]), Interpreter (instructions -> machine [each time the program is run])
- C is a high level language
  - One line in c maps to many lines of assembly code

Programming on Windows

DOS command line: GNU-C

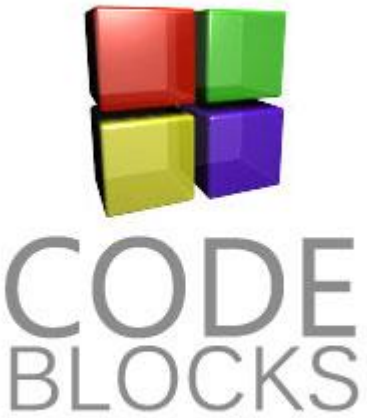
- Text based editors: Notepad++
- MS-DOS/Window command line script

IDE

- Code::Blocks

  http://www.codeblocks.org/
- Eclipse

  https://eclipse.org/cdt/

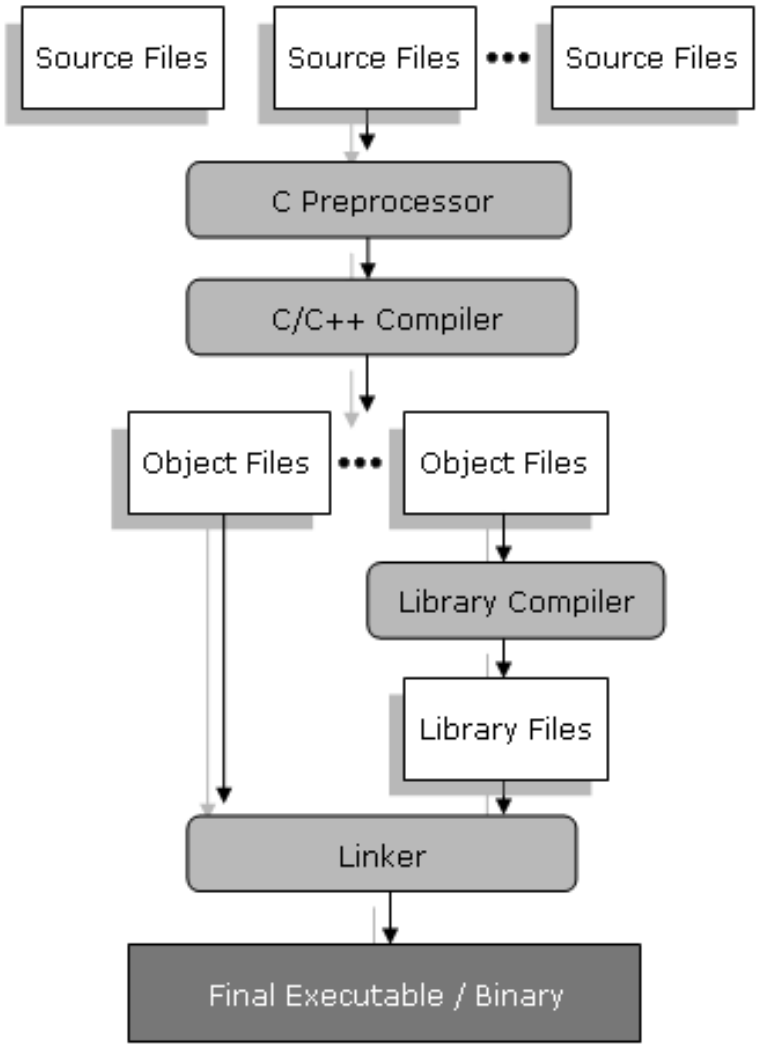


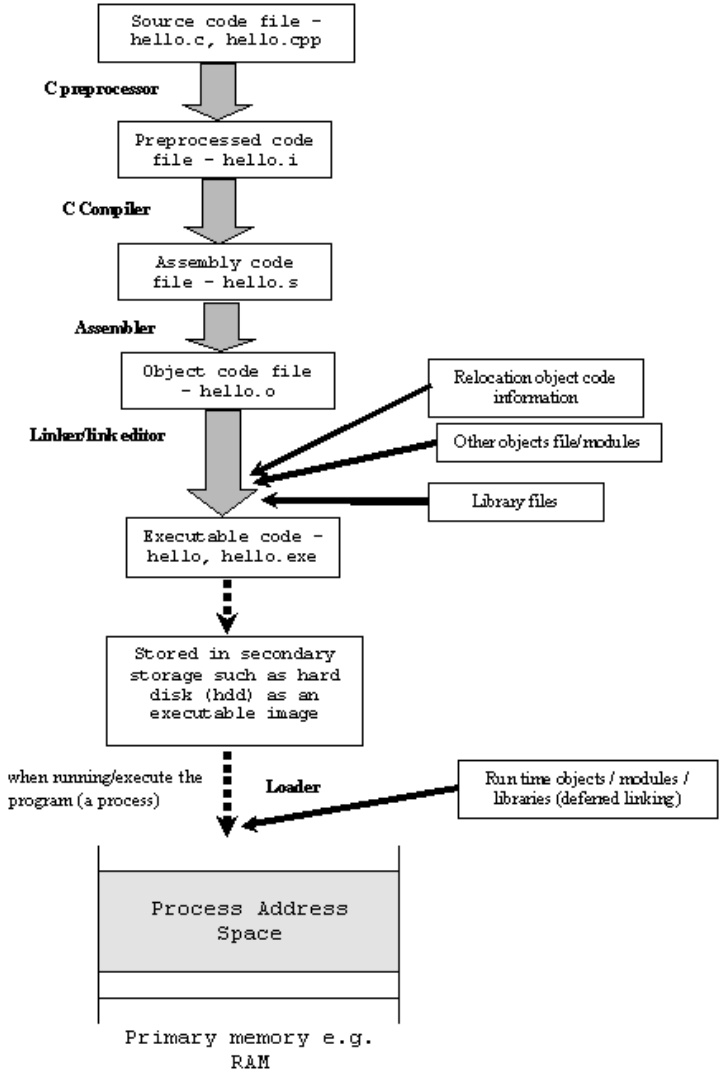
LECTURE 2

My First C Program in DOS Command Line Compilation

My first C program!

```
/* thou shalt begin from somewhere*/
#include <stdio.h>
// program prints hello world
int main() {
  printf ("Hello world!\n");
  return 0;
}
```


Source Files
Source Files
Source Files
C Preprocessor
C/C++ Compiler
Object Files
Object Files
Library Compiler
Library Files
Linker
Final Executable / Binary


Source code file - hello.c, hello.cpp
C preprocessor
Preprocessed code file - hello.i
C Compiler
Assembly code file - hello.s
Assembler
Object code file - hello.o
Relocation object code information
Other objects file/modules
Library files
Linker/link editor
Executable code - hello, hello.exe
Stored in secondary storage such as hard disk (hdd) as an executable image
when running/execute the program (a process)
Loader
Run time objects / modules / libraries (deferred linking)
Process Address Space
Primary memory e.g. RAM

Files, directories and permissions

