

Introduction to Programming and Computational Physics

Lecture 1

Algorithms

Programming languages

Operating systems

Shells

The first C program

What is an algorithm?

A well-ordered and finite set of non-ambiguous and computable operations that leads to a result and terminates in a finite time

A program code is a realization of an algorithm which can be translated into an executable for a computer.

A well-written algorithm

The recipe for cooking 100 g of pasta:

- 1) Put 1 liter of water in a pot
- 2) Put the pot on to cook
- 3) Switch on the kitchen stove
- 4) Repeat step n.5 until the water starts to boil
- 5) Wait one minute
- 6) Add 10 g of salt
- 7) Read the cooking time on the pasta envelop
- 8) Put the pasta in the boiling water
- 9) Wait the time given at step n.7
- 10) Strain your pasta
- 11) End

A badly-written algorithm

\$\$\$

An algorithm to earn money at the Stocks Exchange:

- 1) If the stocks lowered in price in such a way that they can only increase their value... **BUY**
- 2) If the stocks increased in price in such a way that they can only reduce their value... **SELL**

What is wrong with it?

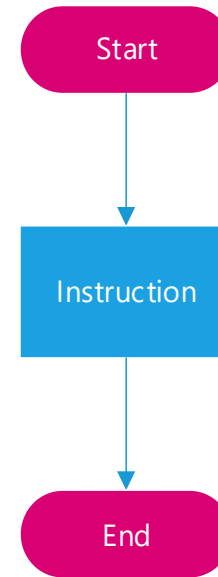
Why algorithms are so important?

Our aim is to build a sequence of primitive operations that can be *automated* → *algorithm*

A *program* is the realization of one or more algorithms with a sequence of primitive operations understood by the *executor*

Structured programming

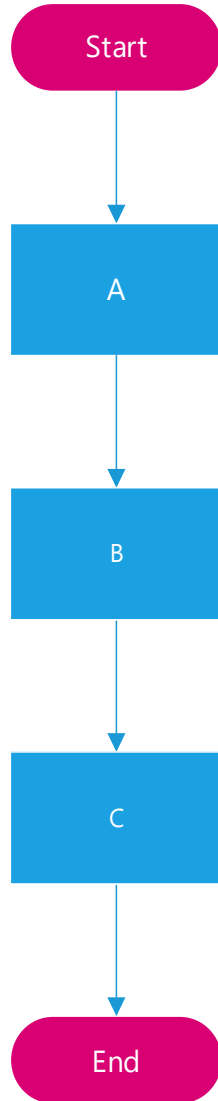
The idea is to execute all the instructions in a sequential way from the beginning to the end of program, with two only *exceptions* allowed, selection and iteration



