C Programming I: Lecture I

Andreas Heinz
FYD400 - HT 2019

Overview of Lecture 1

- **Computers and programming**
- **Programming in general**
- **C: history, development and properties**
- **C: basics**
  - **"Hello World!"**
  - **Structure of a C program**
  - **Simple I/O**

Reading: VtC chapter 2,3

> ***We live in a society absolutely dependent on science and technology and yet have cleverly arranged things so that almost no one understands science and technology. That's a clear prescription for disaster.***
>
> ***Carl Sagan***

List of Prerequisites

None

=> I will start from the very beginning!

What is a Computer?

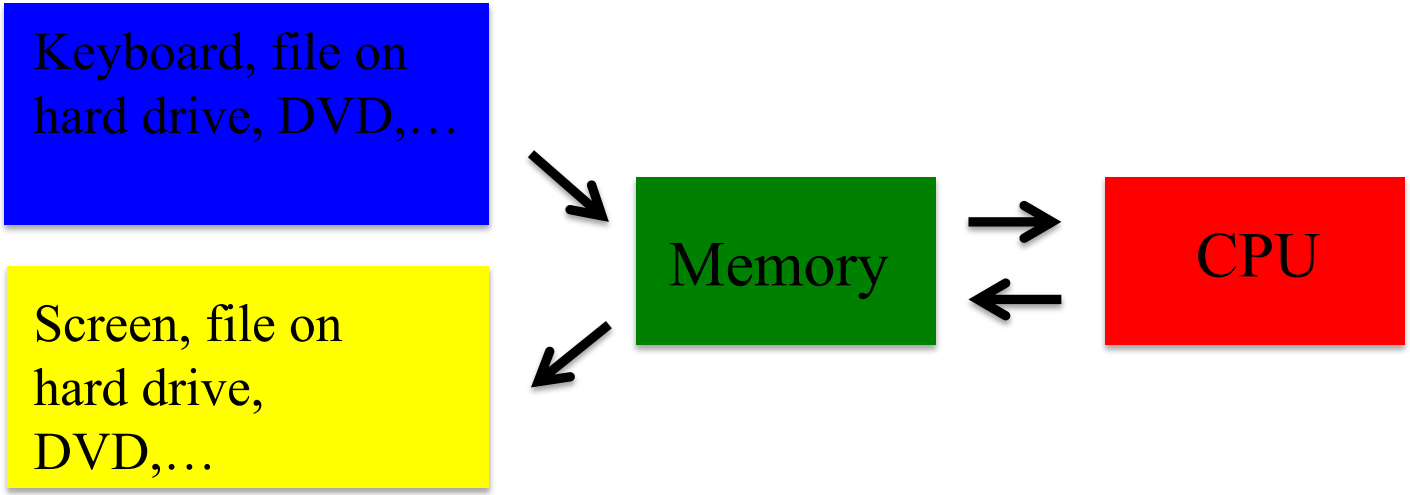
Computers

- "Definition" of a computer:
  - A device, which can manipulate data according to the instructions of a program.
  - Early programming: mechanical, then wires, relays and transistors
  - Now: integrated circuits

What is a computer made of?

