CS - 114 : Computer Workshop

Prof. Chamakuri Nagaiah
Mahindra-École Centrale, Hyderabad
nagaiah.chamakuri@mechyd.ac.in

Course information

- Weekly: 1 main lecture, 1 lab Session (Batch 1: A_1 ; Batch 2: A_2)
- Lab sessions: Dr. Jai Prakash

Evaluation in the course:

- Mid semester 1 –
- Final Exam –
- Assignments
- Lab session and attendance –

Syllabus

- Types, Operators and Expressions : Variable Names, Data Types and Sizes, Constants, Declarations, Arithmetic Operators, Relational and Logical Operators, Type Conversions, Increment and Decrement Operators, Bit-wise Operators.
- Control Flow : Statements and Blocks, if-else, loops, break and continue.
- Functions and Program Structure : Functions Returning Non-integers, local Vs Global variables, Scope Rules, Header Files, Static and register variables, Recursion, ...
- Pointers and Arrays : Pointers and Addresses, Pointer Arrays; Pointers to Pointers, Multi-dimensional Arrays, Pointers to Functions
- Structures : Structures and Functions, Arrays of Structures, Pointers to Structures, Unions, ...
- Input and Output

References

1. The C Programming Language: Brian W Kernighan, Dennis M Ritchie, Prentice Hall India
2. Programming with C (Second Edition) : Byron Gottfried, Third Edition, Schaum's Outlines Series, McGraw–Hill, 2011
3. Programming in ANSI C: Balagurusamy
4. Many other books are available and may serve the same purpose, but the BIGGEST library is "internet library"

How does a computer work

- Stored program concept.
  – Main difference from a calculator.
- What is a program?

How does a computer work

- Stored program concept.
  – Main difference from a calculator.
- What is a program?
  – Set of instructions for carrying out a specific task.
- Where are programs stored?
  – In secondary memory, when first created.
  –Brought into main memory, during execution.

Low- and High-Level Languages

- Machine language and assembly language are called low-level languages.
  – They are closer to the machine.
  – Difficult to use.
- High-level languages are easier to use.
  – They are closer to the programmer.
  –Examples: FORTRAN, COBOL, C, C++, Java.
  – Requires an elaborate process of translation: Using a software called compiler
  – They are portable across platforms.

What is C? Why is it special?

- C is small (only 32 keywords).
- C is common (lots of C code about).
- C is stable (the language doesnâĂŹt change much).
- C is quick running.
- C is the basis for many other languages (Java, C++, Perl, ...).
- It may not feel like it but C is one of the easiest languages to learn.

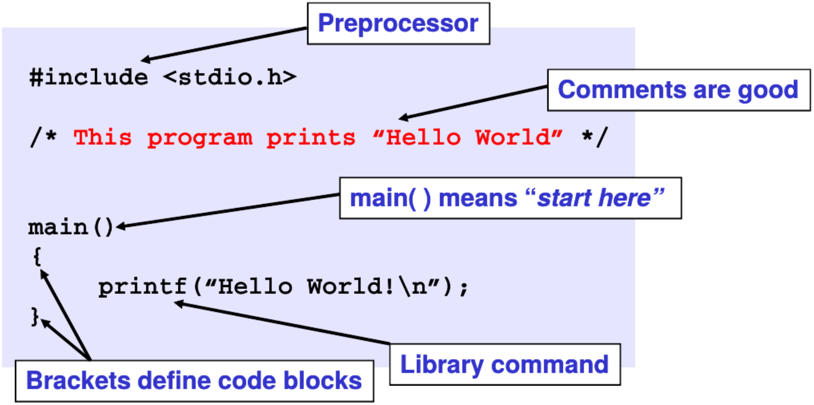
Some programmer jargon

- Source code: The stuff you type into the computer. The program you are writing.
- Compile (build): Taking source code and making a program that the computer can understand.
- Executable: The compiled program that the computer can run.
- Language: The core part of C central to writing C code.
- Library: Added functions for C programming which are bolted on to do certain tasks.
- Header file: Files ending in .h which are included at the start of source code.

Some Terminologies

- Algorithm / Flowchart
  – A step-by-step procedure for solving a particular problem.
  – Independent of the programming language.
- Program
  – A translation of the algorithm/flowchart into a form that can be processed by a computer.
  – Typically written in a high-level language like C, C++, Java, etc.
- Most important concept for problem solving based on using computers
- All temporary results are stored in terms of variables
  – The value of a variable can be changed.
  – The value of a constant do not change.
- Where are they stored?
  – In main memory.

