ASC 8

Dnul vancea e cam racit =(

Data definition directives

Concept of variable:

* Variable – we express the fact that the contents of that item is modifiable
* A variable is **not** a symbol, it is not just a name and a value and we must never think of it as only that
* A variable is a pair of the memory address of some location (the main feature) and the contents
* The symbol is the NAME of the variable, but in case of assembly, it is just a symbol denoting the MEMORY ADRESS of that location
* From a structural point of view, a variable is 4 things that represent its’ structure
  + Name (OPTIONAL)
  + Set of attributes
    - Is in fact also represented of 4 attributes, 3 of which are independent while the fourth is a resultant, an output of the first 3
      * Type
        + (the main attribute, In assembly being Its’ size)
        + Defines the domain of values
      * Visibility domain (scope) - DoV
        + A spacial attribute
        + Intervalul de program-sursa masurat in distante in care acea variabila este accesibila
        + The interval of source-program measured in the distance in which that variable is accesible
      * Lifetime Variable (Extent) - DdV
        + How much does it live, from when you give it birth to when it dies (The memory in which the variable is stored gets cleared)?
        + The moment a variable is ‚born’ is when it’s allocated and the moment it dies is when the space in which the variable is allocated is cleared
        + Measured in seconds
      * Memory class
        + An attribute that dictates the regime of allocation, how a variable is allocated
        + A memory class is usually implicit, but in c and c ++ you have the ability as a programmer to explicitely change the class
        + In c there are 4 memory classes:

**Auto** (the memory class will adapt itself in an automatic way and the memory location will be allocated inside a stack (if local) and in the Global Data Segment (if global or static))

**Register** (if you declare a variable as it and the processor has a free register for a variable, it wil automatically put it inside there, which is beneficial because they are way more accesible and easier to compute and refference)

Static (se refera la mecanismele import-export: If you put it in front of the function, that function becomes LOCAL TO THE MODULE, which is never the general case. Another use of static is that it safely allows you to use local variables in a local way. If you declare a static LOCAL variable, the visibility domain of that variable is still a function, but the lifetime of the variable is the whole runtime)

Extern/External (The linkeditor checks if you have the same variable with the same name in 2 modules – Impossible, but you can have a variable in a module and 16 other refferences to that variable. Extern is an import mechanism. It doesn’t allocate, it DECLARES it)

* + Reference (address)
    - Optional to formal variables
  + Value
    - If you have a refference you already have a value, So if you don’t declare a variable it will already have a residual value
    - Optional to formal variables
* The definition of a variable is represented by declaring its attributes (can be done many times) and its’ allocation (can be done ONCE)
* The definition must ALWAYS be unique, that’s why the linkeditor gives us an error

In the heap you allocate dynamic variables (a variable whose name (It doesn’t have a name) is allocated during runtime(?))

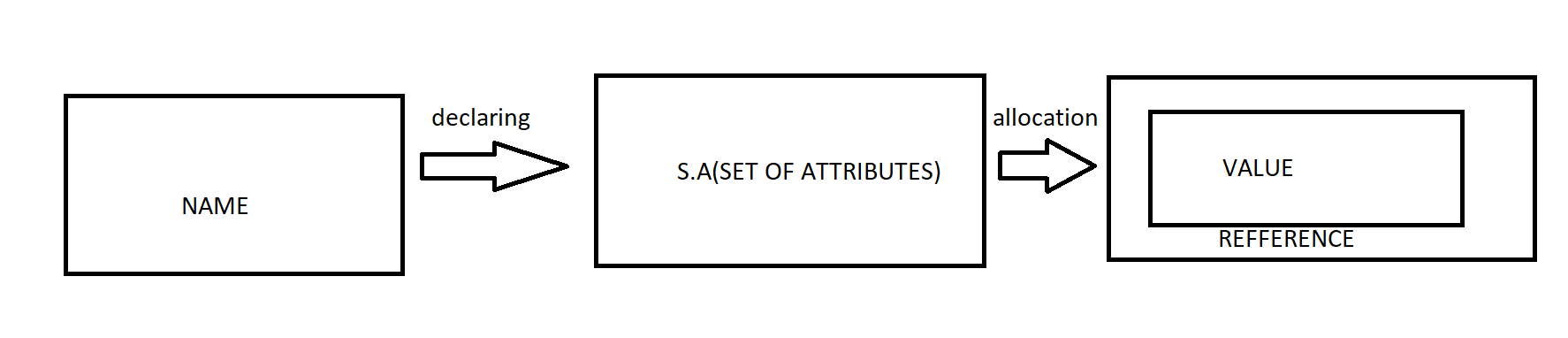
* A dynamic variable is NOT a pointer, but a variable that YOU as a programmer have a mechanism for giving ti birth :
* You use a POINTER and a function new
* P = new (...„SIZEOF”...)
* P is just used to manage and access the dynamic variable, being the only way to access it
* So p is the adress of the dynamic variable, the associated pointer of it, but not the dynamic variable itself
* At the end you write free (P) to remove/terminate/kill the dynamic variable