Computers

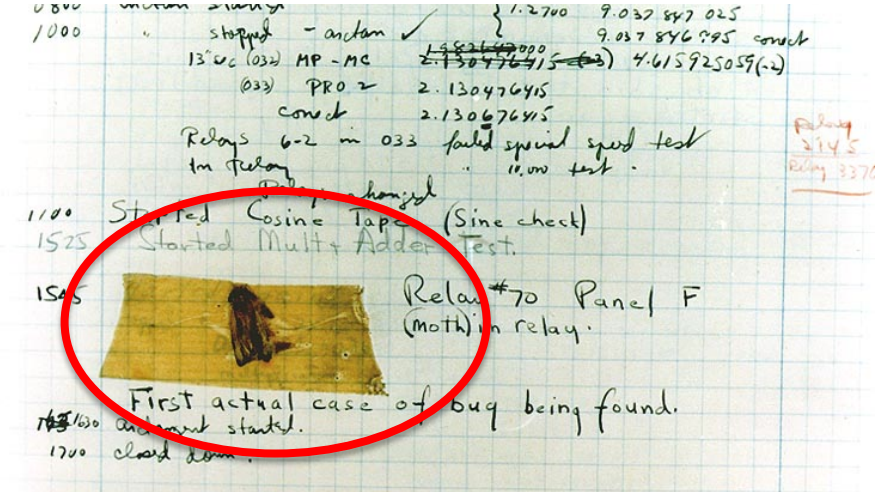
- "Definition" of a computer:
  - A device which can **manipulate data** according to the instructions of a **program**.
  - Early programming: mechanical, then wires, relays and transistors
  - Now: integrated circuits
  - von Neumann: **program** and **data** are both stored in the **memory**



Types of Computers

- **Supercomputer: IBM, Cray,… (originally ever more powerful processors, now massive parallel computers)**
- **Computer cluster: PC, PS-X, … (concept of modern supercomputers)**
- **Personal computer: PC, Mac, …**
- **Embedded systems (microprocessors): cars, refrigerators, data acquisition systems…**
  => need programming: hardware drivers

Computer Programming



- **Computer program: interface between computer and human.**
- **Many different programming languages.**
- **"Computer errors" are rare – usually the human is at fault.**
- **Computers are digital devices (i.e. 0 and 1 only – corresponds to on/off) -> major advantage!**
- **Computers have a limited set of logic operations**
- **Programming languages allow us to translate our intention into commands the computer hardware can understand.**
- **We can do a lot of programming without knowledge of the computer hardware.**

Computer Programs

- **"Definition" of a computer program?**

  Set of instructions which allows the user to control the computer; typically for the manipulation of data

- **Platform: operating system + hardware (e.g. Windows + PC)**
- **Integrated Developer Environment (IDE):**

  LabWindows/CVI, Visual Studio, …

  This is where to type in the code!

  (Note: there are alternatives to IDEs!)

How to Create a Program?

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

IDE

Source code: file.c

Compiler

```
0001001110001010100011000001111 00
0111001010100011110010101001001 01
```

Object file: file.obj

Library

Linker

```
0001001110001010100011000001111 00
0111001010100011110010101001001 01
1000011000111111110011111101011 10
0100110001111010101010011110101 011
```

