Analyse et programmation langage ADA

Informatique 1ère année

Les exceptions

- Déclaration des exceptions
  - Exceptions prédéfinies
  - Exceptions déclarées par l'utilisateur

- Levée d'une exception
  - Automatique
  - Par l'utilisateur

- Traiter une exception

```
exception
    when CONSTRAINT_ERROR => put("il y a un probleme dans la fonction");
                             raise CONSTRAINT_ERROR;
```

Exceptions prédéfinies ADA 95

Constraint_Error : violation de tout type de contrainte (domaine, précision, indice)

. Variable hors des bornes de son type,
. Indice hors de son intervalle,
. Discriminant non-respecté.
. Division par zéro, -- Deprecated : NUMERIC_ERROR :
. Dépassement de capacité.

Programm_Error :

. violation d'une structure de contrôle,
. arrivée sur le end d'une fonction(pas de return)

Storage_Error (dépassement de la mémoire)

. Plus de mémoire disponible pour faire un "new".

Tasking_Error (traitement des tâches, parallélisme)

Numeric_Error (division par zéro dans ADA 83)

Exceptions prédéfinies ADA 95

ADA.IO_EXCEPTIONS : paquetage d'exceptions prédéfinies liées aux entrées-sorties

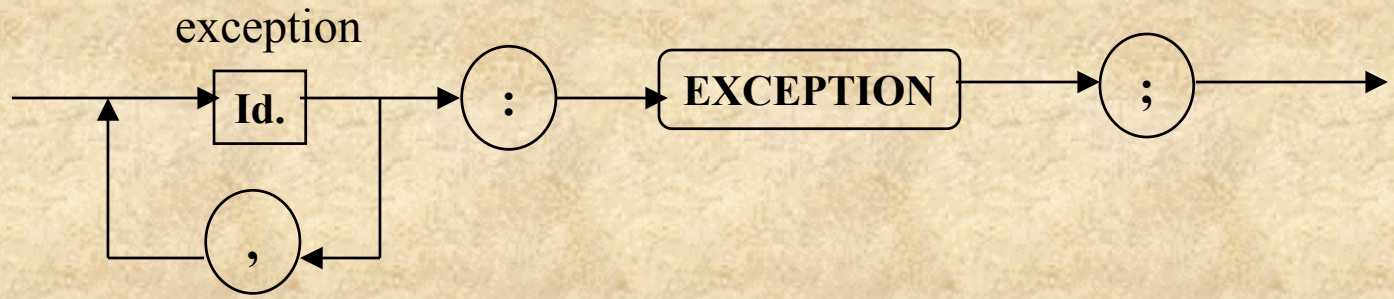
```
package Ada.IO_Exceptions is
   pragma Pure (IO_Exceptions);
   Status_Error  : exception;
   Mode_Error  : exception;
   Name_Error  : exception;
   Use_Error      : exception;
   Device_Error : exception;
   End_Error      : exception;
   Data_Error     : exception;  --activée si la valeur lue avec un get ne correspond pas au format attendu
   Layout_Error : exception;
end Ada.IO_Exceptions;
```

Exemple

```
N : Natural;

get(N);      -- A  → raised ADA.IO_EXCEPTIONS.DATA_ERROR

             -- –2  → raised CONSTRAINT_ERROR
```

Déclaration et levée d'exception

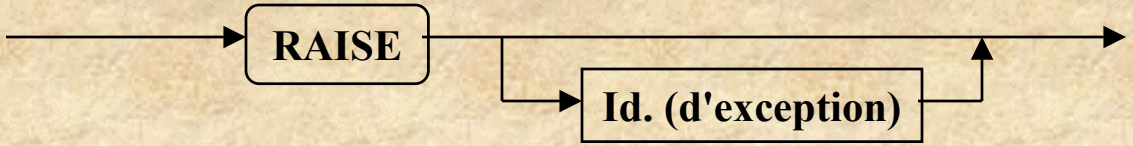
▪ Pour être reconnue comme telle, une exception non prédéfinie doit avoir été préalablement déclarée selon le diagramme suivant :



Exemple :

DENOMINATEUR_NUL, DISCRIMINANT_NEGATIF : EXCEPTION ;

▪ Une exception peut être suscitée par un *ordre*, selon le diagramme suivant :



Exemple : **if** Delta < 0.0 **then raise** DISCRIMINENT_NEGATIF **end if**;

Remarque : L'*ordre* **raise** sans indication d'un identificateur d'exceptions sert uniquement à un preneur en mains d'exceptions.

