

Introduction to Programming

Inst. Nguyễn Minh Huy

Contents



- Computer programming.
- Computer algorithm.

Contents



- **Computer programming.**
- Computer algorithm.

Computer programming



■ Programming concepts:

■ Problem:

➤ Teach a kid to find sum of max and min among 7, 1, 9.

➤ Rules:

➤ The kid knows addition and comparison between two numbers.

➤ Write solution in step-by-step.

■ Programming ~ teaching:

➤ From what computer knows.

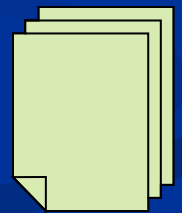
➤ Step-by-step.



Computer



Programming



Program

Computer programming



■ Programming concepts:

- Computer has a set of hard-wired instructions.
- Programming: write instructions for computer.
- Computer program:
 - Written organized instructions.
 - Follow step-by-step.
 - To solve problem.

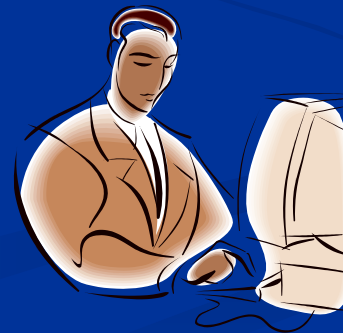


Computer programming



■ Programmer:

- Person who writes the program.
- Programmer, coder, developer.
- To become a good programmer:
 - Logical thinking.
 - Analytical and careful.
 - Hands-on and explore.
- The first programmer?



Ada Lovelace

Computer programming



■ Instruction code:

■ Machine code:

- Binary sequence of '0' and '1'.
- Computer can execute directly.

■ Pseudo-code:

- Natural language.
- Human can understand.

■ Source code:

- Intermediate language.
- Easy to read, not too vague.
- Can be translated to machine code.
 - Compiler vs. Interpreter.
 - Who wrote the first compiler?



Step 1: add a, b.
Step 2: compare a, c.
Step 3: output a.

...

```
#include <stdio.h>

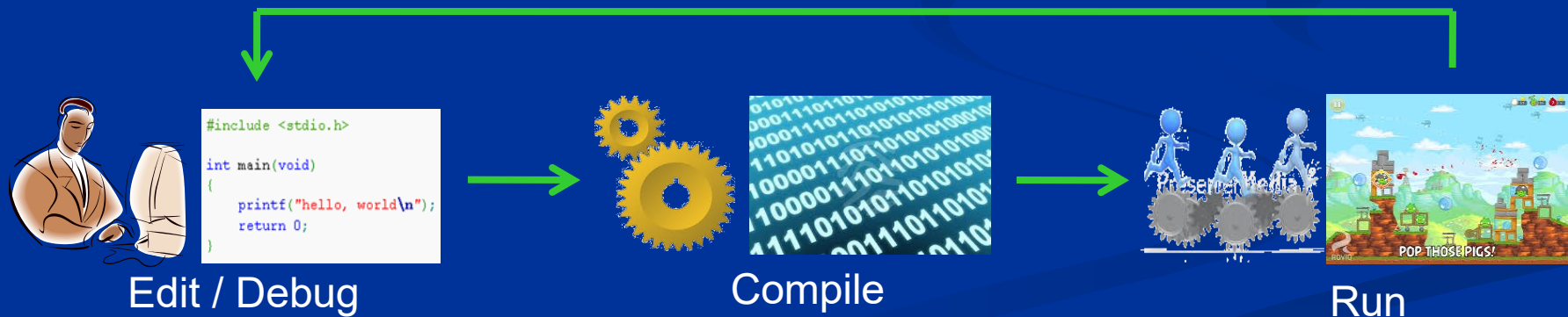
int main(void)
{
    printf("hello, world\n");
    return 0;
}
```

Computer programming



■ Programming cycle:

- Edit: write source code.
 - Compile: source code → machine code.
 - Run: execute machine code to perform tasks.
 - Debug: review source code to fix errors.
- Debug takes the most time!



Computer programming



■ Programming language:

- Intermediate language between human and computer.
- Concise and precise.
- There are thousands of programming languages!
- Learning programming thinking is more important than learning programming language.



Computer programming



■ Programming language:

■ Low-level language:

- 1st generation: machine code.
- 2nd generation: assembly language.