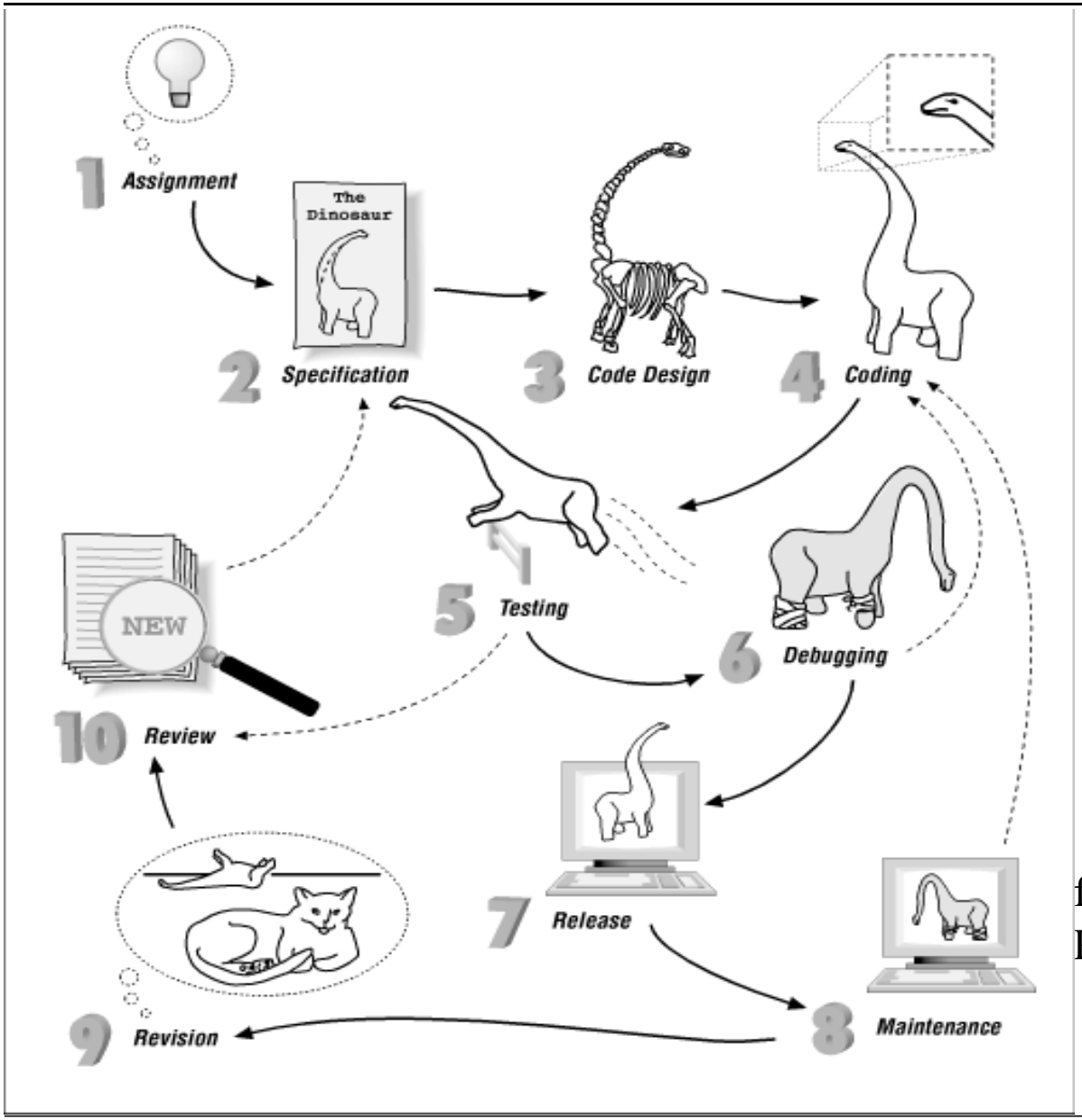
Program (executable file): file.exe

Programming and Planning

The problem: what, how, for whom, when?

1. **Define the problem.**
2. **Outline the solution.**
   -> flow chart
3. **Choose/design an algorithm.**
   -> model of the program
   -> a bare set of instructions
4. **Select a language, compiler, integrated developer environment,… (does not change for every program)**
5. **Convert the algorithm into a program.**
6. **Check the program!**

Life Cycle of a Program



More realistic picture of programming.

from “Practical C Programming” by S. Oualline

Programming

- **Pay attention to detail! Computers are not forgiving and C compilers are (intentionally) far from being foolproof.**

- **"Computer errors" are usually human errors made by either the user or the programmer (or both).**

- **Most important rule:**

C

- **Why C?**
  - **Widely distributed language that allows for fast programs**
  - **"many-level language" – gives great freedom (and responsibility…)**
  - **"small"**

> There is no programming language, no matter how structured, that will prevent programmers from writing bad programs.
>
> L. Fon

- **Why not C?**
  - **Not safe (unless you work for it)**
  - **Old – does not include object-oriented programming**
  - **Learn C++, JAVA or C# directly**

- **Course: C & NI LabWindows/CVI (VI = virtual instrument)**
  - **Integrated developer environment (IDE) using ANSI C**
  - **Graphic interface (and instrument control)**

C: History and Development

- C's: predecessors and goals
  - Algol 60 -> BCPL -> B ->(NB).. C … ca. 1972 Bell Labs (Dennis Ritchie)
  - Took OS UNIX from assembler to a "higher" programming language (1973).
  - Goal:
    - Simplify programming (higher level of abstraction)
    - Separate out the architecture-dependent parts of UNIX
- Standards:
  - ANSI C (1989) (American National Standards Institute), ISO C 99
  - C99
  - C11
- Development
  - Additional features while maintaining backwards compatibitlity
  - New architecture – C widely distributed
  - New languages "based on" C: C++ (pure extension), C#, Java, …

C: “Hello World!”