Directory
drwxr-xr-x 2 echou cse 4096 2008-08-13 22:46 Pictures

File
-rw-r--r-- 1 echou cse 3446 2008-08-14 15:16 test.c

Special files (advanced)

- .a : static library
- .so : shared object code (dynamic)
- .o : object code
- Pipes : fifo / buffered prwx--x--x
- Device files : /dev/cdrom etc.

Programming on Windows

Writing programs

- Use any editor (graphical, console)
- Save file as <filename>.c

Compiling programs

- gcc <filename>.c gcc myfirst.c –o myfirst

Running programs

- ./a.out ./myfirst
  (executable files need to have executable permissions.
  $chmod +x <executable>) // for linux

Demo Program: myfirst.c

Go DOS Command!!!

LECTURE 3

My first C Program in DOS Command Line Compilation

Good Programming Practices

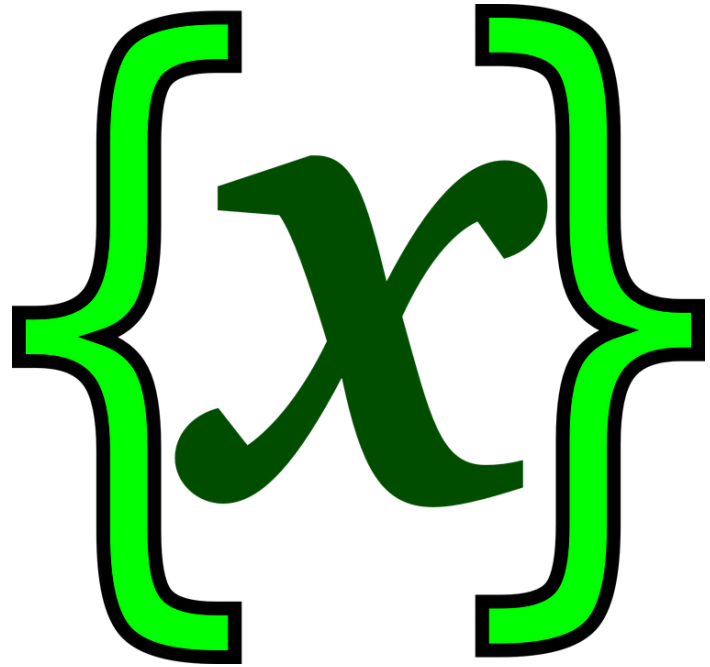
Variables names

- Not too short, not too long
- Always start variable names with small letters
- On work break
  - Capitalize: myVariable, OR
  - Separate: my_variable, OR
  - Lower Case: myvariable

Rules for naming a variable in C

- Only letters, digits and underscore_
- Start with letters or underscore only “[a-zA-Z_]”
- Case-sensitive
- No reserved words, no keywords, no known library file name or variable names.
- There is no rule on how long a variable can be. However, only the first 31 characters of a variable are checked by the compiler. So, the first 31 letters of two variables in a program should be different.

