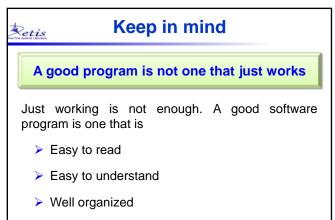
Programming style



> Well commented and documented

Setis Why paying attention to style?

Programming style is fundamental for many reasons:

- 1. It simplifies program reading & comprehension;
- 2. It facilitates program maintenance;
- 3. It reduces the possibility of making mistakes;
- It allows quickly identifying syntactic and semantic errors;
- 5. It avoids irritating project reviewers.

Hence, adopt these rules since the beginning!

Style rules

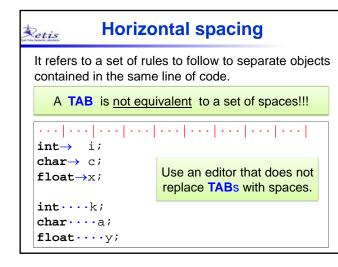
For all these reasons, programmers should follow some general guidelines aimed at improving source readability.

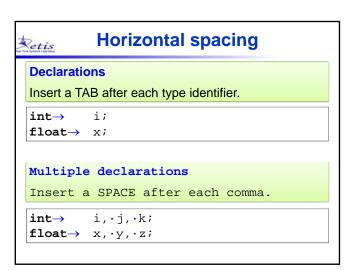
In particular, they concern the following aspects:

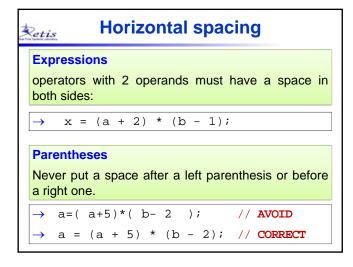
- 1. Horizontal spacing
- 2. Vertical spacing
- 3. Indentation

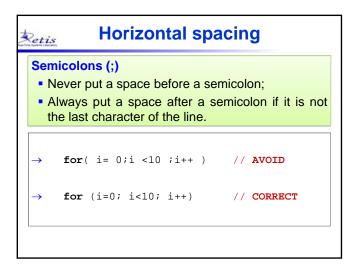
Retis

- 4. Comments
- 5. Code organization

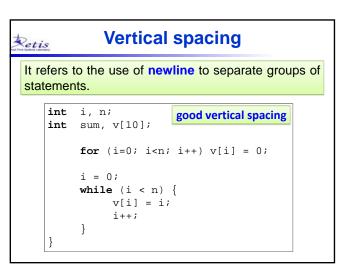


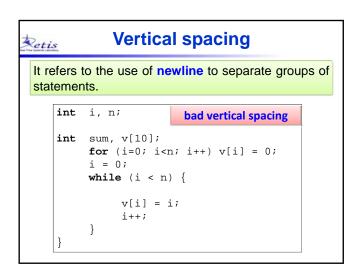


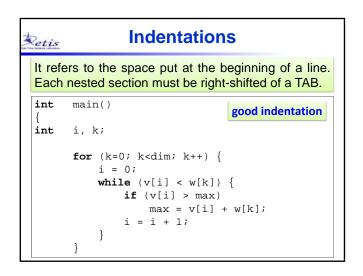


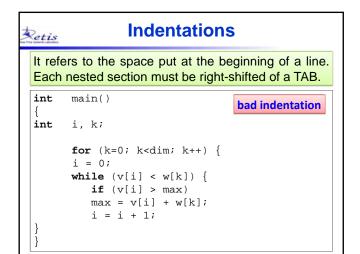


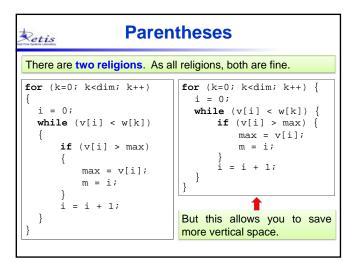
Horizontal spacing		
 Conditional statements Always put a space between the instruction name and the parenthesis specifying the condition. 		
\rightarrow	if (a < b)x = a; // AVOID	
\rightarrow	if (a < b) x = a; // CORRECT	
\rightarrow	<pre>while (v[i] < 0) i = i + 1;</pre>	
\rightarrow	<pre>for (i=0; i<10; i++) v[i] = 0;</pre>	

