Lesson 2 - Introduction to C language

Logical Computational Thinking

Stefano MARTINA

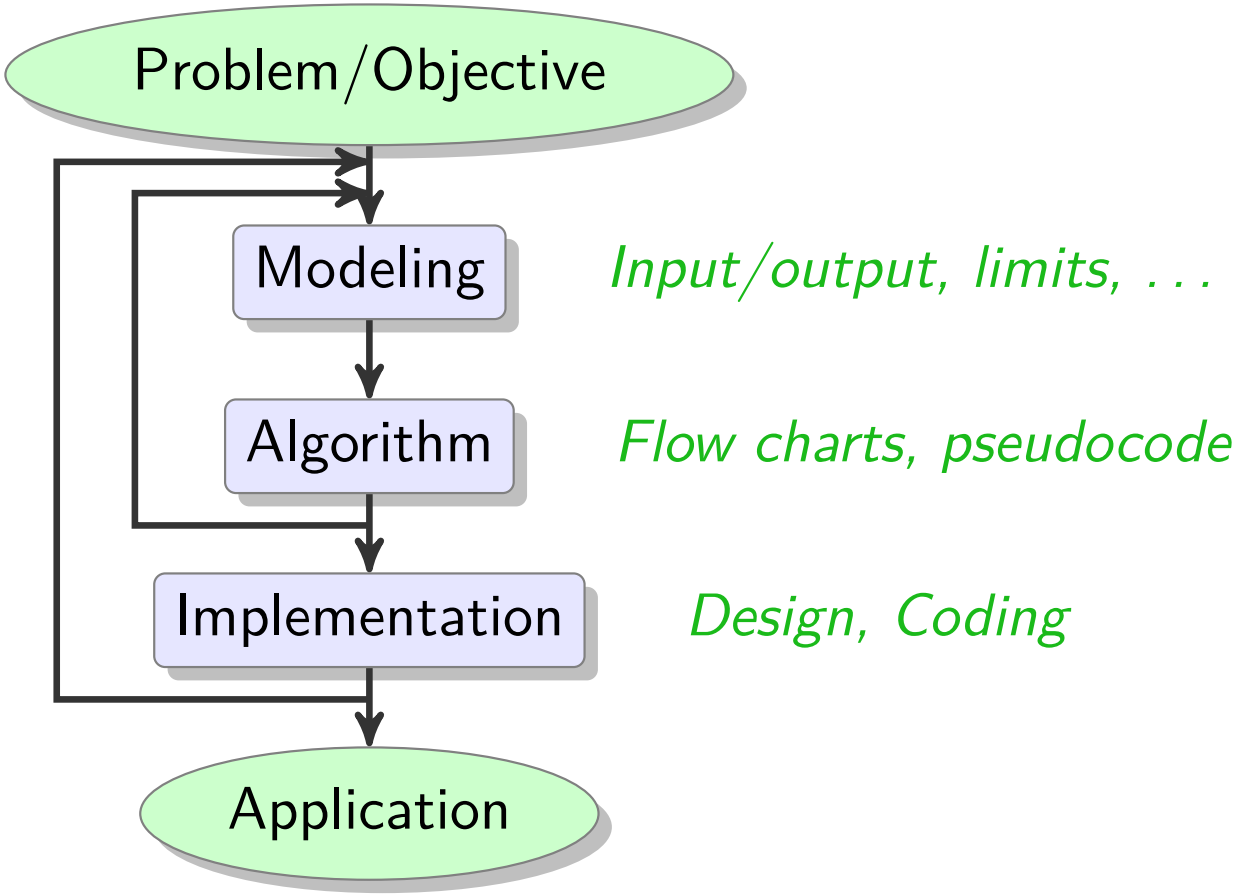
stefano.martina@gmail.com

Tecnológico de Monterrey

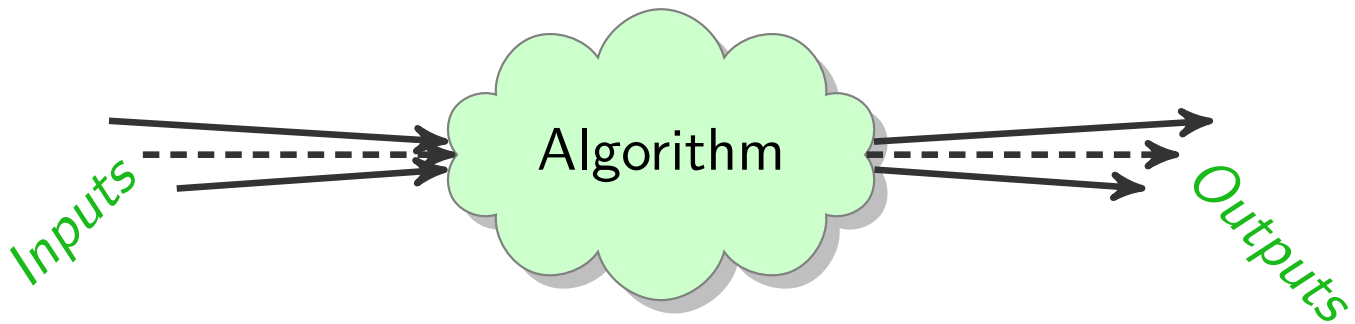

Scuola Leonardo Da Vinci (Firenze)