■ High-level language:

- Procedural: Fortran, Pascal, C.
- Functional: Lisp, Haskell, F#.
- Object oriented: C++, Java, C#.

■ Trends:

- Scripting languages: JavaScript, Python.
- AI prompts: ask AI to generate source code.





■ Programming environment:

- A platform of tools supporting programming tasks.
- Source code editor:
 - Edit text, syntax highlight, code completion.
- Compiler:
 - Translate source code to machine code.
- Terminal:
 - Command-line tool to run program and shell commands.
- Debugger:
 - Run program step-by-step.
 - Monitor computer memory.



■ Programming environment:

■ IDE – Integrated Development Environment:

- A single application combining programming tools all-in-one.
- User friendly, configuration-free, add-in features.
- Cons:
 - Heavy memory and storage usage.
 - Limit configuration skill.
 - Require license.
 - Track user data.



Computer programming



■ C/C++ programming environment:

Tools	Windows	MacOS	Linux
Editor	Notepad++ VS Code (*) Sublime Text (*)	TextMate VS Code (*) Sublime Text (*)	Vim, Emacs VS Code (*) Sublime Text (*)
Compiler	MSVC (*) GCC (+) Clang (+)	Clang GCC	GCC Clang
Debugger	MSVC (*) GDB	LLDB GDB	GDB LLDB
IDE	Visual Studio (*) Code::Blocks CodeLite	Xcode (*) Code::Blocks CodeLite	Code::Blocks CodeLite

(*): Proprietary software and track user data.

(+): Run on Linux simulated environment like MinGW64, WSL.



■ C/C++ programming environment:

■ Set up loosely-coupled environment:

- Windows: MinGW64, GCC, GDB.
 - Project WinLibs: light-weight, not extensible, <https://winlibs.com>
 - Project MSYS2: get MSYS2 and toolchain, <https://www.msys2.org>
- Linux (Ubuntu/Debian): GCC, GDB.
 - Install package **build-essential**.
- MacOS: Clang, LLDB.
 - Terminal command: **xcode-select --install**.

■ Set up IDE:

- Visual Studio (Windows): <https://visualstudio.microsoft.com>
- Xcode (MacOS): **xcode-select --install**, then **Get Xcode**.
- Code::Blocks: <https://www.codeblocks.org>
- CodeLite: <https://codelite.org>

Contents

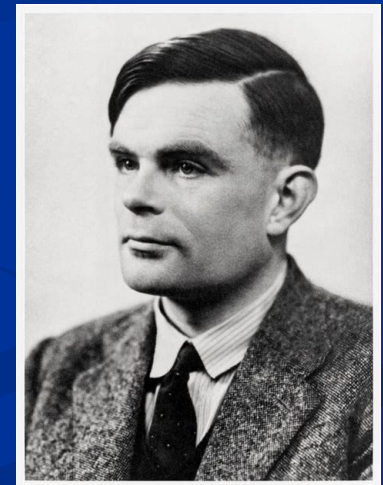


- Computer programming.
- **Computer algorithm.**



■ Algorithm concepts:

- Different machines have different instruction code.
- Turing machine: abstract machine and instructions.
- Algorithm ~ program on Turing machine:
 - Finite steps to solve problem on Turing machine.
 - Does not depend on specific machine.
 - Is more abstract than program.

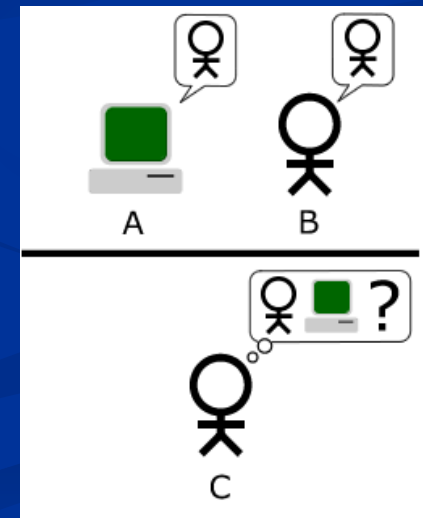


Alan Turing

Computer algorithm



- Algorithm basic instructions:
 - Read input, write output.
 - Arithmetic operations: $+$, $-$, $*$, $/$, $=$.
 - Comparisons: $>$, $<$, $>=$, $<=$, $==$, $!=$.
 - Condition: if-else.
 - Loop: while.
- Can machines think?
 - The Imitation game - Turing test.



