

Computers Programming Course 6

Iulian Năstac

Recap from previous course

Data types

- four basic arithmetic type specifiers:
 - char
 - int
 - float
 - double
 - void
- optional specifiers:
 - signed,
 - unsigned
 - short
 - long

Recap

Variables

- **Variables** are simply names used to refer to some location in memory.
- Types of variables:
 - Local variables
 - Global variables

Recap

A more comprehensive classification

- **Automatic variables**
 - variables which are allocated and deallocated automatically when program flow enters and leaves the variable's context
 - An automatic variable is a variable defined inside a function block
- **External variables**
 - variable defined outside any function block
- **Static local variables**
 - variables that have been allocated statically — whose lifetime extends across the entire run of the program
- **Register variables**
 - register allocation is the process of assigning a large number of target program variables onto a small number of CPU registers

Recap

Standard I/O routines

- are substitute for missing of I/O instructions
- C programming language provides many standard library functions for input and output.
- These functions make up the bulk of the C standard library header `<stdio.h>` (also in `<conio.h>` and `<stdlib.h>`)

synthesis

Output functions	Input functions
printf	scanf
puts	gets
putchar	getchar
putch	getch getche

Recap

Expressions

- An expression in a programming language is a combination of explicit values, constants, variables, operators, and functions that are interpreted according to the particular **rules of precedence** and of association for a particular programming language, which computes and then produces another value.
- This process, like for mathematical expressions, is called evaluation.
- The value can be of various types, such as numerical, string, and logical.

Operators in C

- Programming languages typically support a set of operators, which differ in the calling of syntax and/or the argument passing mode from the language's functions.
- C programming language contains a fixed number of built-in operators.

1	() [] -> . ::	Grouping, scope, array / member access
2	! ~ - + * & sizeof <i>type cast</i> ++x - -x	(most) unary operations, sizeof and type casts
3	* / %	Multiplication, division, modulo
4	+ -	Addition and subtraction
5	<< >>	Bitwise shift left and right
6	< <= > >=	Comparisons: less-than, ...
7	== !=	Comparisons: equal and not equal
8	&	Bitwise AND
9	^	Bitwise exclusive OR
10	 	Bitwise inclusive (normal) OR
11	&&	Logical AND
12	 	Logical OR
13	?: = += -= *= /= %= &= = ^= <<= >>=	Conditional expression (ternary) and assignment operators
14	,	Comma operator

1. Brackets and data structure operators in C

		Associativity
()	<ul style="list-style-type: none">- Function call- Induce a priority in an expression	Left-to-right
[]	- Array subscripting	
.	Element selection by reference	
->	Element selection through pointer	

2. Unary Operators

		Associativity
++	Suffix increment	Left-to-right
--	Suffix decrement	
++	Prefix increment	Right-to-left
--	Prefix decrement	
+	Unary plus	
-	Unary minus	
!	Logical NOT	
~	Bitwise NOT (One's Complement)	
(type)	Type cast	
*	Indirection (dereference)	
&	Address-of	
sizeof	Size-of	

3. Multiplication operators

		Associativity
*	Multiplication	Left-to-right
/	Division	
%	Modulo (remainder)	

Example:

int a,b;

int E1, E2 ;

....

*E1 = (a/b)*b ;*

*E2 = (a/b)*b + a%b ;*

4. Additive operators

		Associativity
+	Addition	Left-to-right
-	Subtraction	

5. Shifting operators

		Associativity
<<	Bitwise left shift	Left-to-right
>>	Bitwise right shift	

Ex.: `x = y << 2;`

assigns x the result of shifting y to the left by two bits.