Create a file, e.g. code.c in an editor – this is your source code

```
/* printing program */

#include <stdio.h>


int main (void)        // sometimes just int main ()

{
        printf (”Hello\n World!\n”);        /* that’s correct */

}
```

To standard output:

```
Hello
  World!
```

Note: C is case sensitive!

The shortest C program

```
main() {}
```

- **complete**
- **correct**
- **but … does nothing**

C: "Hello World!"

```
/* printing program */
#include <stdio.h>

int main (void) // sometimes just int main ()
{
        printf ("Hello\n World!\n");      /* that's correct */
        return 0;
}                                  // classic example
```

Comments: // comment is the rest of the line (in C99)
or comment is enclosed by /* */

- no nested comments
- very important for documentation, readability, and you passing this course!

To be avoided!

```
v,i,j,k,l,s,a[99];
main()
{
        for(scanf("%d”,&s);*a-s;v=a[j*=v]-
a[i],k=i<s,j+=(v=j<s&&(!k&&!!printf(2+"\n\n%c"-(!l<<!j),"
#Q"[l^v?(l^j)&1:2])&&++l||a[i]<s&&v&&v-i+j&&v+i-
j))&&!(l%=s),v||(i==j?a[i+=k]=0:++a[i])>=s*k&&++a[--i])
                ;
}
```

International Obfuscated C Code Contest
“Best small programs” winner 1990: Doron Osovlanski
Program prints all solutions to the 8 queens problem
http://www.ioccc.org

C: “Hello World!”

```
/* printing program */
#include <stdio.h>

int main (void) // sometimes just int main ()
{
        printf (”Hello\n World!\n”);      /* that’s correct */
        return 0;
}                                                  // classic example
```

Pre-processor directive:

- start with “**#**”
- do **NOT** end with “**;**”
- “prepare” the source code for the compiler

Here:

- statement includes standard input/output library functions
- <name.h> vs. “name.h “

C: “Hello World!”

```
/* printing program */

#include <stdio.h>

int main (void)                    // sometimes just int main ()

{
        printf (”Hello\n World!\n”);   /* that’s correct */
        return 0;

}                                  // classic example
```

Function:

- **“int main (void){ …}” - program starts here!**
- **all programs have ONE main function**
- **(almost) everything happens inside functions**
- **function header and function body {…} consisting of statements and expressions**

C: “Hello World!”

```
/* printing program */

#include <stdio.h>


int main (void)                                  // sometimes just int main ()

{
        printf (”Hello\n World!\n”);       /* that’s correct */
        return 0;

}                                                // classic example
```

Statements:

- **call function printf(“…”) – prints series of characters on standard output (in this case on the screen)**
- **“\n” is an escape sequence (see table in VtC)**
- **main function returns status code “0” and ends the program**

C: "Hello World!"

```
/* printing program */

#include <stdio.h>

#include <utility.h>


int main (void) // sometimes just int main ()

{
        printf ("Hello\n World!\n");      /* that's correct */
        while(!KeyHit());

        return 0;

}                                         // classic example
```

Contains function KeyHit()

"Run the program until a key is pressed"

One way to ensure the that window does not close right after execution of the program (in debug mode) Use *getchar()* in *<stdio.h>* as alternative.

Summary

- **Computers: Central Processing Unit (CPU) + Memory + Input/Output (I/O).**
- **Program and data are stored in memory.**
- **Computer program: set of instructions to manipulate data.**
- **Compiling + linking gives an executable.**
- **Structured programming: proper algorithm, implement, comment, test, debug, debug, debug…**
- **History of C – related to the history of UNIX/LINUX.**
- **"Hello World!" – a first program.**