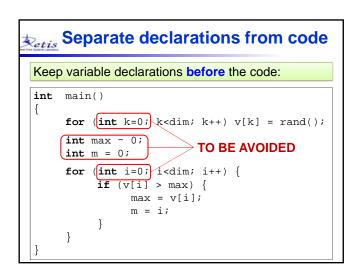


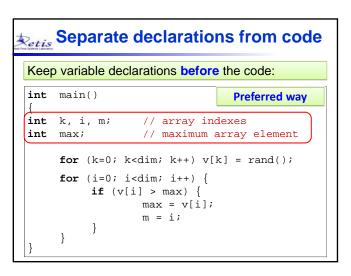


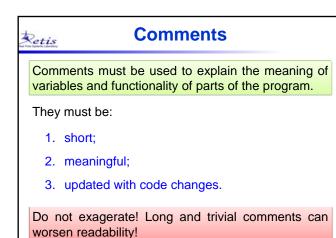


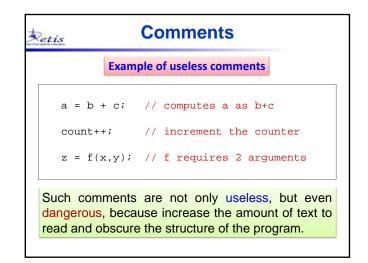












Comments

They should be inserted in the following situations:

- 1. at the <u>beginning of a file</u>, to explain its contain and functionality;
- <u>next to each declaration</u> of variable, constant, or data structure, to explain its meaning;
- <u>before each function</u>, to explain its functionality, the meaning of each argument and the return value (if any);
- 4. before non trivial operations;

Types of comments

Depending on their position and length, different styles should be used to write a comment.

Comments to variables

Retis

Should be on the right of the declaration, separated by one or more TABs:

```
int r; // circle radius
int x, y; // center coordinates
```

Types of comments

One line comments

Retis

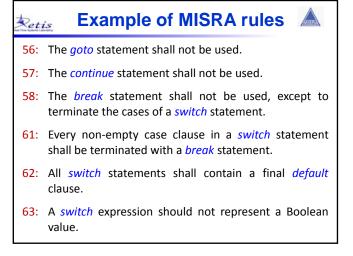
They should be written just before the instruction (or group of instructions) to be explained:

// initialize array v
for (i=0; i<dim; i++) v[i] = 0;</pre>

Types of comments		
Long comments They should be written before the code to be explained, highlighting them with some border.		
<pre>// // Function even(n) returns 1 if n // is an even number, 0 otherwise //</pre>		
<pre>int even(int n) { if (n % 2 == 0) return 1; else return 0; }</pre>		



URL: http://caxapa.ru/thumbs/468328/misra-c-2004.pdf



Example of MISRA rules

- 65: Floating point variables shall not be used as loop counters.
- 67: Numeric variables being used within a for loop for iteration counting should not be modified in the body of the loop functions.
- 69: Functions with a variable number of arguments shall not be used.
- 82: A function should have a single exit point.
- 101: Pointer arithmetic shall not be used.

Retis

Retis

118: Dynamic heap memory allocation shall not be used.