Our First C Program: Hello World



```
#include <stdio.h>

/* This program prints "Hello World" */

main()
{
    printf("Hello World!\n");
}
```

About spaces ...

```
#include <stdio.h> /* This program prints "Hello World" */
int main( ) {printf("Hello World!\n");}
```

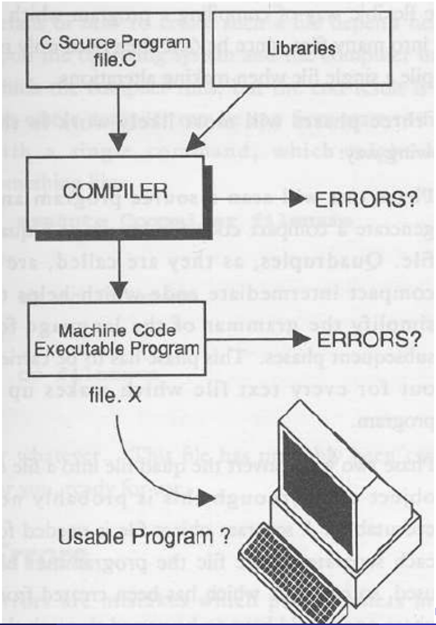
```
#include <stdio.h>
/* This program
prints "Hello
World"
*/
int
main( )
{
printf("Hello
World!
\n")
;
}
```

Both of these programs are exactly the same as the original as far as your compiler is concerned.

The Compiler

- Phase 1 scans a source program, perhaps generating an intermediate code which helps to simplify the grammar of the language for subsequent processing. It then converts the intermediate code into a file of object code (though this is usually not executable yet). A separate object file is built for each separate source file. In the GNU C compiler, these two stages are run with the command gcc -c and the output is one or more .o files.
- Phase 2 is a Linker. This program appends standard library code to the object file so that the code is complete and can "stand alone". A C compiler linker suffers the slightly arduous task of linking together all the functions in the C program. Even at this stage, the compiler can fail, if it finds that it has a reference to a function which does not exist. With the GNU C compiler this stage is activated by the command gcc -o or ld.
- Errors : Syntax and logical errors???

The Compiler



Keywords of C

- Flow control (6) : if, else, return, switch, case, default
- Loops (5) : for, do, while, break, continue
- Common types (5) : int, float, double, char, void
- Structures (3) : struct, typedef, union
- Counting and sizing things (2) : enum, sizeof
- Rare but still useful types (7) : extern, signed, unsigned, long, short, static, const
- Evil keywords which we avoid (1) : goto
- Wierdies (3) : auto, register, volatile

Data Types

Three common data types used:

- Integer : can store only whole numbers
  – Examples: 25, -56, 1, 0
  – 16 bits or 32 bits (Actual number of bits vary from one computer to another)
- Floating point : can store numbers with fractional values.
  – Examples: 3.14159, 5.0, -12345.345
  – 32 bits or 64 bits
- Character : can store a character
  – Examples: ’A’, ’a’, ’*’, ’3’, ’ ’, ’+’
  – 8 bits (ASCII code) or 16 bits (UNICODE, used in Java)

- In addition to +, -, * and / we can also use +=, -=, *=, /=, – and % (modulo)

`n++`	*increment n*	
`n--`	*decrement n*	
`a+=5`	*is equivalent to*	`a = a+5;`
`a-=5`	*is equivalent to*	`a = a-5;`
`a*=5`	*is equivalent to*	`a = a*5;`
`a/=5`	*is equivalent to*	`a = a/5;`

`(x % y)` gives the remainder when `x` is divided by `y`

Problem solving

- Step 1:

  –Clearly specify the problem to be solved.
- Step 2:

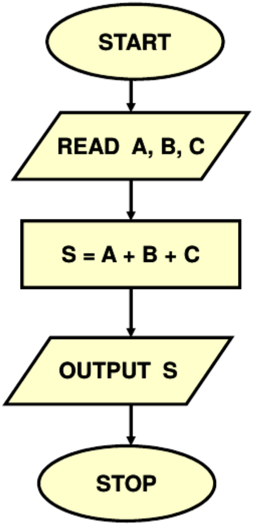
  – Draw flowchart or write algorithm.
- Step 3:

  – Convert flowchart (algorithm) into program code.
- Step 4:

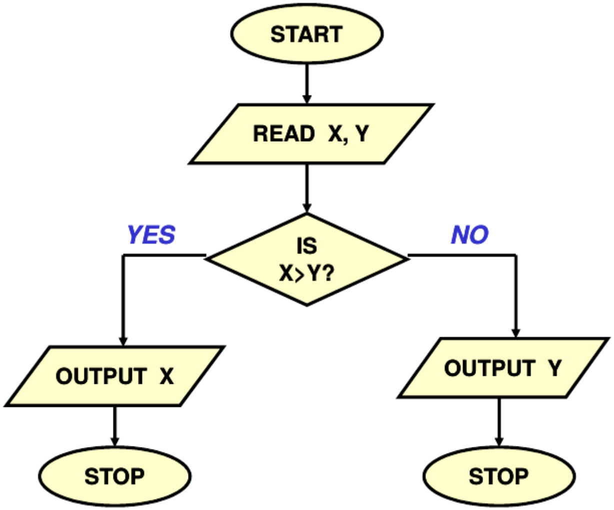
  –Compile the program into object code.
- Step 5:

  – Execute the program.