Notes:

<< shift to the left (the left operand) with a number of binary positions indicated by the right operand

>> shift to the right (the left operand) with a number of binary positions indicated by the right operand

- The remaining bits become **0**
- The left operand must be integer
- The right operand is converted to integer

6. Relational operators

		Associativity
<	Less than	Left-to-right
<=	Less than or equal to	
>	Greater than	
>=	Greater than or equal to	

7. Equality operators

		Associativity
==	Equal to	Left-to-right
!=	Not equal to	

8. Logic bit operators

		Associativity
&	Bitwise AND	Left-to-right
^	Bitwise XOR (exclusive or)	
 	Bitwise OR (inclusive or)	

9. Logic operators

		Associativity
&&	Logical AND	Left-to-right
 	Logical OR	Left-to-right

10. Conditional operator

- In C programming, ... **?** ... **:** ... is a ternary operator

Format:

condition **?** *value_if_true* **:** *value_if_false*

- The *condition* is evaluated *true* or *false* as a Boolean expression
- associativity: **Right-to-Left**
- Ex.:
variable = condition **?** value_if_true **:** value_if_false ;

The **?** **:** operator is similar, in a way, with conditional expressions (***if-then-else*** format).

Note:

- One should ensures the right syntax of an expression that contains conditional operators.

For example:

$E = ET1 ? (ET2 ? e1 : e2) : e3;$

is written in the right way, while

$E = ET1 ? e1 : ET2 ? e2 : e3;$

is confusing.

11. Assignment operators

		Associativity
=	Direct assignment	Right-to-left
+=	Assignment by sum	
-=	Assignment by difference	
*=	Assignment by product	
/=	Assignment by quotient	
%=	Assignment by remainder	
<<=	Assignment by bitwise left shift	
>>=	Assignment by bitwise right shift	
&=	Assignment by bitwise AND	
^=	Assignment by bitwise XOR	
=	Assignment by bitwise OR	