Structured programming

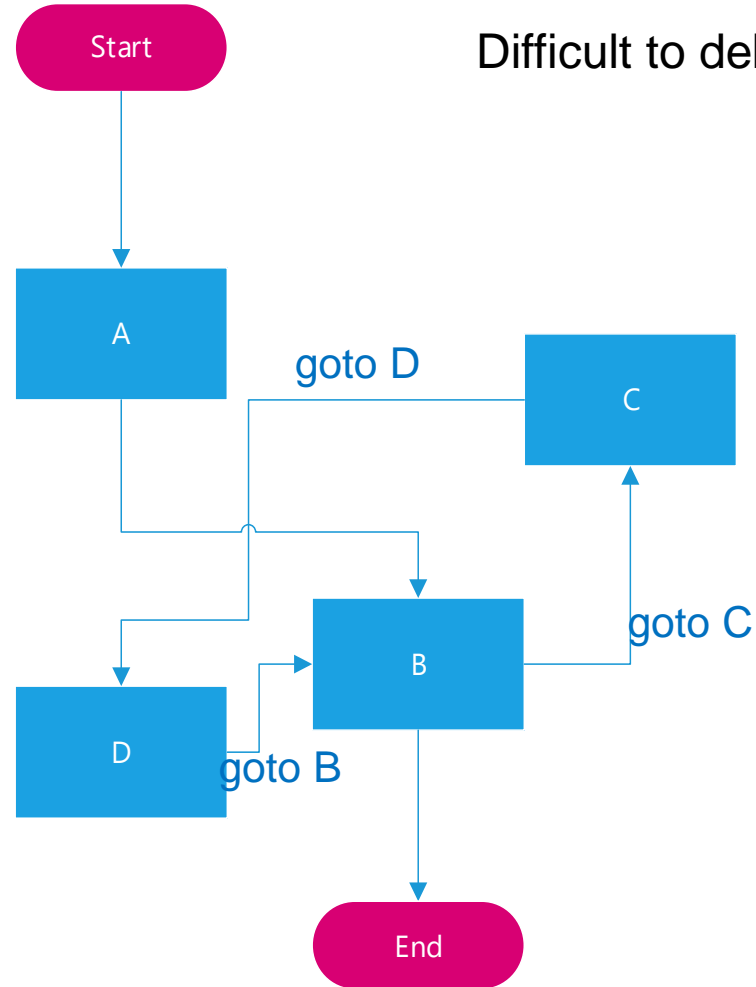
Structured
Clear main sequence

Easy to debug



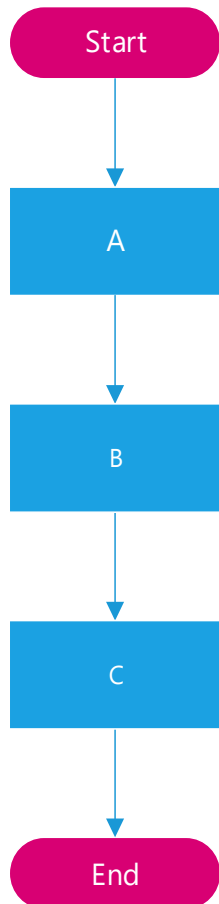
Not structured
sequence jump around

Difficult to debug

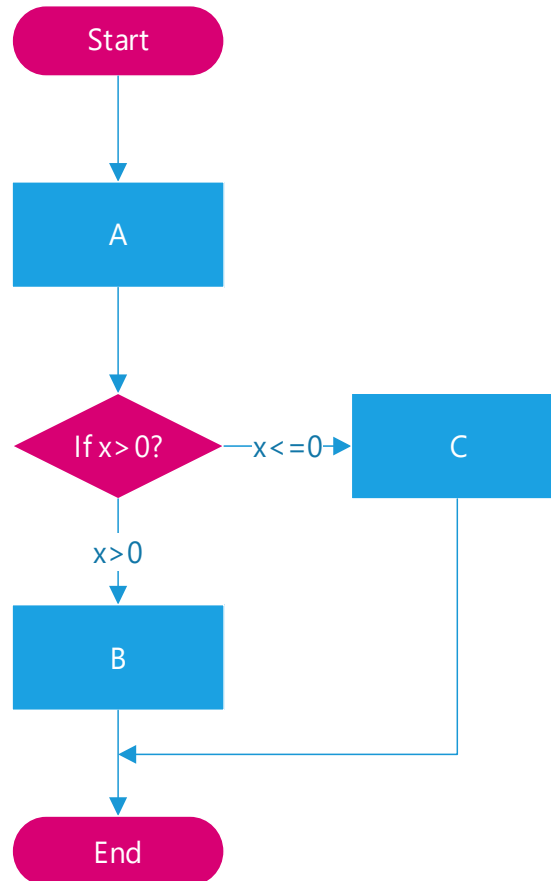


Structured programming

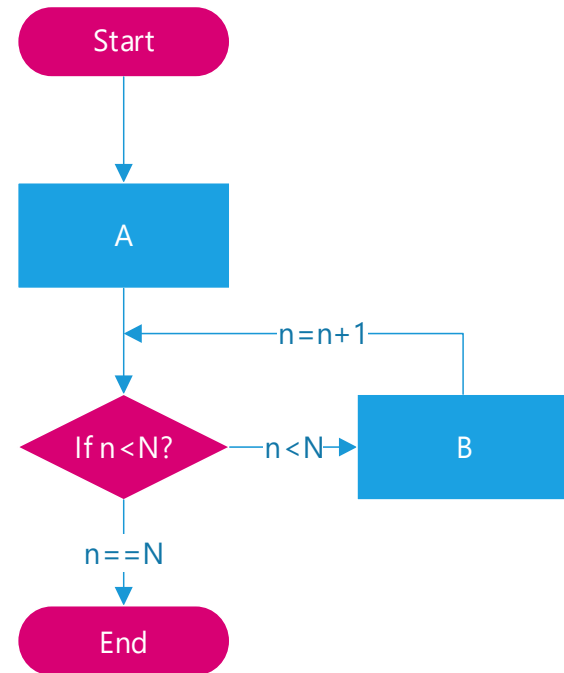
Sequence



Selection: **if, if else**



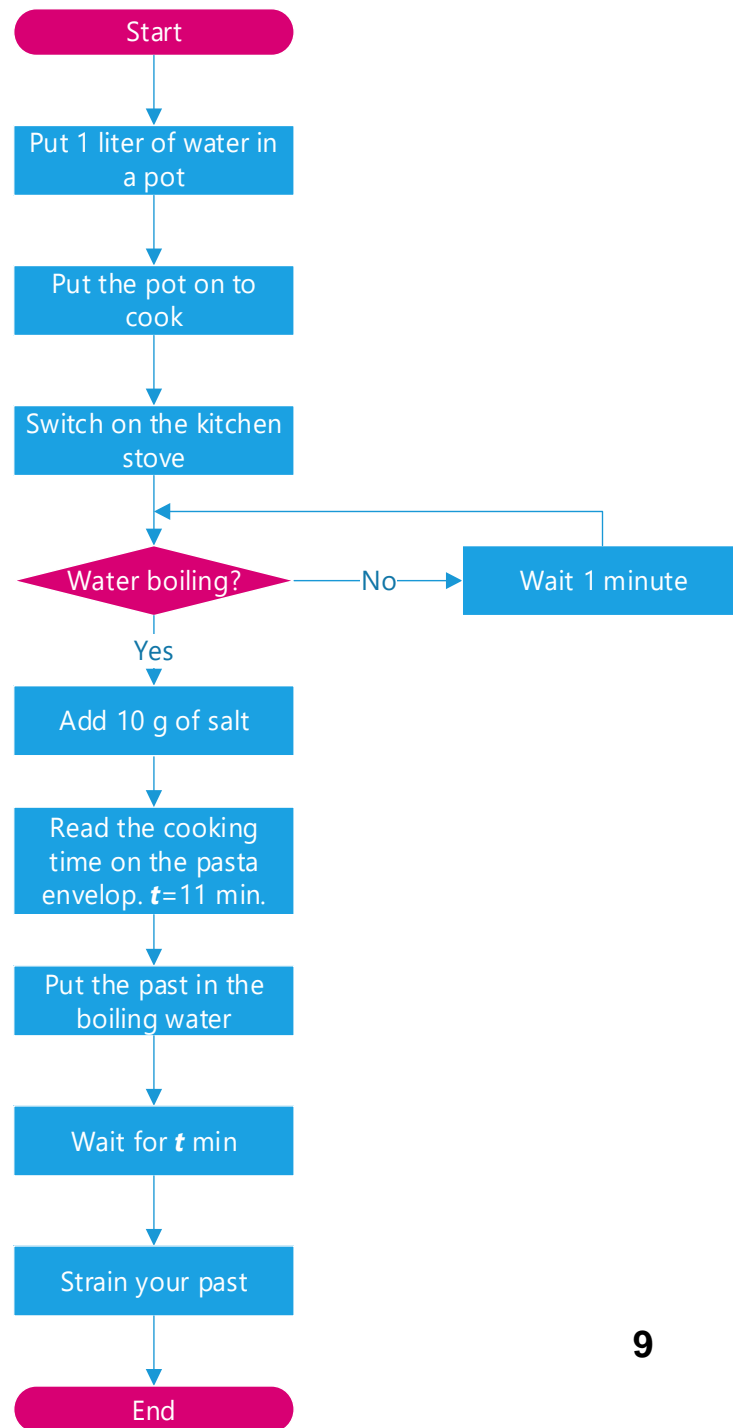
Iteration: **for, while**



A well-written algorithm

The recipe for cooking 100 g of pasta:

- 1) Put 1 liter of water in a pot
- 2) Put the pot on to cook
- 3) Switch on the kitchen stove
- 4) Repeat step n.5 until the water starts to boil
- 5) Wait one minute
- 6) Add 10 g of salt
- 7) Read the cooking time on the pasta envelop. $t=11$ min.
- 8) Put the pasta in the boiling water
- 9) Wait the time given at step n.7
- 10) Strain your pasta
- 11) End



Programming languages

An algorithm written in a *natural* language (English, German, Italian) can't be executed from a computer: we need a *formal* language. It must be a language provided with a set of rules in order to avoid any possible ambiguity.