Allocation – means to associate the space necessary for that variable

Definition = declaring its’ attributes + allocation

Allocation is an unique process

Declaring the attributes can be done many times



**IMPORTANT**!

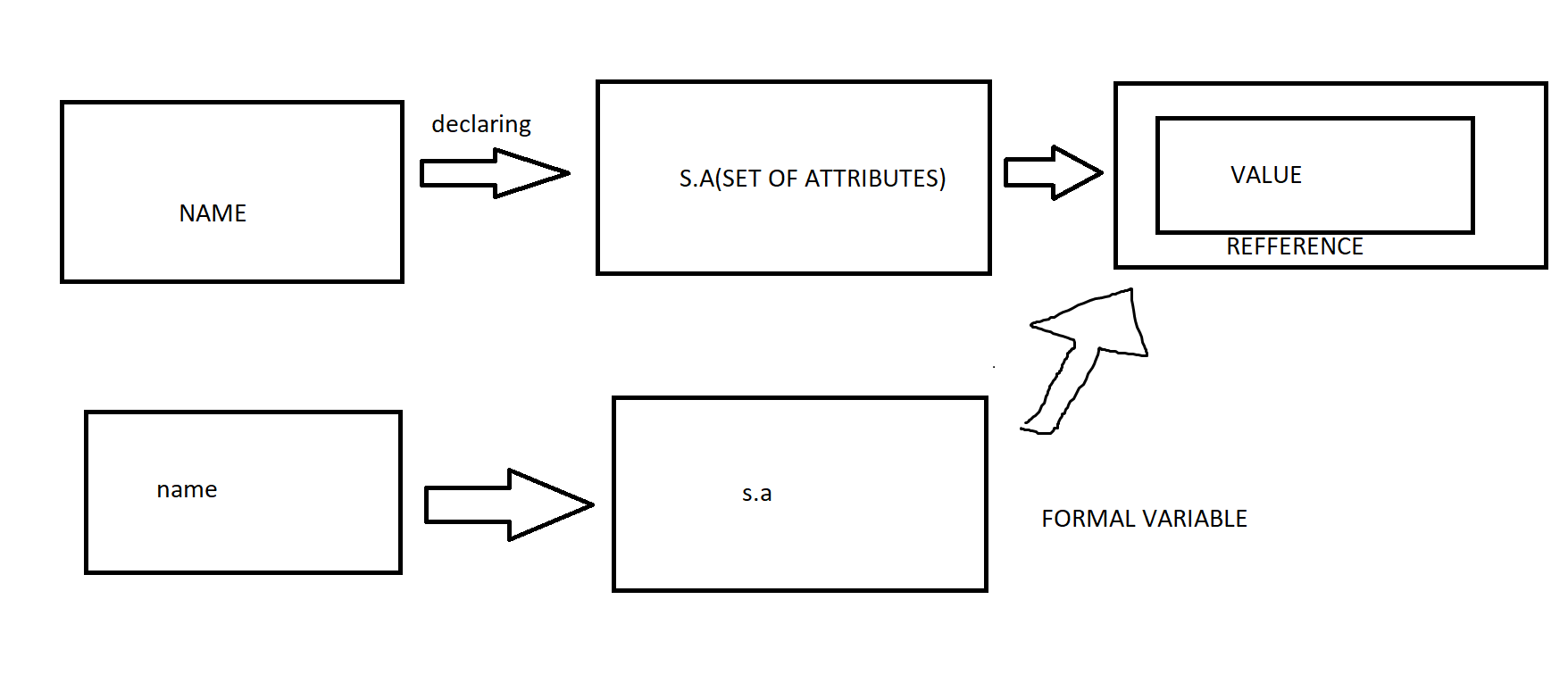
A GLOBAL VARIABLE LASTS THE ENTIRE RUNTIME