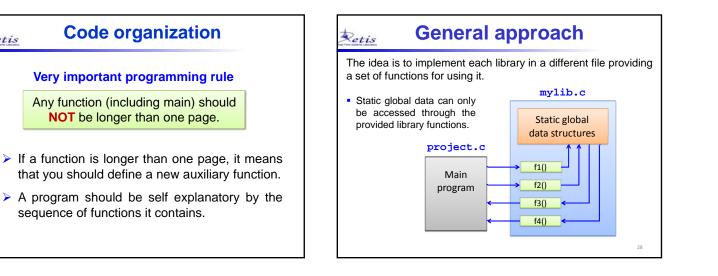
Code organization

Organize code according to the following order:

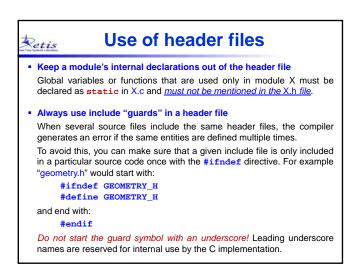
- 1. Header files (standard first, yours later)
- 2. Global constants (separate them into groups)
- 3. Function prototypes

Retis

- 4. Global data structures
- Functions definitions
- 6. Tasks definitions
- 7. Main function

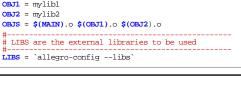


Retis	Use of header files		
C programs are normally organized into separately compiled modules.			
Module:	group of declarations and functions that are <u>developed and</u> <u>maintained separately</u> and possibly reused in different projects. Good examples are the math and string Standard Libraries.		
Except for the main module, each module X consists of a			
source file	(X.c): contains global variable definitions, initializations and function definitions.		
header file	<u>e</u> (X.h): contains only : structure type declarations, function prototypes, and extern global variable declarations.		
The X.	c file must include the X.h file		
 Global variables must be declared as extern in the X.h file 			
	modules can access the functionality in module X by inserting ude "X.h"		
	s to be compiled only if changed; the rest of the times the linker will s code into the final executable without needing to recompile it.		



Use of header files **Makefile** Retis Retis A.h should include all strictly needed header files, but no more #-----# Target file to be compiled by default #-----If a structure type defined in module X is used as a member variable of a structure type A, then you must include X.h in A.h, so that the compiler MAIN = balls #-----# CC is the compiler to be used #----knows how large the X member is. However, do not include header files needed only by the .c file. CC = gcc For instance <math.h> is usually needed only by the function definitions, therefore it should be included in the .c file, not in the .h file. File A.c should first include its A.h, then the other required headers #----# OBJS are the object files to be linked #-----• Never include a source .c file for any reason! OBJ1 = mylib1 OBJ2 = mylib2 **OBJS = \$(MAIN)**.o **\$(OBJ1)**.o **\$(OBJ2**).o Read more on:

http://www.umich.edu/~eecs381/handouts/CHeaderFileGuidelines.pdf



Makefile	
# # Dependencies	
#	
\$(MAIN): \$(OBJS) \$(CC) -0 \$(MAIN) \$(OBJS) \$(LIBS) \$(CFLAG	S)
\$(MAIN).0: \$(MAIN).C	
\$(CC) -c \$(MAIN).c	
\$(OBJ1).o: \$(OBJ1).c	
\$(CC) -c \$(OBJ1).c	
\$(OBJ2).o: \$(OBJ2).c	
\$(CC) -c \$(OBJ2).c	
# Command that can be specified inline: make cle	ean
#clean:	
rm -rf *o \$(MAIN)	
	33

Retis Important guidelines			
	Your project will be evaluated also based on the level of compliance with the following guidelines		
1.	Write code according to the given style rules		
2.	Do not use dynamic memory allocation		
3.	Define all local variables at the beginning of a function		
4.	Avoid using nested critical sections (use more local variables)		
5.	Avoid numeric constants in the code (exceptions: 0, 0.5, 1, 2,)		
6.	Avoid functions longer than one page (use new functions)		
7.	Avoid lines longer than 80 characters (lines can be split)		
8.	Never use the goto and continue statements		
9.	Use break only in the switch statement		
10.	Organize your program in different source files (no more than 3)		