

Exception Handling

- **Exception handling is a language feature that allows the programmer to handle runtime "exceptional conditions."**
- **What is an "exceptional condition"?**
 - hardware error
 - failure in underlying software
 - any anomalous event
- **It need not be erroneous -- just something that requires special handling.**

Terminology

- An exception is *raised*, or *signalled*, when its associated condition occurs.
- The code that is executed after an exception is raised is called the *exception handler*. This code processes the exception.

Issues

- **Form of handler.**
 - Complete program unit, or code segment?
- **Location of handler.**
 - Are exceptions handled in unit where raised, in calling unit, or elsewhere?
- **Binding of handlers to exceptions.**
 - Static or dynamic?
- **Transfer of control after exception is handled.**
 - Allow unit that raised exception to continue executing?

Issues (continued)

- **Default exception handlers.**
 - Should they be provided?
- **Specification of user-defined exceptions.**
 - Form, location, scope.
- **Built-in exceptions.**
 - Can the user raise them explicitly?
- **Disabling of exceptions.**
 - Should it be allowed?

Exceptions in PL/I

- **Conditions = exceptions**
- **Built-in and user-defined**
- **Default handlers for built-in conditions, but can be overridden.**
- **Dynamic binding of handlers to exceptions**
- **Handlers are code segments, no parameters**
- **After handling exception, can send control anywhere. Default handlers go to *raise* of or cause.**

PL/I Example

```
declare condition bad_input;  
...  
on condition bad_input  
begin;  
...  
end;  
...  
  
read(x);  
if (x < 0) or (x > 10) then  
    signal condition bad_input;
```

Exceptions in CLU

- **More restricted than PL/I**
- **Static binding of handlers to exceptions**
- **Handlers are attached to statements**
- **Exceptions must be handled by calling routine**
- **Unit raising exception is terminated; control transfers to statement following that with handler**
- **No disabling of exceptions**
- **Handlers can have parameters**
- **Exception failure raised if an exception has no handler**

CLU Example

```
begin
  x := f(y);
  z := g(h);
end
  except when bad_input(c):
    . . .
  end
```

```
f = proc (<formals>)
  signals(bad_input(char))
begin
  . . .
  signal(bad_input(. . .))
  . . .
```


Exceptions in Ada

- **Less restrictive than CLU, more controlled than PL/I**
- **Static binding of handlers to exceptions, but if no local handler, exception is propagated back call chain**
- **Handlers have no parameters**
- **Block that raises exception terminates, but enclosing block may continue execution.**
- **Disabling of exceptions possible**

Ada — Error Recovery

```
procedure Sort (X: in out ELEM_ARRAY ) is
    Copy: ELEM_ARRAY ;
begin
    -- Take a copy of the array to be sorted.
    for i in ELEM_ARRAY'RANGE loop
        Copy (i) := X (i) ;
    end loop ;
    -- Code here to sort the array X in ascending order
    -- Now test that the array is actually sorted
    for i in ELEM_ARRAY'FIRST..ELEM_ARRAY'LAST-1 loop
        if X (i) > X (i + 1) then
            -- a problem has been detected - raise exception
            raise Sort_error ;
        end if ;
    end loop ;
exception
    -- restore state and indicate to calling procedure
    -- that a problem has arisen
    when Sort_error =>
        for i in ELEM_ARRAY'RANGE loop
            X (i) := Copy (i) ;
        end loop ;
    raise ;
    -- unexpected exception. Restore state and indicate
    -- that the sort has failed
    when Others =>
        for i in ELEM_ARRAY'RANGE loop
            X (i) := Copy (i) ;
        end loop ;
        raise Sort_error;
end Sort ;
```

Summary

- **Trade-offs between power, flexibility (PL/I) and safety (CLU).**
 - Ada provides a compromise.
- **But is exception handling really necessary?**
 - Arguments both ways (Black, "Exception Handling: The Case Against")

Handling Exceptions without Exception Handling

- **Two approaches:**
 - **Pass a "status variable."**
 - **Pass a subroutine to be called under certain conditions.**
- **In both cases, the exception handling is provided by the caller.**
- **To handle an exception locally, simply insert appropriate code.**