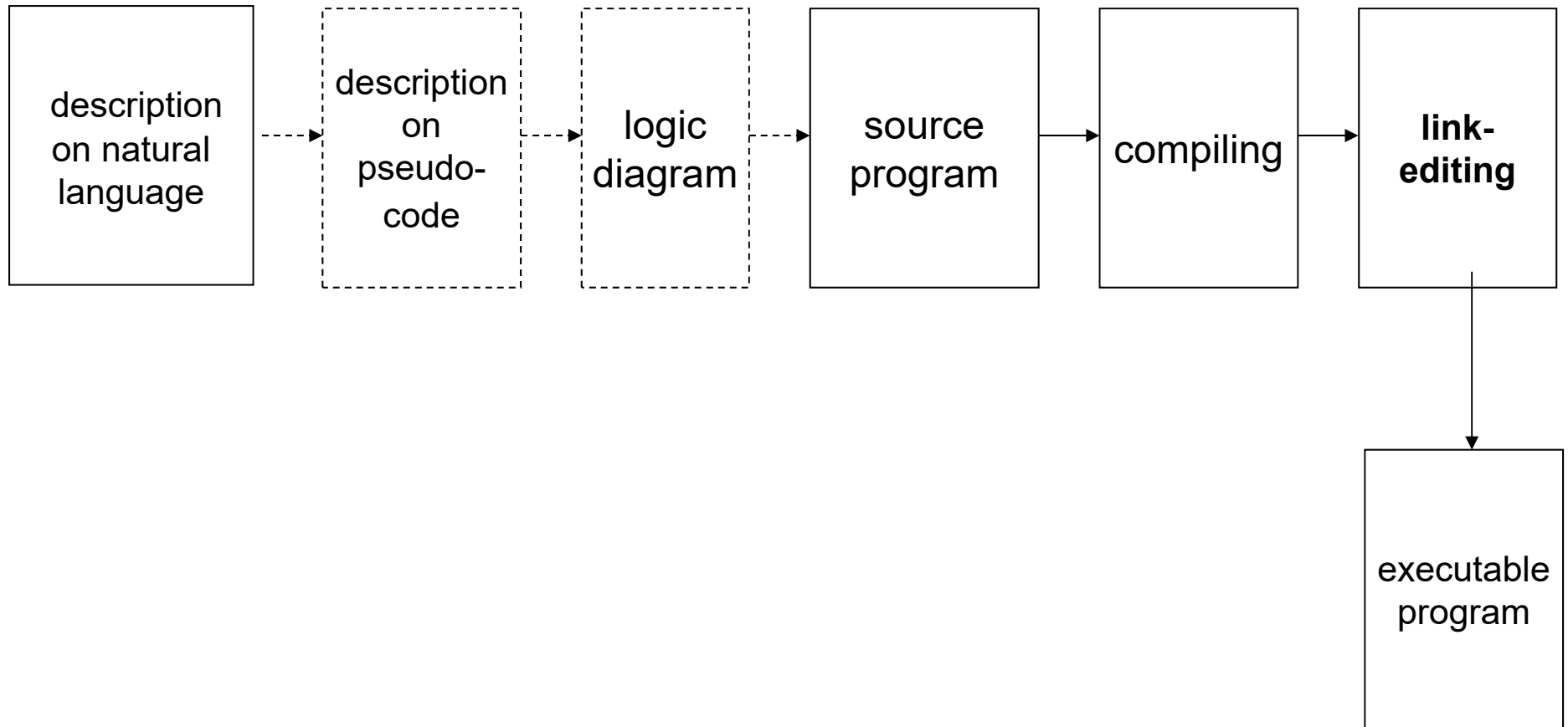
12. Comma

- **comma acts as separator between function parameters or variables**
- **associativity: Left-to-right**

Note that comma cannot be used in indexing multidimensional array

Ex.: the code $A[i, j]$ evaluates to $A[j]$ with the i discarded, instead of the correct $A[i][j]$

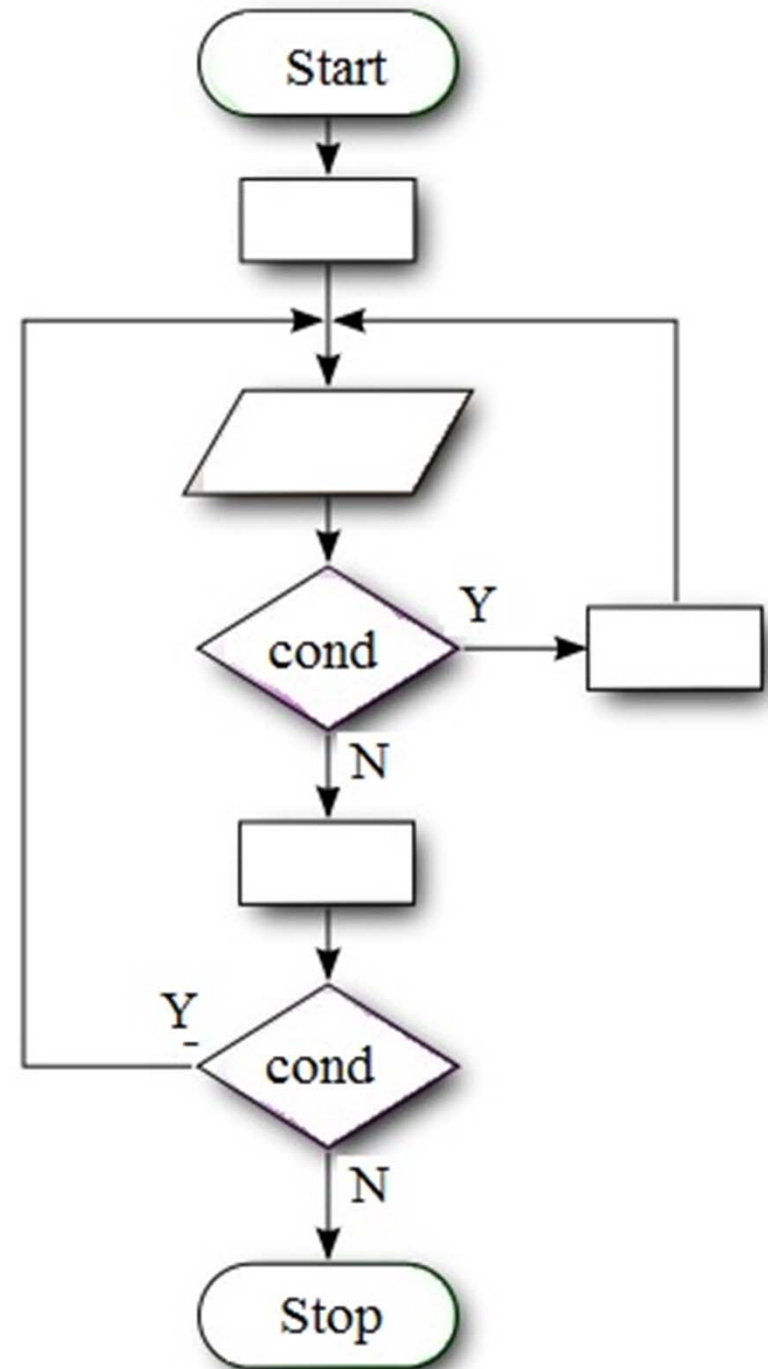
The design of a program includes several steps



Flow chart

(logic diagram)

- A **flow chart** is a schematic representation of an algorithm or a process, or the step-by-step solution of a problem.
- Flow charts use suitably annotated geometric figures connected by flow lines for the purpose of designing or documenting a process or program.