A program is actually an algorithm written in a formal language.

The C language

1969:

[Ken Thompson](#) (Bell Telephone) wrote the B language: a first attempt to define a high-level language for operating system implementation.

1972:

[Dennis Ritchie](#) wrote an evolution of the B language: the C language. The UNIX operating system was almost entirely written in C.

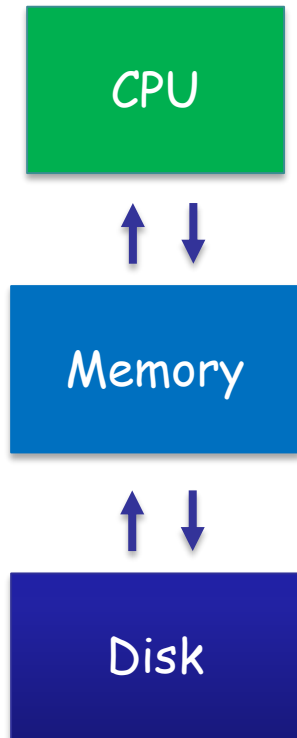
1973-1980:

The C language spreads to many other platforms. The first *libraries* are born and the first reference book is written in 1978: [Kernighan & Ritchie, "C Programming Language"](#).

1983 - 1999:

The American National Standards Institute defines the standard [ANSI C](#): a collection of rules to be followed by any C compiler.

What is a computer



Perform calculation

Process

An instance of computer program executed by a thread or by multiple threads.

Fast data read/write
Temporal within a process

Thread

A sequence of instructions.

Slow data read/write
Permanent

In our course, we will limit to a single thread programming.