Computer algorithm



■ Algorithm representations:

■ Pseudo-code:

- Use natural language.
- Notations:

Instructions	Notations	Example
Read input Write output	Input Output	-Step 1: Input a, b, c. -Step 2: Output a + b + c.
Arithmetic operations	+, -, *, /, %, =	-Step 1: a = 5. -Step 2: b = (a + 5) / (a - 3).
Condition	If <condition> ... Else ...	-Step 1: Input a. -Step 2: If a < 0 Output "Negative" Else Output "Positive".
Loop	While <condition> ...	-Step 1: a = 0. -Step 2: While a < 10 Output a. a = a + 1.



■ Algorithm representations:

■ Pseudo-code:

* Find max among 3 numbers:

-Step 1: **Input** a, b, c.

-Step 2: $\text{max} = a$.

-Step 3: **If** $b > \text{max}$
 $\text{max} = b$.

-Step 4: **If** $c > \text{max}$
 $\text{max} = c$.

-Step 5: **Output** max.

* Compute factorial of N:

-Step 1: **Input** N.

-Step 2: $S = 1$.

-Step 3: **While** $N > 1$
 $S = S * N$.
 $N = N - 1$.

-Step 4: **Output** S.



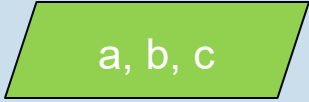
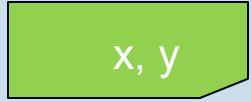

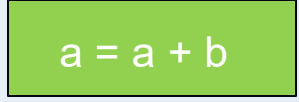
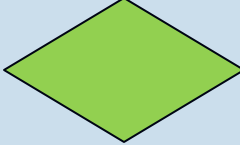
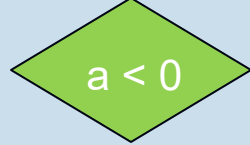

Computer algorithm



■ Algorithm representations:

■ Flowchart:

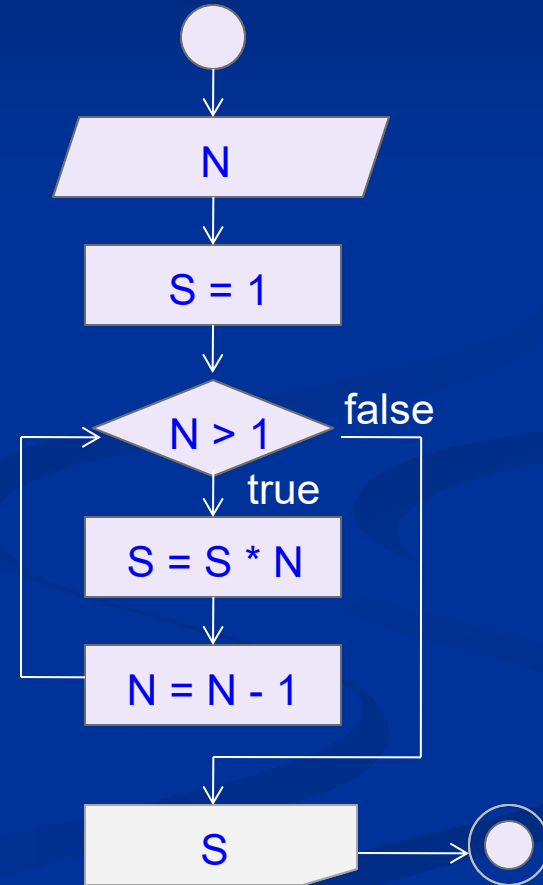
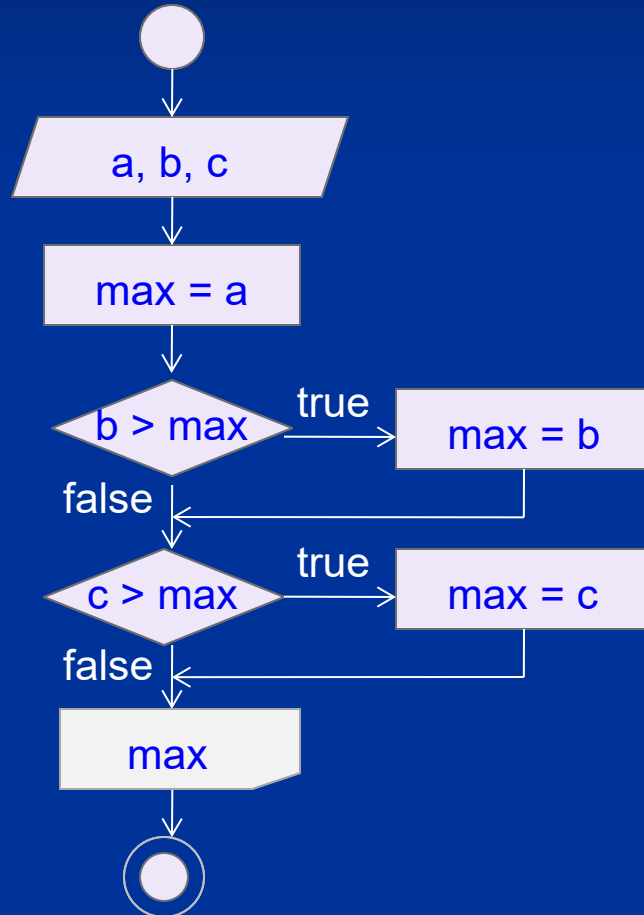
- Use diagram.
- Visualize flow of process.
- Notations:

Intrusions	Notations	Example
Read input Write output	 	 
Arithmetic operations		
Condition		
Transition		

Computer algorithm



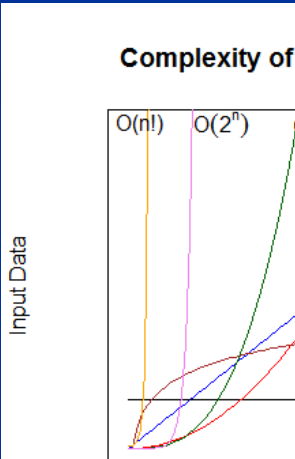
■ Algorithm representations:

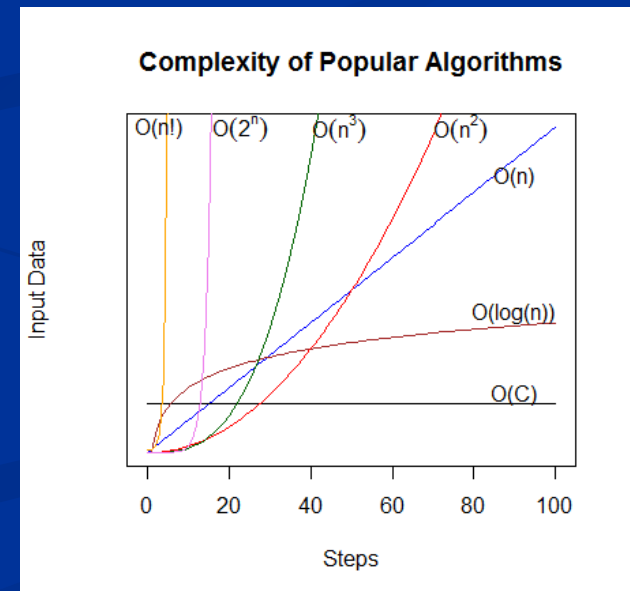




■ Algorithm efficiency:

- Complexity:

- Algorithm performance varies with different inputs.
 - Big-O notation: $O()$, growth rate.
 - Time complexity: input \sim instructions.
 - Space complexity: input \sim memory.
 - Classes of complexity:
 - Constant: $O(1)$.
 - Logarithmic: $O(\log(n))$.
 - Linear: $O(n)$.
 - Polynomial: $O(n^2)$, $O(n^3)$, ...
 - Exponential: $O(2^n)$.
 - Factorial: $O(n!)$.
- 



Computer algorithm



■ Algorithm efficiency:

■ Popular algorithm complexities:

Algorithm	Complexity	Form
Finding max (3 numbers)	$O(1)$	Constant
Binary Search(N elements)	$O(\log(n))$	Logarithmic
Factorial $N!$	$O(n)$	Linear
Sorting (N elements)	Bubblesort: $O(n^2)$ Quicksort: $O(n \cdot \log(n))$	Polynomial
Traveling Salesman (N city)	$O(K^n)$ $O(n!)$	Exponential Factorial

■ Simplicity:

- Easier to understand and implement.
- Reduce cost in debugging and maintenance.