10 September 2015

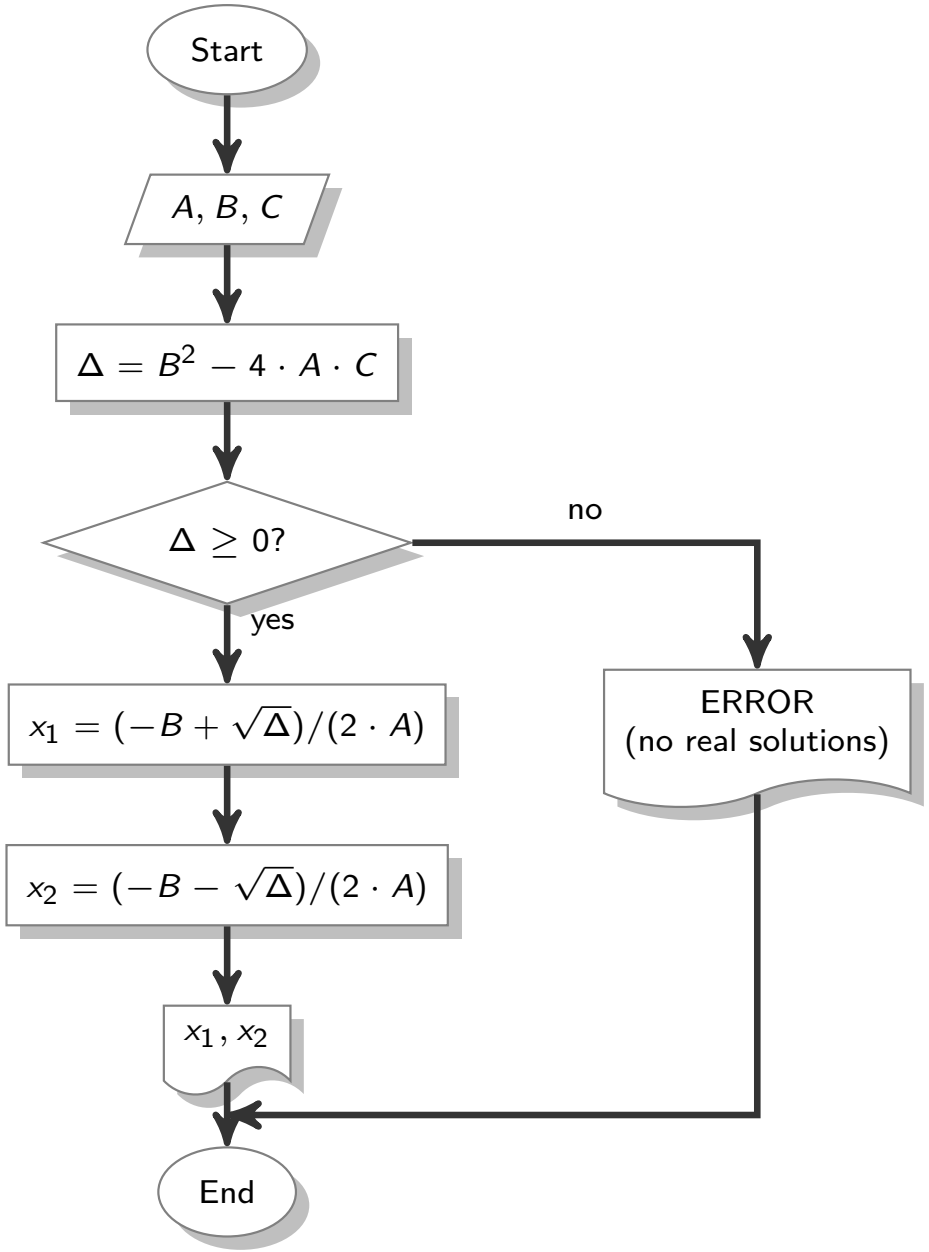
Review


Problem/Objective
Modeling
Input/output, limits, . . .
Algorithm
Flow charts, pseudocode
Implementation
Design, Coding
Application

Model



Algorithm


Start
A, B, C
$\Delta = B^2 - 4 \cdot A \cdot C$
$\Delta \geq 0$?
no
yes
$x_1 = (-B + \sqrt{\Delta})/(2 \cdot A)$
$x_2 = (-B - \sqrt{\Delta})/(2 \cdot A)$
x_1, x_2
ERROR
(no real solutions)
End

C language

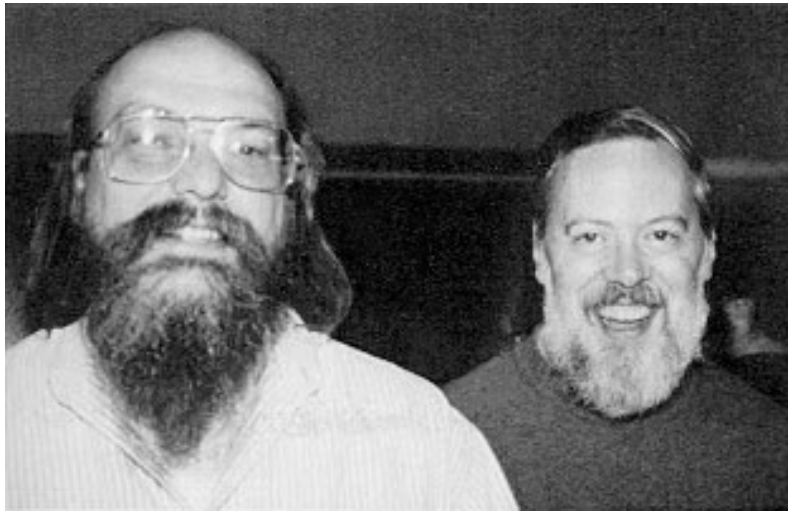


Figure: Ken Thompson and Dennis Ritchie

History

- ✓ The language C is developed in the 70's by Dennis Ritchie
- ✓ Along with the Unix system, created by Ken Thompson and Dennis Ritchie in the same years

Basics

Blocks

```
{
    ...
}
```

- ✓ Good practice to indent the code to the right every time a block is open, and to the left when is closed.

Comments

```
//Single line comment

/* Multiple
lines
comments */
```

Main and libraries

Libraries

```
1 #include<stdio.h>
2 #include<math.h>
```

- ✓ For including code defined elsewhere.
- ✓ Can be custom libraries or standard libraries like:
  - `stdio.h` stands for *STandard Input Output* and provide basic interface with the terminal;
  - `math.h` provide some useful mathematical functions, like `pow` and `sqrt` .
- ✓ The `#include` directives must be write on top of the source file (before the main)

Main

✓ Is the entry point for the program.

```
1  int main(void) {
2     ...
3     return 0;
4  }
```

or

```
1  int main(int argc, char *argv[]) {
2     ...
3     return 0;
4  }
```

Example

hello_world.c

```c
//Compile it with gcc hello_world.c -o
    ↪ hello_world

#include<stdio.h>  //library for input/output

int main(void) {  //begin of main
   printf("Hello␣world!\n");  //output of
       ↪ string
   return 0;
}
```