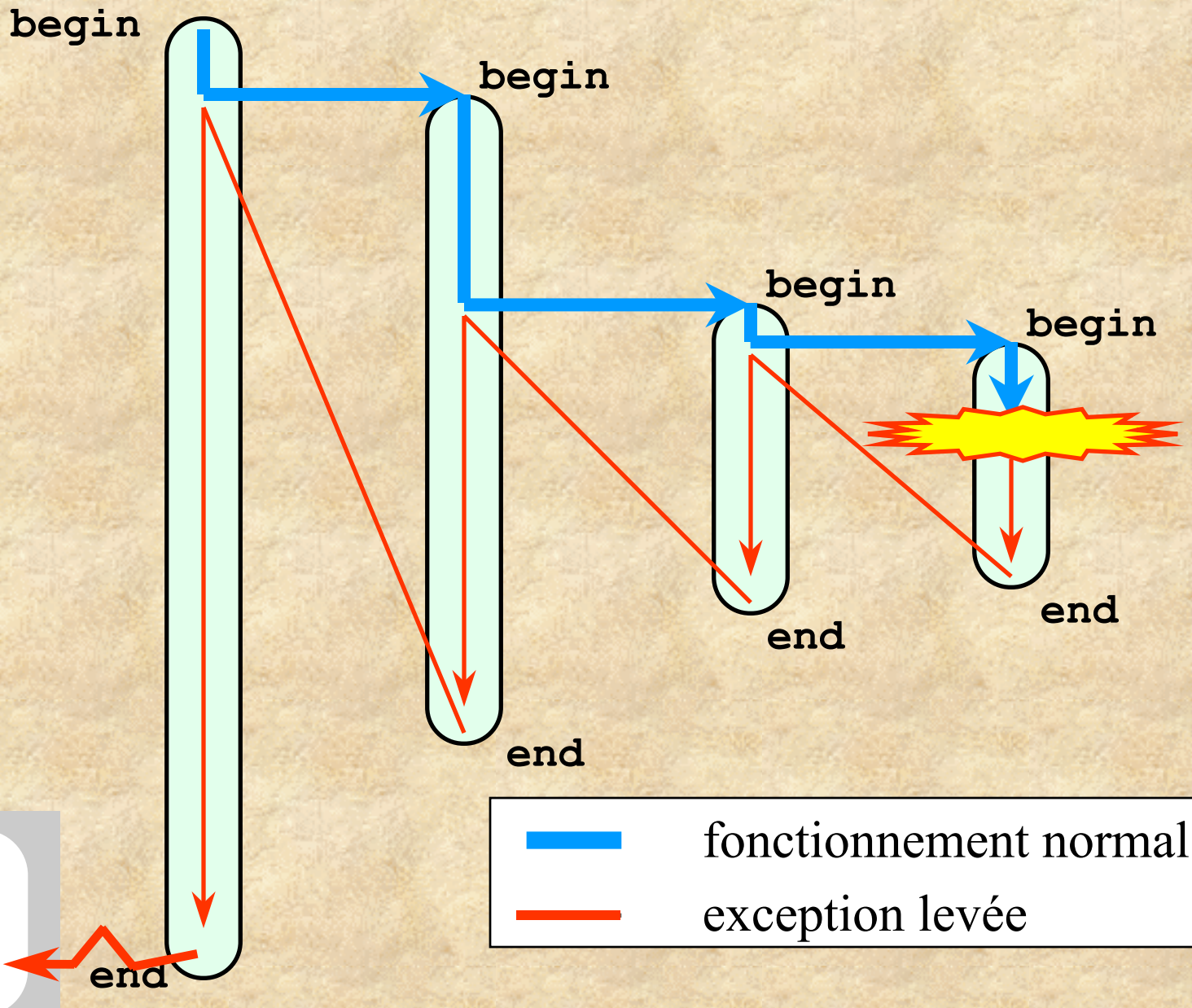
Bloc de traitement d'une exception

- **Le bloc de traitement d'exception est donné par le diagramme ci-dessous**

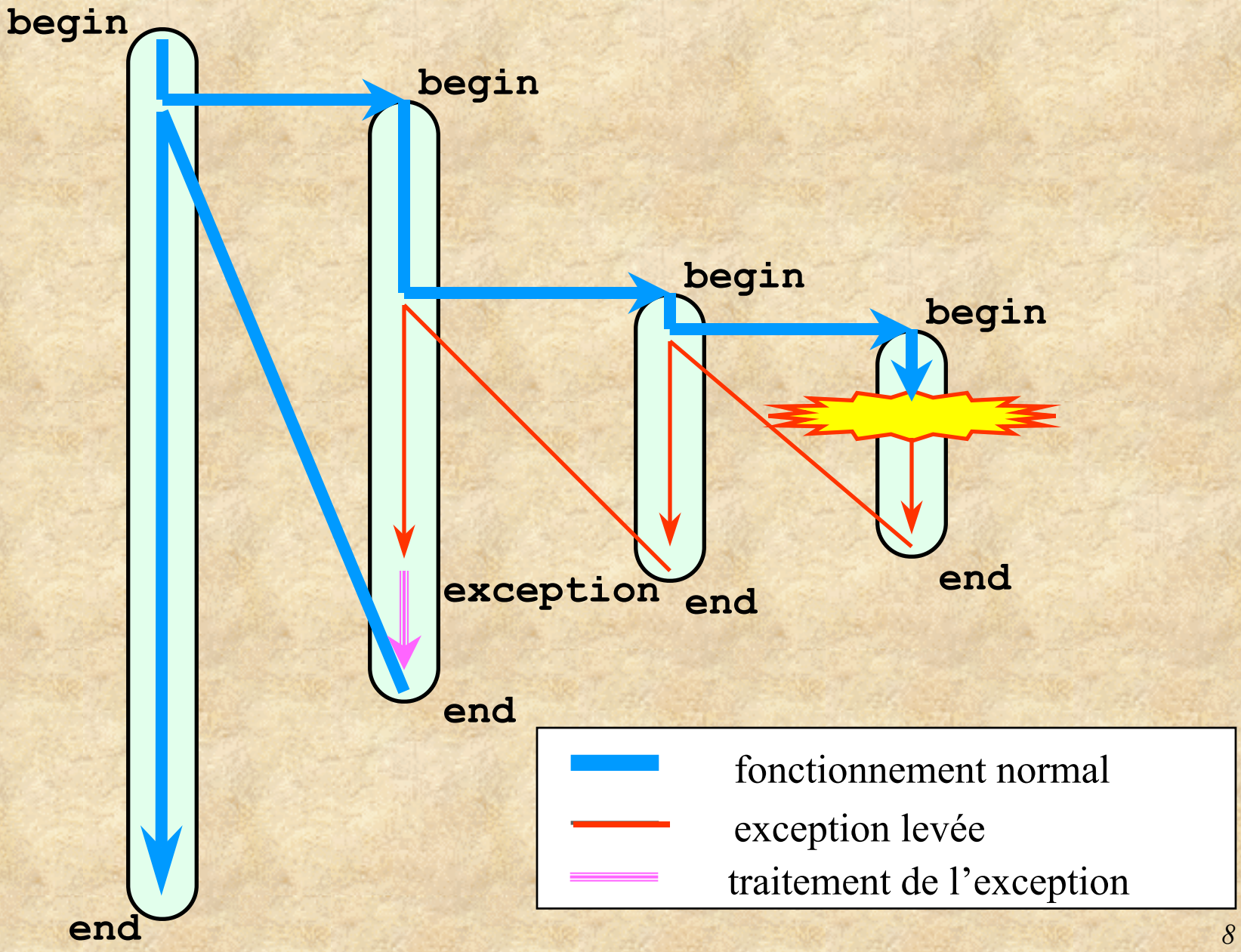
```
Discriminent_Negatif : Exception ;   -- declarer une nouvelle exception

exception -- sinon traitement des exceptions
        when Constraint_Error =>  Put_Line (" violation d'une contraine ");
        when Data_Error =>        Put_Line (" mauvaise type pour la saisie de Val ");
        when Discriminent_Negatif =>  Put_Line(" delta negatif ");
        when others =>  Put_Line("  erreur inconnue ");
```

Remarque : évitez **when others => null**; -- masquer l'erreur

Propagation d'une exception dans des blocs imbriqués



Propagation et traitement d'une exception dans un bloc



Propagation et traitement d'une exception

La capture peut se faire à tous les niveaux de propagation de l'exception,
mais toujours en *fin* d'un bloc d'exécution.

Il faut noter que l'endroit où l'on choisit de capturer une exception n'est pas indifférent :
on n'accède pas au même environnement à l'intérieur ou à l'extérieur d'une fonction,

Une exception donnée, une fois capturée, une exception est désactivée

Traitement d'une exception à l'interieur d'une procédure

- **Lorsqu'une exception est levée, on peut intercepter cette exception, et faire un traitement.**

```
procedure Calcul is
   Discriminent_Negatif : exception ;   -- declarer une nouvelle exception
   subtype Mon_Reel is Float range -6.0..6.0; -- pas de package d'entree/sortie, c'est un sous-type
   Val    : Mon_Reel ;
   Y, Delta : Float ;
begin  -- Calcul
          Get(Val);
          Skip_Line;
          Delta := Val;
          if  Delta < 0.0 then raise Discriminent_Negatif  end if;
          Y:= sqrt(Delta);
          exception -- sinon traitement des exceptions
                        when Constraint_Error => Skip_Line;
                                                 Put_Line (" Mauvaise domaine de la saisie de Val ");
                        when Data_Error => Skip_Line;
                                                 Put_Line (" Mauvaise type pour la saisie de Val ");
                        when Discriminent_Negatif =>  Put_Line(" delta negatif ");
end Calcul;
Remarque  : évitez when others => null;
```