CPU = brain



Memory = desk

Disk = shelves



Operating system

An **operating system (OS)** is a set of computer programs that manage the hardware and software resources of a computer. Its basic tasks are:

- Processing management
- Memory management
- Recognizing Input and sending Output
- Controlling peripheral drivers
- Networking

They provide a *software platform* on top of which other programs, called *application programs* can run

The most popular: Microsoft Windows

Unix/Linux

Mac OS X

Android



Shell

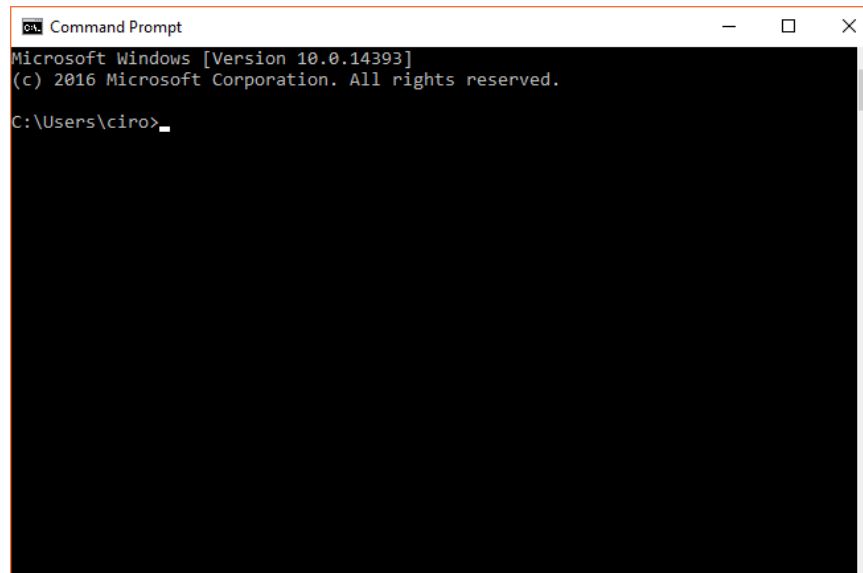
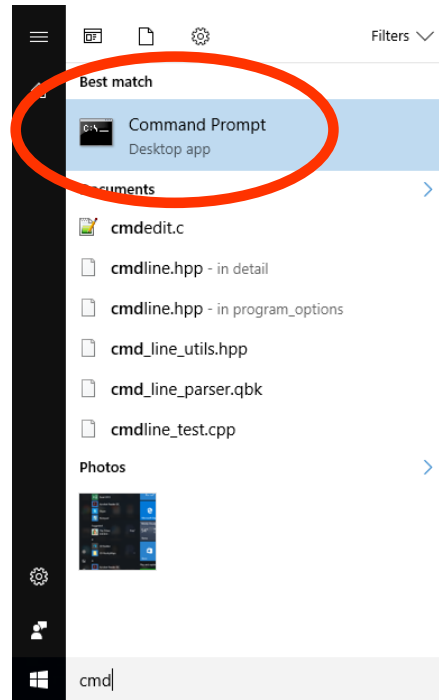
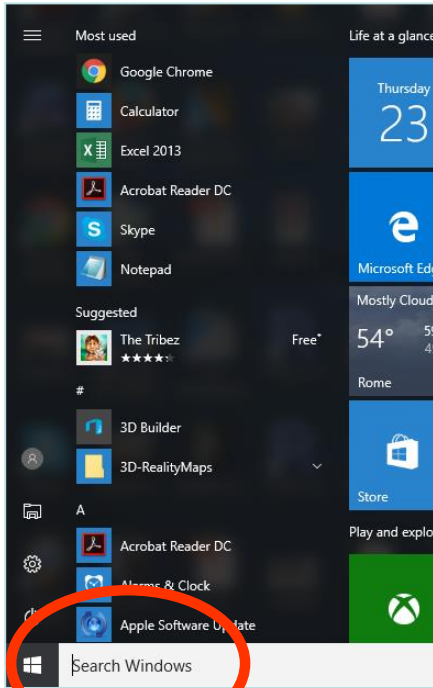
An *operating system shell* is a software that provides an interface between users and the OS.

Basic features:

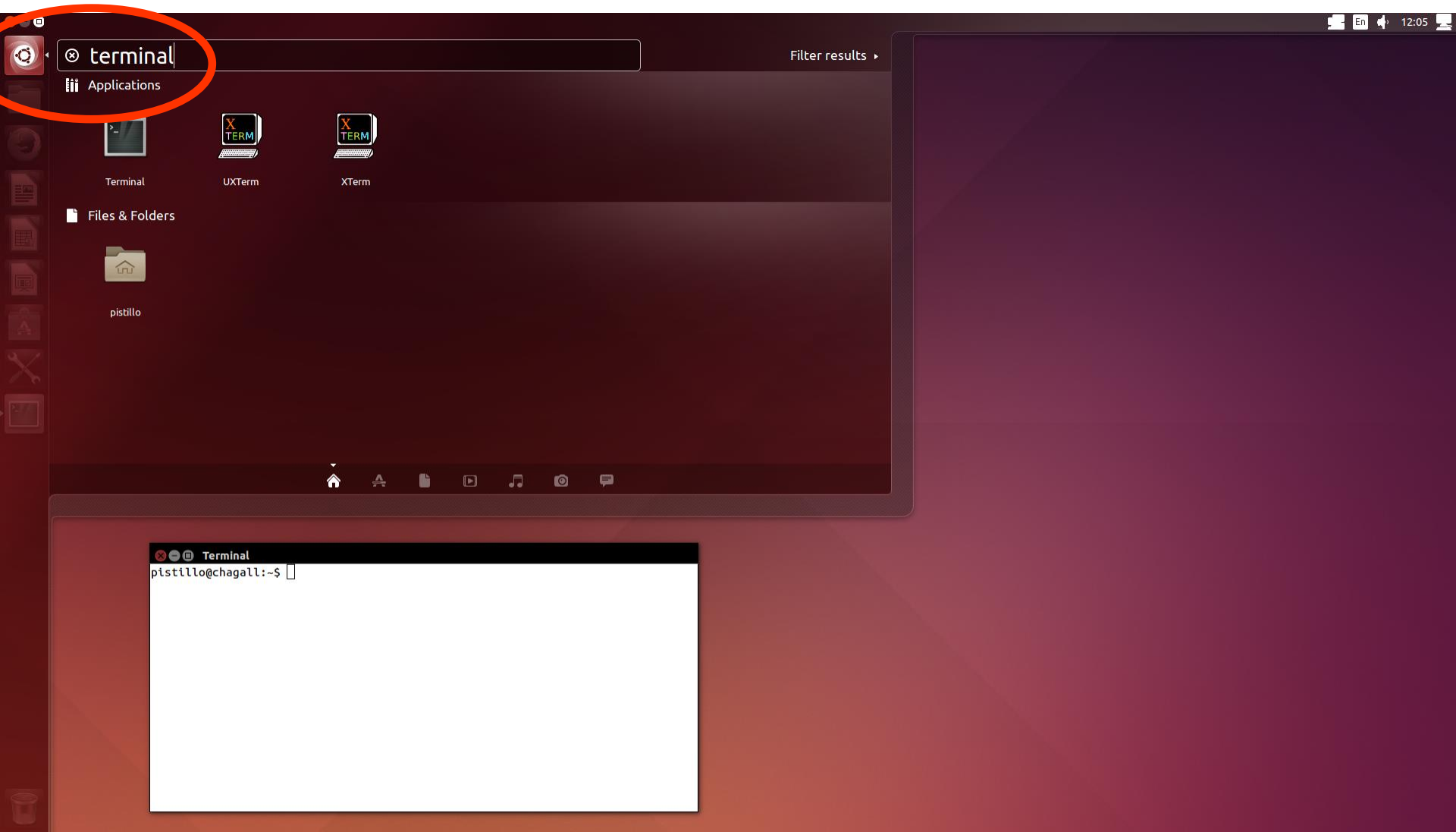
- to call (or "launch") another program
- viewing the contents of directories
- copying/moving files

It can work as *command line interface* (CLI) or *graphical user interface* (GUI)

CLI for Windows



CLI for Linux (Ubuntu)



text editor

It is a program for text file editing. They are usually provided with the OS.