Variables

Initialization

```
1 int var1;                    //default initial value
2 float var2 = 3.1415; //custom initial value
3 int var3, var4, var5;//multiple initialization
```

- ✓ C is case sensitive, `int` ≠ `Int` ≠ `INT` .
- ✓ Allowed names can contains `[A-Z,a-z,0-9,_]` , cannot begin with a number.
- ✓ Good practices are: to use camel case or `_` for composed words, and to start with lower case. I.e. `camelCaseExample` .
- ✓ Also when possible define variables on method begin.

Variables

Assignation

```
1  var1 = 42;
```

- ✓ Is possible to use also expressions on the right of $=$
- ✓ Also with other variables or the same variable.

```
1  var3 = var4 + var5;
2  var3 = var3 - 1;
```

Variable types

Integer

type	size	min value	max value
`char`	1 byte	-128	127
`short`	2 bytes	-32,768	32,767
`int`	4 bytes	-2,147,483,648	2,147,483,647
`long`	8 bytes	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
`unsigned char`	1 byte	0	255
`unsigned short`	2 bytes	0	65,535
`unsigned int`	4 bytes	0	4,294,967,295
`unsigned long`	8 bytes	0	18,446,744,073,709,551,615

✓ `char` is used also for the characters of the ascii table.

Variable types

Floating point

type	size	min value	max value	epsilon
`float`	4 bytes	$1.175494e^{-38}$	$3.402823e^{38}$	$1.192093e^{-07}$
`double`	8 bytes	$2.225074e^{-308}$	$1.797693e^{308}$	$2.220446e^{-16}$
`long double`	16 bytes	$3.362103e^{-4932}$	$1.189731e^{4932}$	$1.084202e^{-19}$

Boolean

- ✓ Is possible to use the type `bool`, and the values `true` and `false`.
- ✓ You need to add `#include<stdbool.h>`.
- ✓ Not really useful because you can use any integer type with the values:
  - 0 for false;
  - $\neq 0$ for true.

Operators

Arithmetic

In priority order:

- ✓ `*` , `/` , `%` ;
- ✓ `+` , `-` ;