Récupération d'une exception dans un sous programme et propagation vers le programme appelant

```
procedure Exemple is
    A : Positive;
    Erreur_Saisie : exception ;

    procedure Saisir (Val : out Positive) is
    begin -- Saisir
       Put ("entrer la valeur ");
       Get (Val); skip_line ;
       exception
          when Constraint_Error   =>
                Put_Line (" Débordement de type lors de la saisie de Val "); Skip_Line;
                raise Erreur_Saisie ;
          when Data_Error         =>
               Put_Line (" Mauvaise type pour la saisie de Val "); Skip_Line;
                raise Erreur_Saisie;
    end Saisir;

begin -- Exemple

   Saisir ( A ) ; -- appel de la procedure Saisir

   exception
       when Erreur_Saisie   => Put_Line (" Probleme dans la procedure de saisie des donnees ");

 end Exemple;
```


Exception non traitée : le programme continue avec des données erronées

Traitement d'une exception appliqué au contrôle de saisie

- Contrôle de la saisie au clavier d'une variable d'un type donnée. Si la saisie est :
  - ✓ un caractère, alors l'exception levée est de type Data_Error
  - ✓ un nombre négatif ou nul, alors l'exception levée est de type Constraint_Error
  - ✓ un entier positif, la saisie est correcte, on quitte cette procédure avec la bonne valeur lue de Val.

```
procedure Saisir (Val : out Positive)  is
begin
        loop
              begin
                   Get(Val);
                   Skip_Line;
                   exit;          -- ou return Val si c'est une fonction qui retourne la valeur lue Val
                   exception   -- sinon traitement des exceptions
                      when Constraint_Error => Skip_Line;
                                               Put_Line (" Débordement de type lors de la saisie de Val ");
                                               Put_Line("  Refaire la saisie… ");
                      when Data_Error => Skip_Line;
                                               Put_Line (" Mauvaise type pour la saisie de Val ");
                                               Put_Line("  Refaire la saisie… ");
               end;
         end loop;
end Saisir;
```

➡ Le sous programme fournit toujours une valeur entière positive

Application des exceptions à la gestion d'un menu

```
with Ada.Text_IO;              use Ada.Text_IO;              -- Module d'entree/sortie de texte
with Ada.Integer_Text_IO;      use Ada.Integer_Text_IO;      -- Module d'entree/sortie d'entiers

-- Procedure principale
procedure Gestion_Menu is

   -- Type T_Menu (type enumeration)
  type T_Menu is (Propagation, Selection, Insertion, Quitter);
  package Menu_IO is new enumeration_IO(T_Menu);
  use Menu_IO;

  -- Specifications des fonctions et des procedures utilisees
  -- ***************************************************************
  procedure Afficher_Menu;
  function  Menu return T_Menu;
```

Application des exceptions à la gestion d'un menu

```
-------------------------------------------------------------------------
-- Nom          : Afficher_Menu
-- But          : Afficher le menu ci-dessous
-- Parametres   :
-- Exception    :
-------------------------------------------------------------------------
procedure Afficher_Menu is
begin
        New_Line;
        Put_Line("    ******* Quelle operation desirez vous effectuer *******");
        New_Line;
        for Menu in T_Menu loop
                Put("       ");
                Put(Menu, T_Menu'Width);
                Put(" ----------------------------:  ");
               Put(T_Menu'Pos(Menu)+1, 0);
               New_Line;
        end loop;
        New_Line;
        Put_Line("    ****************************************************************");
        New_line;
end Afficher_Menu;
```

Application des exceptions à la gestion d'un menu

```
function Menu return T_Menu is
subtype T_Choix is Positive range T_Menu'Pos(T_Menu'First)+1..T_Menu'Pos(T_Menu'Last)+1;
Choix : T_Choix;
begin -- Menu
                Afficher_Menu;
                 Put(" Saisir votre Choix : ");
                 loop
                        begin
                                Get(Choix);
                                New_Line(2);
                                Skip_Line;
                                return T_Menu'Val(Choix-1);
                         exception
                                when Data_Error | Constraint_Error =>
                                Skip_line; New_line;
                                Put_Line(" Saisie incorrecte");
                                Put(" Veuillez entrez une nouvelle valeur: ");
                         end;
                 end loop;
 end Menu;
```