More on parameter passing:
  a) Parameter association
  b) Default parameters
  c) Procedures as parameters
  2) Modules: Ada packages, C modules, C header files
  3) Programs
  4) Generics and Abstract Data Types (ADTs)

14. PROGRAM COMPOSITION II

1. Four levels of hierarchy can be distinguished:
   a) Blocks.
   b) Sub-programs (procedures and functions).
   c) Modules, packages and tasks.
   3) Programs.

PARAMETER ASSOCIATION

- The normal procedure when calling a routine is to list all the actual parameters in order separated by commas.
- This is called *positional parameter association*.
- This means that any actual parameters supplied must agree with the formal parameters in number, order and type.
- Most imperative languages (Ada and C included) support positional parameter association.
- The alternative is *keyword or named* association.

```plaintext
procedure POS_PARAM_ASSOC is
  -- ADD_UP procedure
  procedure ADD_UP(X, Y, Z : INTEGER) is
  begin
    PUT(X+Y+Z); NEW_LINE;
  end ADD_UP;
  -- Top level
  begin
    ADD_UP(2, 4, 6);
  end POS_PARAM_ASSOC;
```

```plaintext
procedure NAMED_PARAM_ASSOC is
  -- ADD_UP procedure
  procedure ADD_UP(X, Y, Z : INTEGER) is
  begin
    PUT(X+Y+Z); NEW_LINE;
  end ADD_UP;
  begin
    ADD_UP(X=>2, Y=>4, Z=>6);
    ADD_UP(Z=>6, X=>2, Y=>4);
    ADD_UP(2, Z=>6, Y=>4);
  end NAMED_PARAM_ASSOC;
```

```
```

NAMED PARAMETER ASSOCIATION

- Note that the parameters do not need to be ordered in a particular way.
- Note also that we can mix positional and named parameter association provided that the positional parameters precede the named parameters.
- The claimed advantage is that this gives a clearer reading of the code.
- Anybody reading the code does not need to know exactly the formal parameters used or the order in which they appear before being able to understand the significance of the call.
DEFAULT PARAMETERS

- In some language parameters may be given default values.
- For example Ada in parameters may be given a default values when the formal parameter is specified.
- in out parameters and out parameters may not have default values.

PROCEDURES AS PARAMETERS

- Foregoing all assumes that we wish to pass data items. Some languages support passing of procedures (e.g. Pascal).
- Example: in numerical analysis, where methods for finding (say) roots of functions etc. can be written to be independent of any particular routine, it is useful to be able to pass appropriate functions to solving routine.

MODULES

- Overview
- Creating and using a package (in Ada)
- Creating and using a C module
- Creating and using a C header (.h) file.

MODULES

- A number of related declarations of types, variables and routines can be grouped together into a module or package.
- The concept of modules (and modular programming) appeared in the 1970's in response to the increasing size of programs (in Modula-2 everything is a module).
package TIMES TWO is
   -- Specification
   function TIMES_TWO ( X : integer ) return integer;
   end TIMES TWO;
   -- Body
   package body TIMES TWO is
      function TIMES_TWO ( X : integer ) return integer is
         begin
            return ( X * 2 );
         end;
   end TIMES TWO;
end TIMES TWO;

CREATING AN ADA PACKAGE

Ada packages comprise two parts:
1) A specification part which gives the user information about the resources contained and how they are used.
2) A package body that details the resources

Note that the body is not visible to the user.

CREATING A C MODULE

#include <stdio.h>
#include "timesTwo.h"

void main(void) {
   int x = 4;
   printf("Two times %d = %d\n", x, timesTwo(x));
}

Creating a C header file

/* SPECIFICATION */
int timesTwo(int);
/* BODY */
int timesTwo(int x) {
   return (x * 2);
}

Compilation:
cc -Aa -c timesTwo.c
cc -Aa -c example.c
cc -Aa -o example.o timesTwo.o

Whereas a C module is first compiled to create an object file (.o) and later linked into the "main" program, a header file is "compiled in" at the same time as the "main" program.