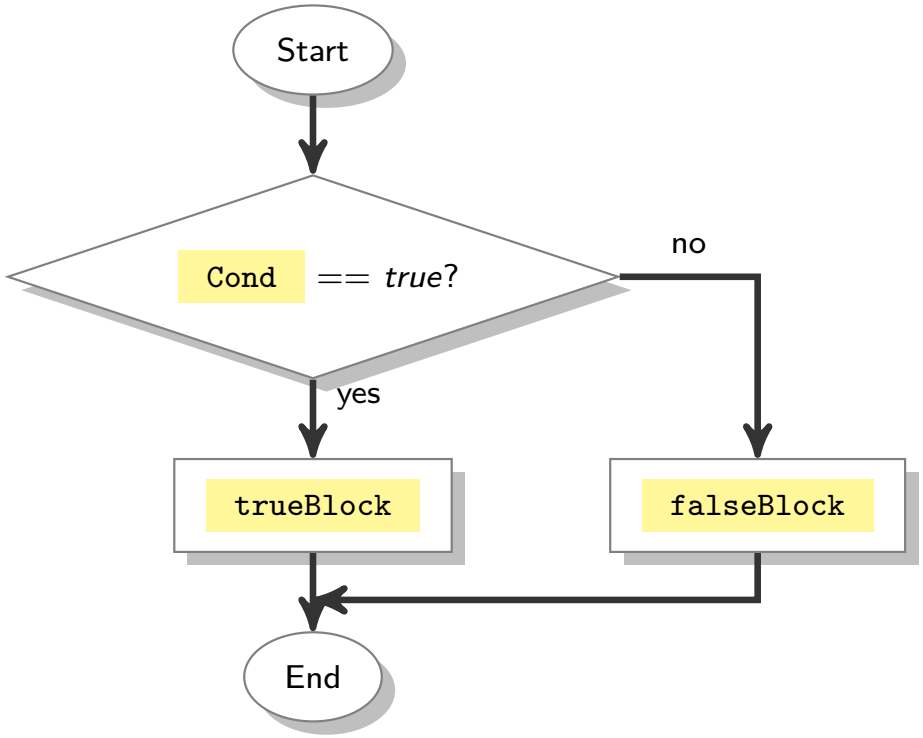
Logic

- ✓ `&&` , `||` ;
- ✓ `!` ;
- ✓ `==` , `!=` , `<` , `>` , `<=` , `>=`

- ✓ Is possible to make groups with `(` and `)` , i.e.:

```
1  (a+b)*c
```

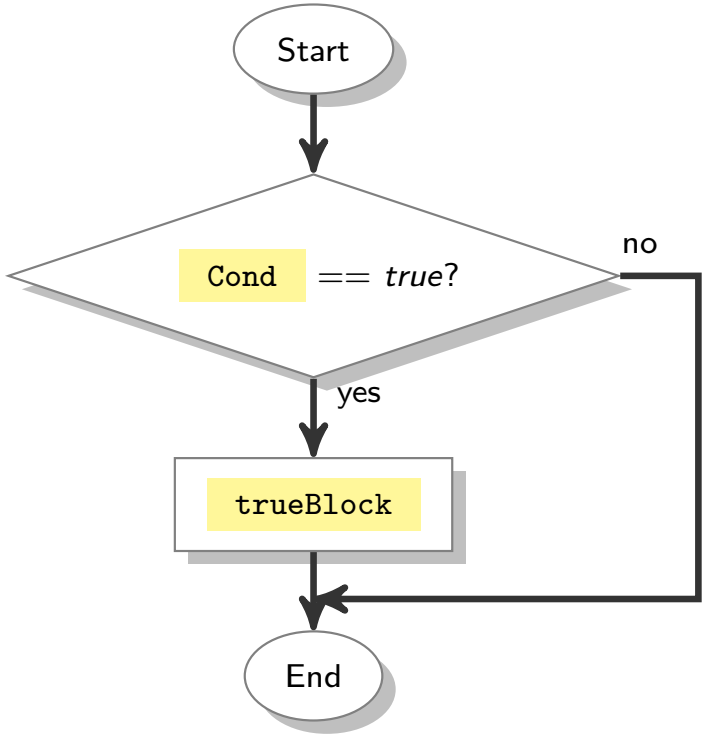
Selections




```
1  if(cond) {
2     trueBlock;
3  } else {
4     falseBlock;
5  }
```