A typical flowchart may have the following kinds of symbols:

- **Start** and **Stop** symbols - represented as circles, ovals or rounded (fillet) rectangles.
- **Arrows** - showing "flow of control". An arrow coming from one symbol and ending at another symbol represents that control passes to the symbol the arrow points to.
- **Generic processing steps** -represented as rectangles.
- **Input/Output** - represented as a parallelogram.
- **Conditional or decision** -represented as a diamond (rhombus) showing where a decision is necessary, commonly a Yes/No question or True/False test.
- etc.



Flowchart start / stop



Flowchart process



Flowchart connector


















Flowchart selection










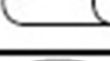








Flow line

Flowchart Symbol Cheat Sheet

Flowchart Symbol	Name (Alternates)	Description
	Process	An operation or action step.
	Terminator	A start or stop point in a process.
	Decision	A question or branch in the process.
	Delay	A waiting period.
	Predefined Process	A formally defined sub-process.
	Alternate Process	An alternate to the normal process step.
	Data (I/O)	Indicates data inputs and outputs to and from a process.
	Document	A document or report.
	Multi-Document	Same as Document, except, well, multiple documents.
	Preparation	A preparation or set-up process step.
	Display	A machine display.
	Manual Input	Manually input into a system.
	Manual Operation	A process step that isn't automated.
	Card	A old computer punch card.
	Punched Tape	An old computer punched tape input.

Flowchart Symbol Cheat Sheet

Flowchart Symbol	Name (Alternates)	Description
	Connector	A jump from one point to another.
	Off-Page Connector	Continuation onto another page.
	Transfer	Transfer of materials.
	Or	Logical OR
	Summing Junction	Logical AND
	Collate	Organizing data into a standard format or arrangement.
	Sort	Sorting of data into some pre-defined order.
	Merge (Storage)	Merge multiple processes into one. Also used to show raw material storage.
	Extract (Measurement) (Finished Goods)	Extract (split processes) or more commonly - a measurement or finished goods.
	Stored Data	A general data storage flowchart symbol.
	Magnetic Disk (Database)	A database.
	Direct Access Storage	Storage on a hard drive.
	Internal Storage	Data stored in memory.
	Sequential Access Storage (Magnetic Tape)	An old reel of tape.
	Callout	One of many callout symbols used to add comments to a flowchart
	Flow Line	Indicates the direction of flow for materials and/or information

Pseudocode

- **Pseudocode** is an informal high-level description of the operating principle of a computer program or other algorithm.
- No standard for pseudocode syntax exists, as a program in pseudocode is not an executable program.
- A programmer who needs to implement a specific algorithm, especially an unfamiliar one, will often start with a pseudocode description, and then "translate" that description into the target programming language and modify it to interact correctly with the rest of the program.

Pseudocode may vary widely in style

- Since, pseudocode generally does not actually obey the syntax rules of any particular language, there is no systematic standard form.
- Popular syntax sources include Pascal, BASIC, C, C++, Java, Lisp, and ALGOL. Variable declarations are typically omitted.

C style pseudo code

void function fizzbuzz

for (i = 1; i<=100; i++)

{ set print_number to true;

if i is divisible by 3

print "Fizz" ;

set print_number to false;

if i is divisible by 5

print "Buzz" ;

set print_number to false;

if print_number, print i;

print a newline;

}

Instructions (Flow Control)

- There are several flow control statements in C programming language.
- Basically, C instructions can be organized as:
 - Selection instructions
 - Loop instructions
 - Jump instructions
 - Label instructions
 - Expression instructions
 - Block instructions