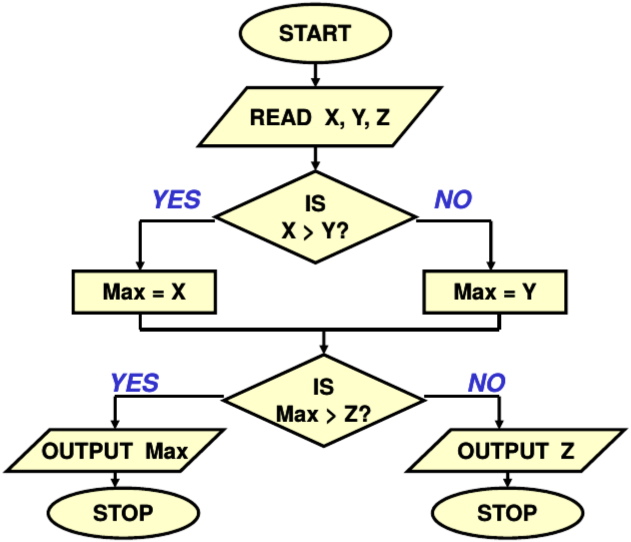
Example 1: Adding three numbers



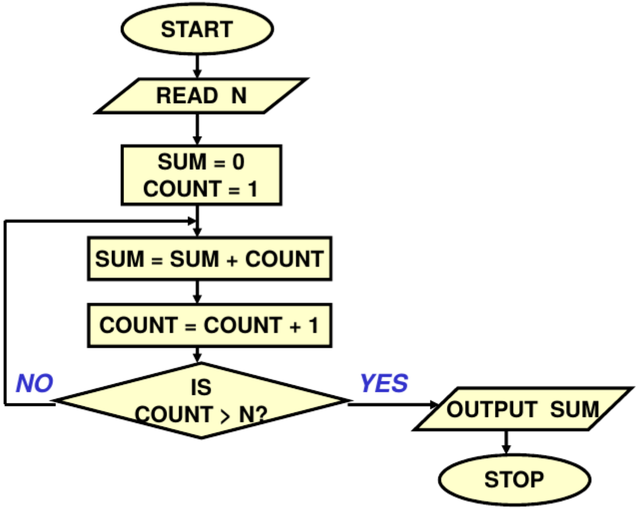
Example 2: Larger of two numbers



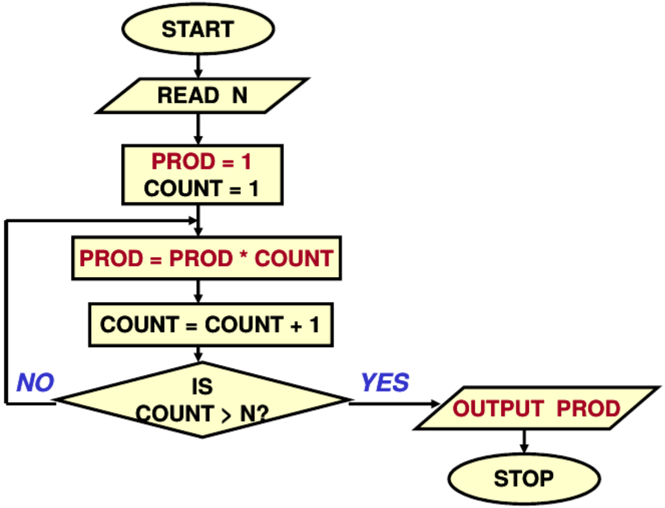
Example 3: Largest of three numbers



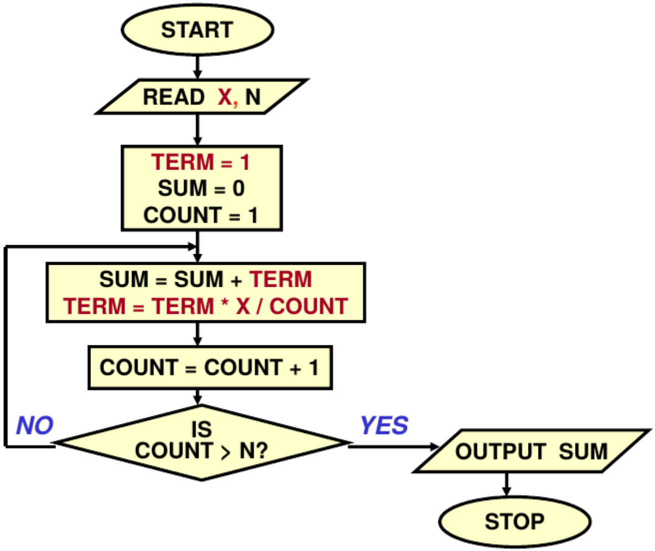
Example 4: Sum of first N natural numbers



Example 5: Computing Factorial



Example 6: Computing e^x series up to N terms


START
READ X, N
TERM = 1
SUM = 0
COUNT = 1
SUM = SUM + TERM
TERM = TERM * X / COUNT
COUNT = COUNT + 1
IS COUNT > N?
NO
YES
OUTPUT SUM
STOP