

Introduction

Creating a computer program only makes sense if the volume of calculations would exceed manual possibilities. Such situations may arise if many similar problems need to be solved, if complex and time-consuming calculations are required, or if large amounts of data need to be processed. Some specific knowledge is needed to create a calculation program. The efficiency and performance of the software will depend on the knowledge and experience of those who collaborate to make it. In the following we will focus on the creation of simple applications using a high-level programming language (close to spoken language).

The programming stages are generally summarised by 3 phases: conception (the logical level of problem solving with the development or choice of the appropriate algorithm), coding (transcription of the algorithm into a machine-accessible programming language), testing and implementation (checking correctness with test data and fine-tuning the program). These phases can be carried out both in an empirical way, and in a structured manner, which is more efficient than the first (allowing for structured development and testing). Preferring the structured approach, instead of the 3 generic phases mentioned above, we present the following sequence of steps, considered necessary to develop a program:

- problem recognition and definition;
- selection and description of the solving method;
- translating the description of the method into a programming language (creation of the source file);
- the making of the program by compiling (translating the source file into machine code, generating the object image) and link editing (filling the object image with elements from the language library, generating the executable file);
- running and testing the resulted software.

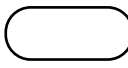
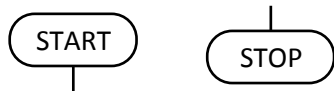
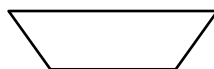
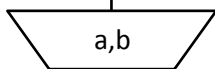

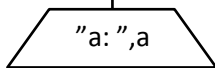
As these steps were discussed in the previous semester (in the "Applied Computer Science" course), we will focus on the presentation of structured logic schemes and the use of the Fortran language to create console applications.

Structured flowcharts (logic schemes)

A flowchart is a graphical tool that can be used to represent the steps of an algorithm in the form of blocks (symbols) connected by lines. In order to use this tool in a structured way, some fundamental principles must be known, such as:

- Schemes are drawn and read from top to bottom (exceptions are marked with arrows).
- Blocks can have only one entry point (except the starting one, which has only one exit), and the number of exits depends on the type of block: modules and those representing input/output operations or attribution only one exit, conditional ones depending on the type of expression (logical 2, for true or false; arithmetic 3, for negative, null or positive), and the end block has no exit point.
- Structured primitives (having a single entry point and a single exit point) are used as much as possible in the composition of the schemes, chaining them sequentially.

A module can contain anything, provided it has only one input and one output. The contents of a module must be detailed separately, where appropriate. The symbols used in the composition of flowcharts are shown in the following table:

Use	Symbol	Variants (examples)
Start block (marked START) or end block (marked STOP)		
Input block (marked with the elements to be read)		
Output block (marked with elements to be written)		

Alternative to input/output block (note whether input or output)		
Attribution block (contains a single expression, whose value is attributed to the variable on the left)		
Decision blocks (output is branched according to the type of expression evaluated, conditions are also marked)	 	
Module or procedure block (marked with module name)	 	
Inner connector (for interrupt or continuation within the same page, marked correspondingly)		
External connector (for interrupt or continuation between different pages, with matching mark)		

To create a program, you need a text editor (ASCII) and a compiler and link editor package. You can choose **Code::Blocks** (<https://www.codeblocks.org/>) for Windows, Linux and MacOS operating systems, or **JDOODLE** (<https://www.jdoodle.com/execute-fortran-online/>) if you prefer to work online (through a browser). For Windows operating systems, we recommend the **Force 2** package with the **G95 Fortran** compiler (<https://force.lepsch.com/p/download.html>), which will also be used in class. To be able to use free-form in Force 2, after installation you will need to fill in the Compilation Options with `-ffree-form` (as illustrated) and experiment with the convenient running mode (Batch command file, direct EXE, or Console):