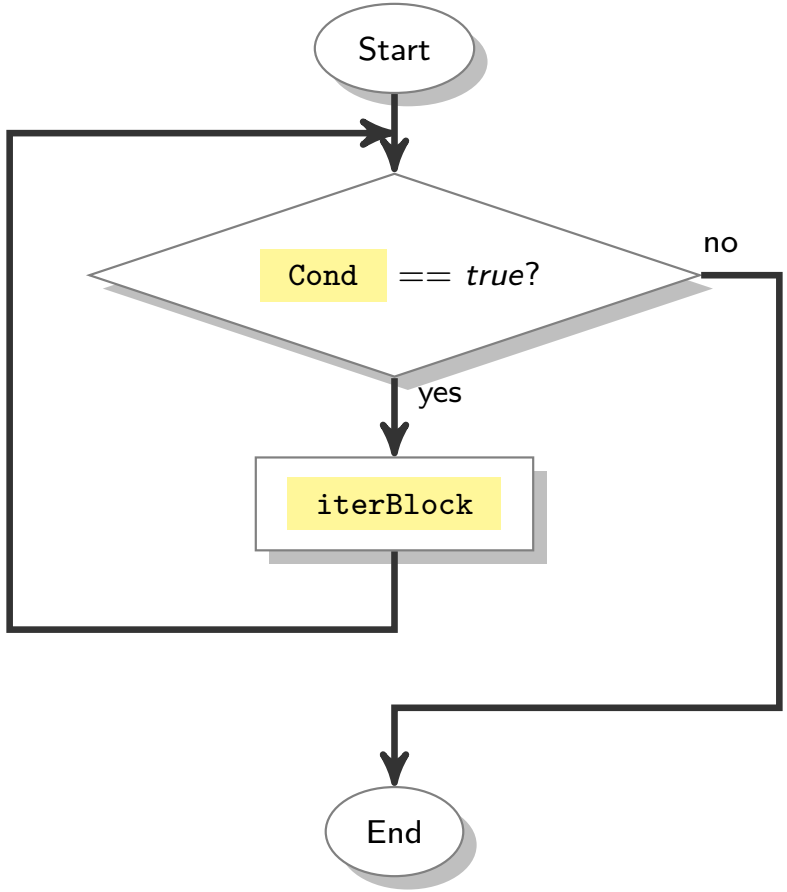
Selections

The `else` block is optional.



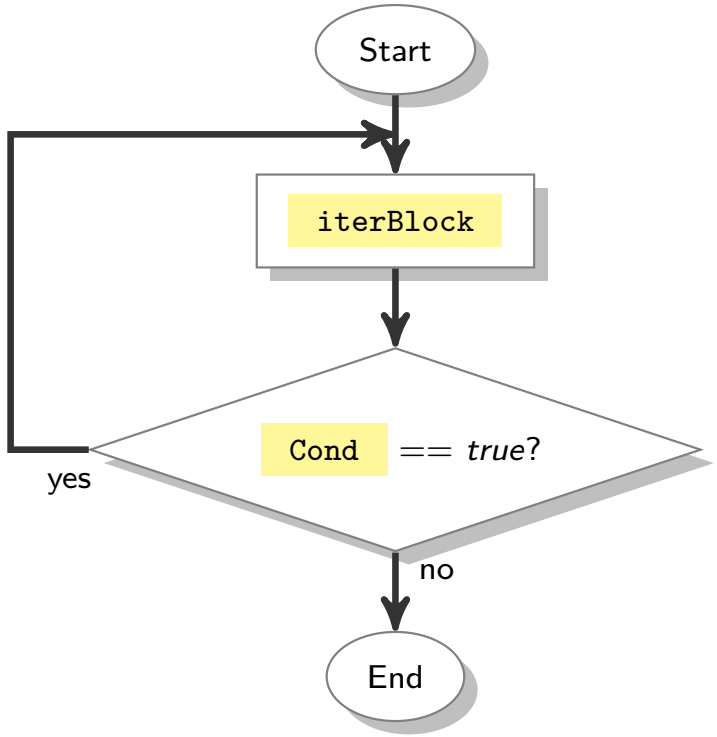

```
1  if(cond) {
2    trueBlock;
3  }
```