Procedure is as follows, first create a .h file:

/* SPECIFICATION */
int timesTwo(int);
/* BODY */
int timesTwo(int x) {
   return (x * 2);
}

Include this in code as follows:

#include <stdio.h>
#include "timesTwo.h"

void main(void) {
   int x = 4;
   printf("Two times %d = %d\n", x, timesTwo(x));
}

Programs
The concept of programs has existed since the conception of computer programming.

With respect to C and Ada a program consists of one or more object files linked together.

Note that individual object files are not independent.

The main issues in program composition from object files are:
1) How to indicate which objects are part of the desired program — linking
2) Providing an answer to the question “where do we start processing from?”

Ada assumes that there exists a method outside the language which allows the programmer to specify the name of a procedure and which will then construct an executable binary program for that procedure containing all the necessary modules.

specifying program composition in ada

Ada assumes that there exists a method outside the language which allows the programmer to specify the name of a procedure and which will then construct an executable binary program for that procedure containing all the necessary modules.

Specifying Program Composition in C

C assumes that the programmer will specify all necessary user object files in a specialised invocation of the system linker.

This invocation searches the object files for a routine with the fixed external name main and constructs an executable binary program which, when called, will execute the routine main.

Specifying Program Composition in C

C assumes that the programmer will specify all necessary user object files in a specialised invocation of the system linker.

This invocation searches the object files for a routine with the fixed external name main and constructs an executable binary program which, when called, will execute the routine main.

Generics

Much of the code written by programmers is very specific.

For example any algorithm used to manipulate linked lists also specifies the type of the items in the list.

It is desirable to be able to write a general algorithm applicable to linked lists of any type.

This is achieved using a construct known as a generic.

A generic “unit” can be considered to be a template from which actual units can be created at compile time.

Ada supports generics, C does not.

Consider the following Ada procedure:

```ada
procedure SNAP (VAL_1, VAL_2 : in out INTEGER) is
    TEMP : INTEGER;
begin
    TEMP := VAL_1;
    VAL_1 := VAL_2;
    VAL_2 := TEMP;
end SNAP;
```

This “swaps” two integer values.

If we wanted to swap to floating point values or two characters we would require two further procedures.

Alternatively we can use a generic procedure.

Generic "swap" procedure:

```ada
-- Specification of generic parameters
package generic is
    type ELEMENT is private;
    procedure SNAP (VAL_1, VAL_2 : in out ELEMENT) is
        TEMP : ELEMENT;
    begin
        TEMP := VAL_1;
        VAL_1 := VAL_2;
        VAL_2 := TEMP;
    end SNAP;
end generic;
```
• This must now be compiled before it can be used.

```plaintext
with SWAP, CS_IO; use CS_IO;
procedure SWAP_THINGS is
  procedure SWAP_INTEGER is new SWAP(INTEGER);
  procedure SWAP_FLOAT is new SWAP(FLOAT);
  INT_1, INT_2 : INTEGER;
  FLOAT_1, FLOAT_2 : FLOAT;
begin
  INT_1 := 2; INT_2 := 4;
  SWAP_INTEGER(INT_1, INT_2);
  FLOAT_1 := 2.22; FLOAT_2 := 4.44;
  SWAP_FLOAT(FLOAT_1, FLOAT_2);
end SWAP_THINGS;
```

**ABSTRACT DATA TYPES**

• A data type defines a set of values and a set of operations that may be performed on those values.
• An abstract data type (ADT) links a type definition with the operations that can be applied to it.
• Values associated with an ADT can only be created and manipulated through the defined operations.
• The advantages offered by ADTs concern:
  - "information hiding" and
  - "software reuse".
• ADTs are an important concept in Object Oriented Programming and Functional Programming....

**SUMMARY**

1) More on parameter passing:
   a) Parameter association
   b) Default parameters
   c) Procedures as parameters
2) Modules: Ada packages, C modules, C header files
3) Programs
4) Generics and Abstract Data Types (ADTs)