■ Computer programming:

- Write instructions for computer to solve problem.
- Programming cycle: edit, compile, run, debug.
- Programming language:
 - Intermediate language to communicate with computer.
 - Low-level vs high-level.
 - Trends: scripting languages, AI prompts.
- Programming environment:
 - Platform of tools supporting programming tasks.
 - Editor, compiler, terminal, debugger.
 - IDE: Integrated Development Environment.





■ Computer algorithm:

- Finite steps to solve a problem on Turing machine.
- Turing machine: abstract machine and instructions.
- Basic instructions:
 - Input, output.
 - Arithmetic operations, comparisons.
 - Condition, loop.
- Representations: pseudo-code, flowchart.
- Efficiency:
 - Complexity: Big-O notation, growth rate.
 - Simplicity: easy to understand or implement.

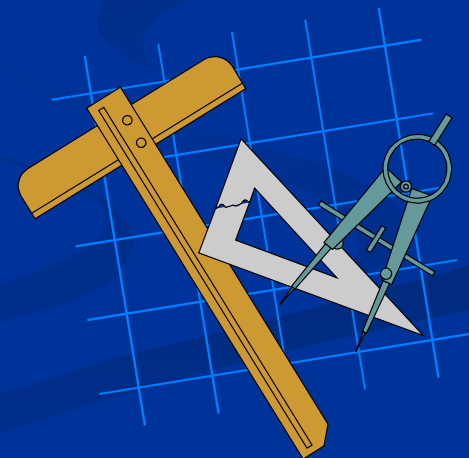




■ Practice 1.1:

Write algorithm to compute person age as follow:

- Input person birth-year.
- Compute and output person age.

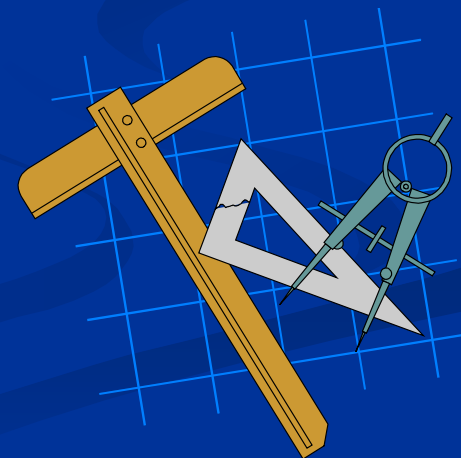




■ Practice 1.2:

Write algorithm to compute lucky number of a car as follow:

- Input car registration number (a 5-digit number).
- Compute and output the lucky number.

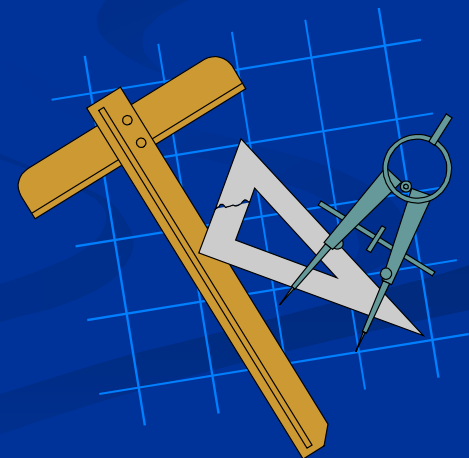




■ Practice 1.3:

Write algorithm to simulate a calculator as follow:

- Input two integers.
- Input an operation (+, -, *, /).
- Perform the operation on two integers and output result.

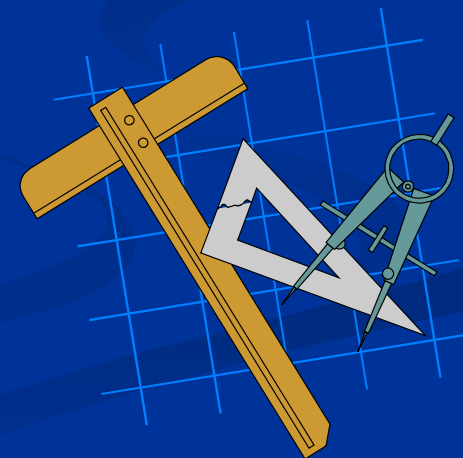




■ Practice 1.4:

Write algorithm to rank a student as follow:

- Input literature, math, physics points.
- Calculate student GPA.
- Rank student based on GPA:
 - + Excellent: $\text{GPA} \geq 8.5$.
 - + Good: $\text{GPA} \geq 7.0$.
 - + Fair: $\text{GPA} \geq 5.0$.
 - + Failed: $\text{GPA} < 5.0$.
- Output GPA and rank.