While iteration




```
1  while(cond) {
2     iterBlock;
3  }
```

Do-while iteration




```
1  do {
2     iterBlock;
3  } while(cond);
```

For iteration

```
for(iniz; cond; oper) {
   iterBlock;
}
```

is equivalent to:

```
iniz;
while(cond) {
   iterBlock;
   oper;
}
```

For iteration example

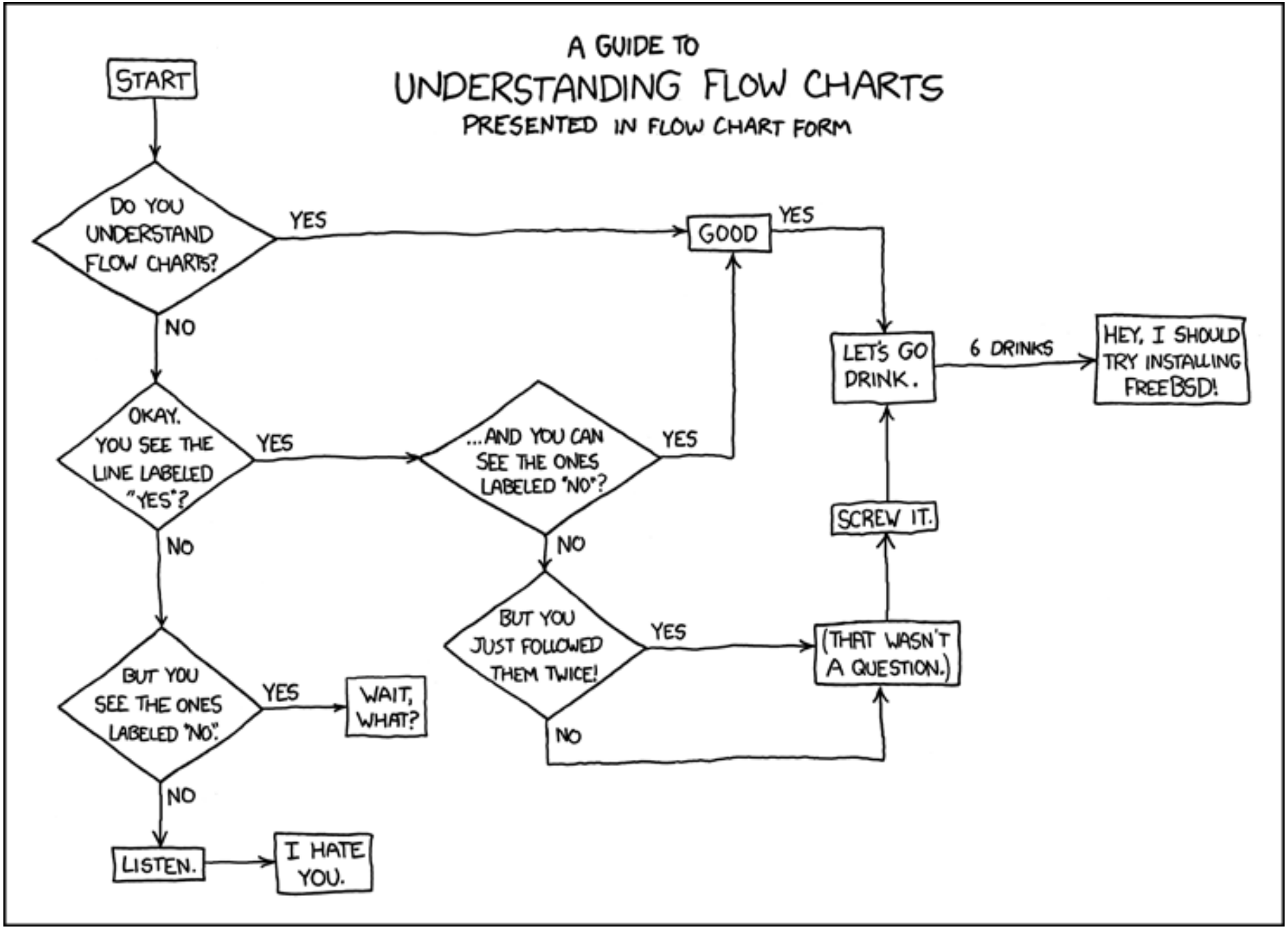
forTest.c

```c
#include<stdio.h>

int main(void) {
   int i; //initialize var

   //iterate from i=1 while i<=42
   //incrementing i on each loop
   for(i = 1; i <= 42; i++) {
      printf("*"); //print an *
   }
   printf("\n");

   return 0;
}
```


A GUIDE TO
UNDERSTANDING FLOW CHARTS
PRESENTED IN FLOW CHART FORM
START
DO YOU UNDERSTAND FLOW CHARTS?
YES
GOOD
YES
NO
LET'S GO DRINK.
6 DRINKS
HEY, I SHOULD TRY INSTALLING FREEBSD!
OKAY. YOU SEE THE LINE LABELED "YES"?
YES
...AND YOU CAN SEE THE ONES LABELED "NO"?
YES
SCREW IT.
NO
NO
BUT YOU JUST FOLLOWED THEM TWICE!
YES
(THAT WASN'T A QUESTION.)
BUT YOU SEE THE ONES LABELED "NO".
YES
WAIT, WHAT?
NO
NO
LISTEN.
I HATE YOU.