Windows: notepad, wordpad, word, notepad++

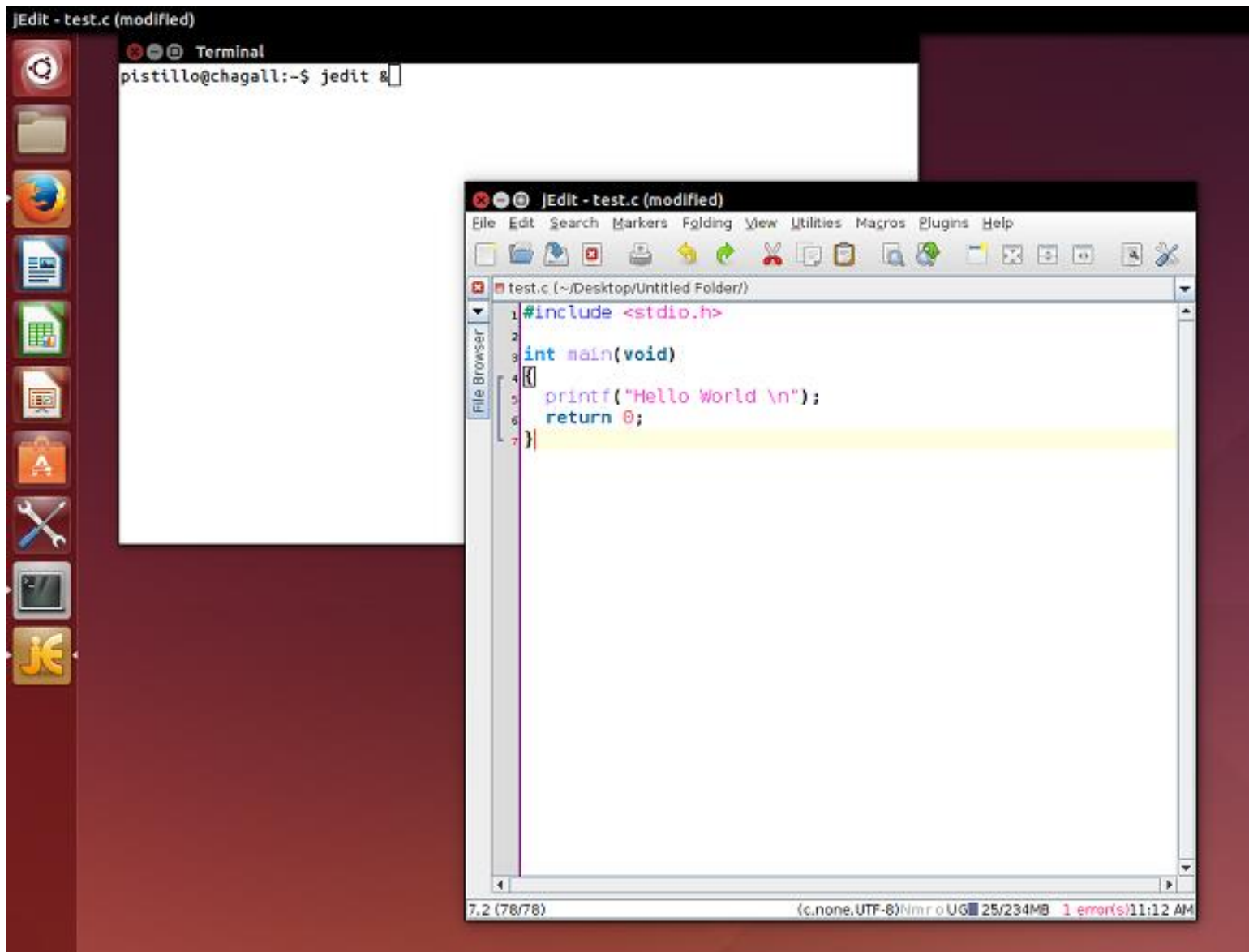
Linux: vi, emacs, gedit

Mac: Xcode, TextWrangler

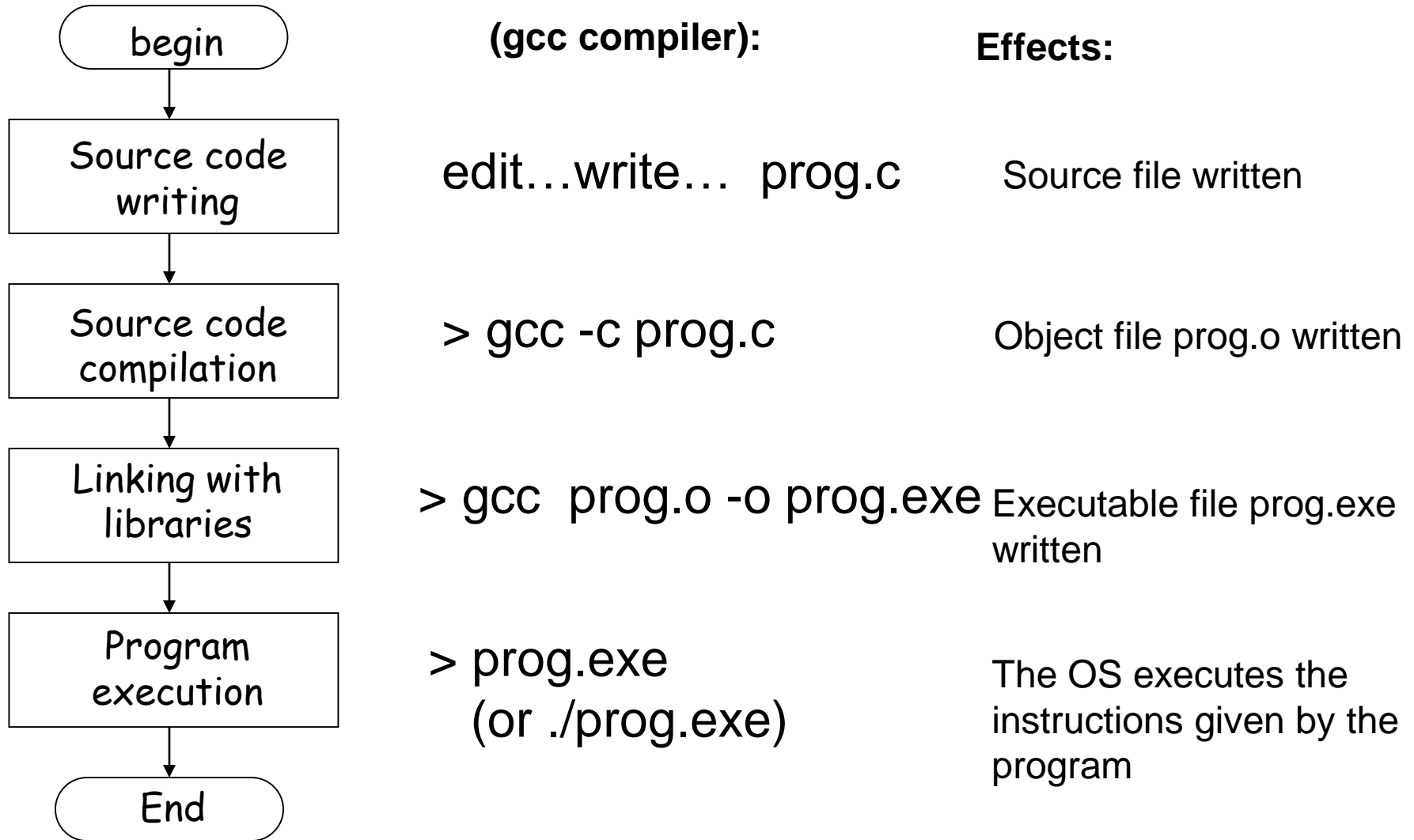
Some of them are designed for writing the program language source code. Typical features:

- search and replace
- copy, cut and paste
- text formatting (indentation)
- undo and redo
- syntax checks
- ...

jedit: an editor for Linux

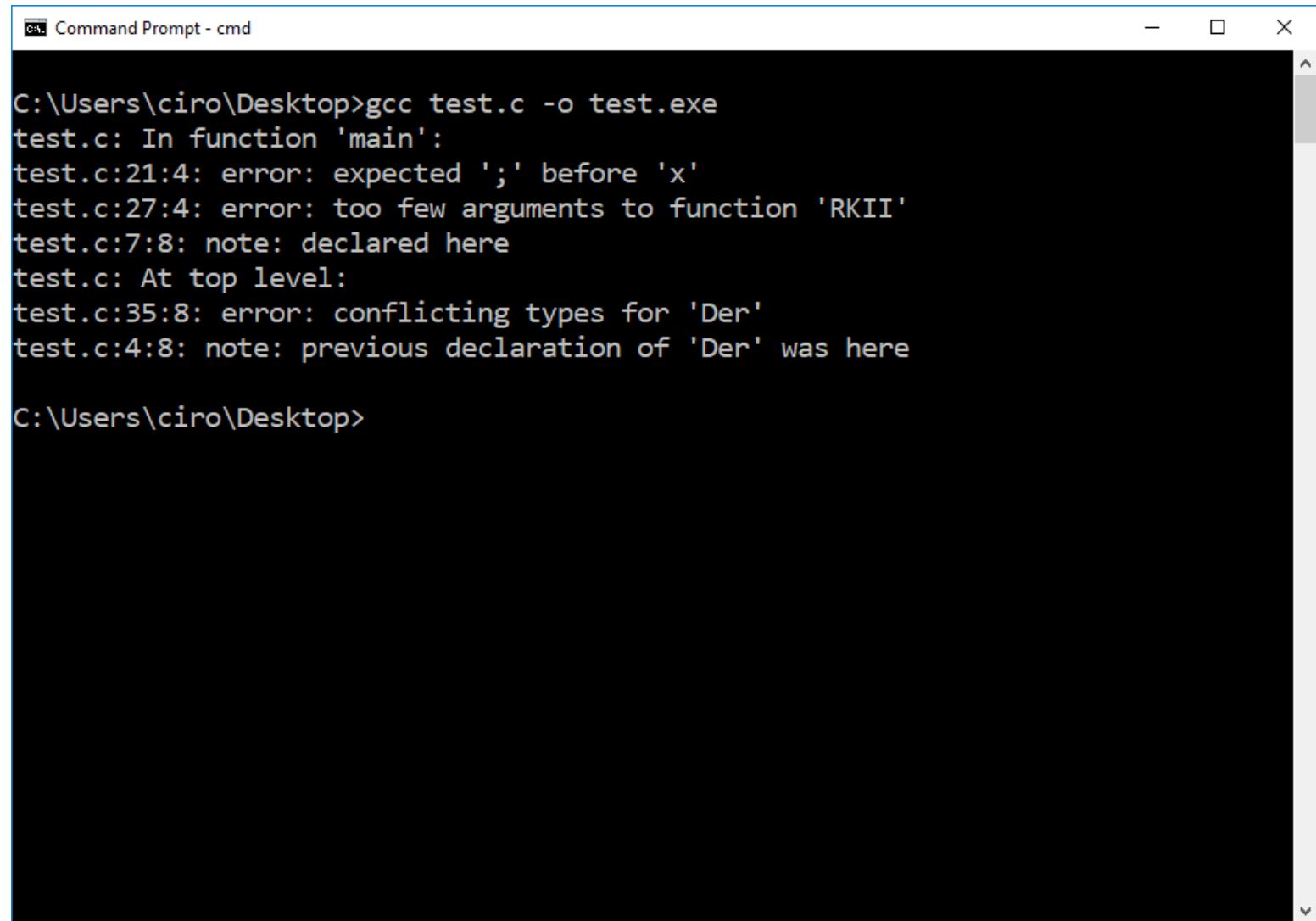


To make a program with C



You can also directly type: > gcc prog.c -o prog.exe (compilation+linking) 19

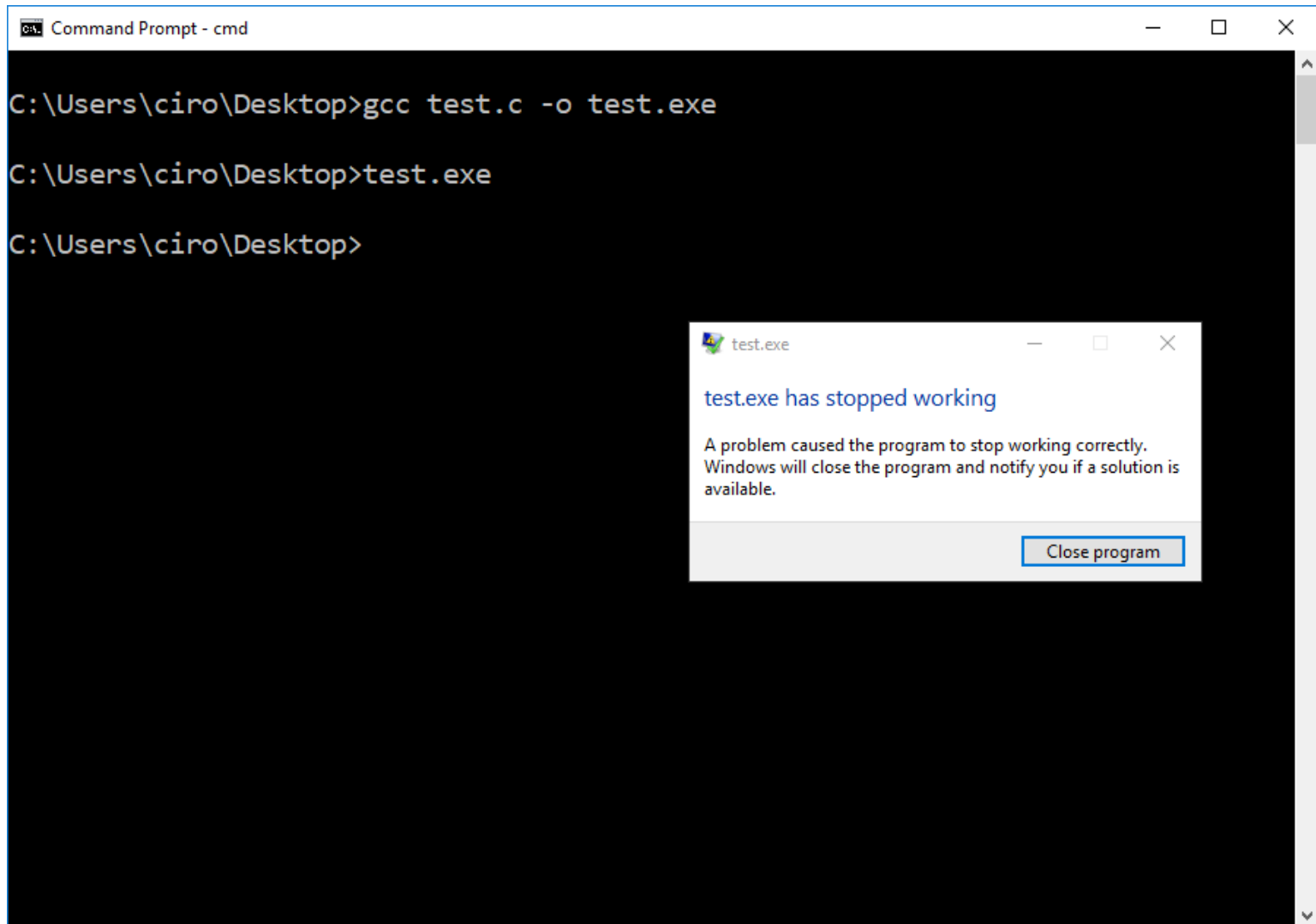
The compiler will help you sometimes...



```
Command Prompt - cmd
C:\Users\ciro\Desktop>gcc test.c -o test.exe
test.c: In function 'main':
test.c:21:4: error: expected ';' before 'x'
test.c:27:4: error: too few arguments to function 'RKII'
test.c:7:8: note: declared here
test.c: At top level:
test.c:35:8: error: conflicting types for 'Der'
test.c:4:8: note: previous declaration of 'Der' was here

C:\Users\ciro\Desktop>
```

...but not always



The first C program



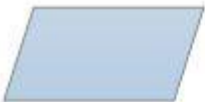
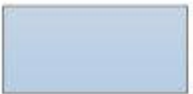

```
/* This is my first C program,  
   just to say hallo to the world */  
// I include the I/O library  
  
#include <stdio.h>  
  
int main() //this is the main function  
{  
    printf("Hallo, World");  
    printf("\n");  
  
    return 0;  
}
```

Comments: the compiler ignore what is between /* and */ or what follows // (till the end of the row)

All the instructions end with a semicolon

start/end of a block (here, the main function)

Flowchart symbols

Symbol	Name	Function
	Start/end	An oval represents a start or end point
	Arrows	A line is a connector that shows relationships between the representative shapes
	Input/Output	A parallelogram represents input or output
	Process	A rectangle represents a process
	Decision	A diamond indicates a decision