A LOCAL VARIABLE LASTS THE DURATION OF THE FUNCTION IN WHICH IT IS CALLED

THE FORMAL PARAMETERS OF A FUNCTION APPEAR IN A HEADDING

F (int a, int b, int c); a,b,c – formal parameters which have a name and a set of attributes but have NO REFFERENCE OR VALUE, THEY BELONG TO x,y,z

F (x,y,z)



Times – a dirrective that forces (?)

Equ – you don’t allocate space for it like when you declare data definitions

EXAMPLES:

* A1 db 0,1,2‚’xyz’ – the assembler generates bytes: 00 01 02 ‚x’ ‚y’ ‚z’ (78 79 7A)
* A1 db 300, „F” + 3 – in tasm there would have been a syntax error (300 = 1 2C so it doesn’t fit). In the nasm assemblor, it tolerates it while issuing a warning (byte data 300 exceeds bounds): 2C ‚F’+3
* A2 TIMES 3 db 44h – 3 bytes generated: 44 44 44
* A3 times 11 db 5,13 – 33 bytes generated: 5 1 3 5 1 3 5 1 3 ...
* A4 db a2+1 – the offset of a2 RELATIVE TO THE DATA SEGMENT, 8 (0,1,2,x,y,z,2c,’F’+3). In oly you will have 00401000h – decided by oly and linker: NOT 9, but SYNTAX ERROR; A4 db a2+1 ⬄ mob Ah a2
* A41 dw a2+1, ’bc’ – 09 10 ‚b’ ‚c’ (62h, 63h)
* A42 dw 1009h – SYNTAX ERROR
* A5 dd a2+1,’bcd’ –
* A6 times 4 db ’13’ – ‚1’, ‚3’ – 2 bytes
* A61 times 4 dw ’13’ – no difference from A6. It doesn’t matter: ‚1’, ‚3’ – a word
* A7 db a2 – SYNTAX ERROR
* A8 dw a2 – RIGHT
* A9 dd a2 – 00401008
* A10 dq a2 – PLIN DE 0-URI
* A11 db [a2] – OPERANDS CAN PERFORM ONLY COMPUTATIONS WITH CONSTANTS DETERMINABLE AT ASSEMBLY TIME (IMMEDIATE VALUES AND OFFSET, NOT THE CONTENTS OF REGISTERS OR MEMORY) => **SYNTAX ERROR**
* A12 dw[a2] ] – OPERANDS CAN PERFORM ONLY COMPUTATIONS WITH CONSTANTS DETERMINABLE AT ASSEMBLY TIME (IMMEDIATE VALUES AND OFFSET, NOT THE CONTENTS OF REGISTERS OR MEMORY) => **SYNTAX ERROR**
* A13 dd dword [a2] ] – OPERANDS CAN PERFORM ONLY COMPUTATIONS WITH CONSTANTS DETERMINABLE AT ASSEMBLY TIME (IMMEDIATE VALUES AND OFFSET, NOT THE CONTENTS OF REGISTERS OR MEMORY) => **SYNTAX ERROR**
* A14 dq [a2] - ... => **SYNTAX ERROR**
* A15 dd eax - ... => **SYNTAX ERROR**
* A16 dd [eax] - ... => **SYNTAX ERROR**