C is a strongly typed language. What this means it that, the type of a variable cannot be changed.



Good Programming Practices

Put comments

```
#include <stdio.h>
int main() {
   /* this program adds
   two numbers */
   int a = 4; //first number
   int b = 5; //second number
   int res = 0; //result
   res = a + b;
   printf("%d + %d = %d\n", a, b, res);
   return 0;
}
```

A Simple C Program with Comments

```
/* Comment Block */
// comment line
```

- begin with /* and end with */ indicating that these two lines are a comment.
- You insert comments to document programs and improve program readability.
- Comments do not cause the computer to perform any action when the program is run.

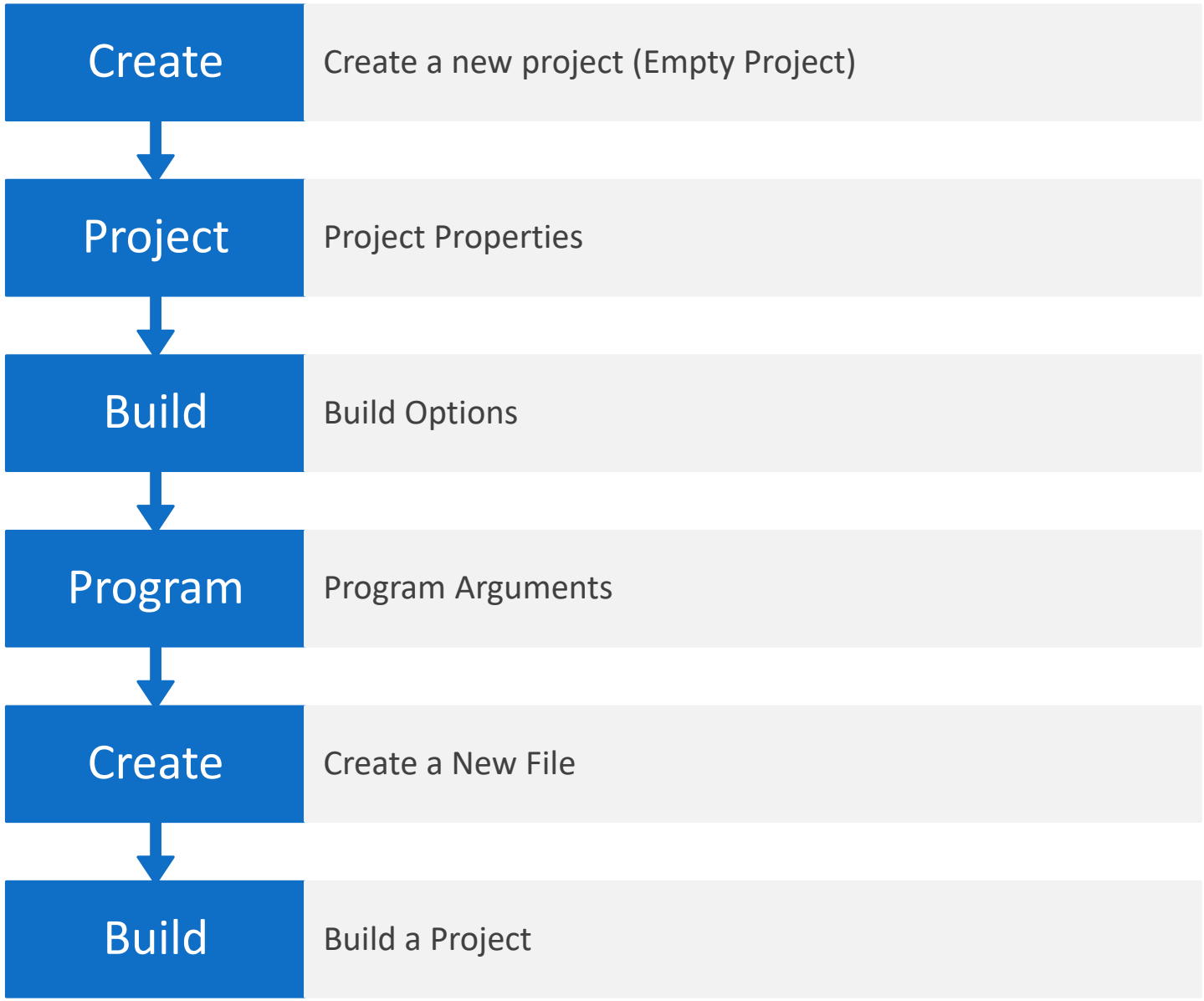
Good Programming Practices

- Your code may be used by somebody else
- The code may be long
- Should be easy to understand for you and for others
- Saves lot of errors and makes debugging easier
- Speeds up program development
- Put code of a certain functionality (module) in a file.

LECTURE 4

Code::Blocks Project Building

Code::Blocks Project Building



Demo: Create myfirst in Code::Blocks

Go Code::Blocks!!!