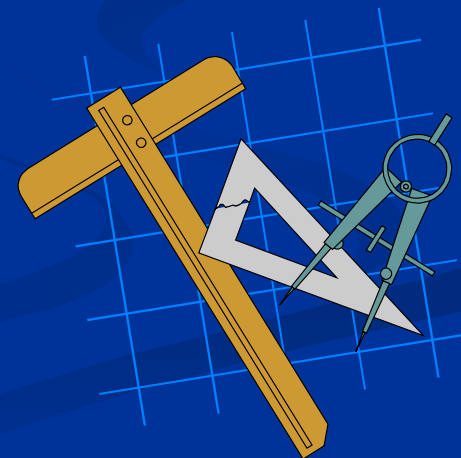




■ Practice 1.5:

Write algorithm to count number of days in a month as follow:

- Input a month and a year.
- Count number of days in that month and output result.





■ Practice 1.6:

Write algorithm to compute exponentiation of an integer as follow:

- Input x and n .
- Compute x^n and output result.

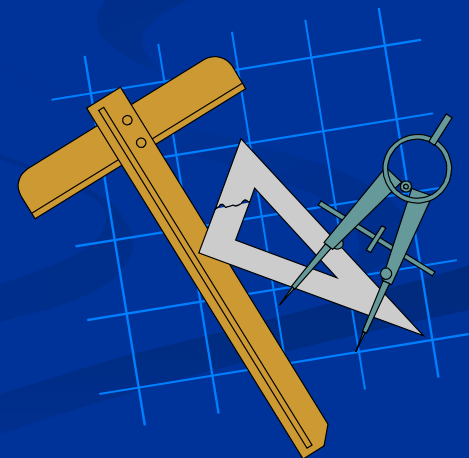




■ Practice 1.7:

Write algorithm to do the following:

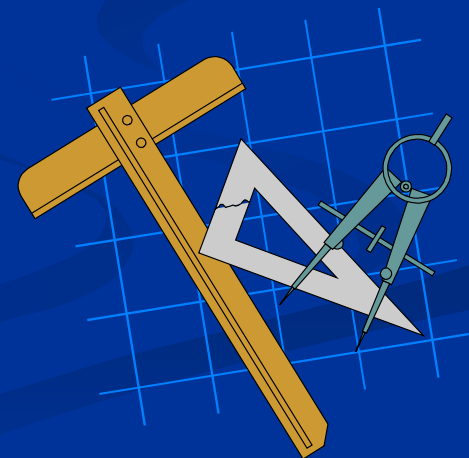
- Input 2 integers x , n ($n > 0$).
- Calculate and output $S = x + x^2 + \dots + x^n$.





■ Practice 1.8:

Set up a C/C++ programming environment on your computer.





■ Practice 1.9:

Edit, compile and run the following C/C++ program.

What does the program ask for?

What does the program print to screen?

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int  a, b, c, max;
```

```
    printf("Enter 3 integers = ");
```

```
    scanf("%d %d %d", &a, &b, &c);
```

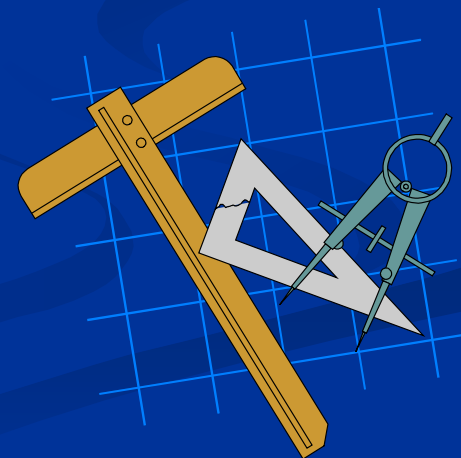
```
    max = a;
```

```
    if (b > max) max = b;
```

```
    if (c > max) max = c;
```

```
    printf("max = %d", max);
```

```
}
```





■ Practice 1.10:

Edit, compile, run and debug the following C/C++ program.

a) What does the program ask for and print to screen?

b) When enter $n = 10$, list the values of s when $n = 3$.

```
#include <stdio.h>
```

```
int main()
{
    int      n;
    long long s;
    printf("Enter an integer = ");
    scanf("%d", &n);
    for (s = 1; n > 0; n--)
        s = s * n;
    printf("Factorial = %lld ", s);
}
```

