

Multidimensional Arrays as Parameters

● In C/C++

```
void fun (int matrix [][][10]) {  
    // ...  
}  
  
int main () {  
    int mat [5][10];  
    ...  
    fun (mat);  
    ...  
}
```

1

Multidimensional Arrays as Parameters

● In Ada

```
type MTYPE is array (INTEGER range <>  
                    INTEGER range <>)  
                    of FLOAT;  
  
MAT : MTYPE (1..100, 1..20);  
  
function FOO (M : in MTYPE)  
    return FLOAT is  
  
    ...  
  
    for ROW in M'range(1) loop  
  
        ...
```

2

Implementation of Subprograms

● Activation record

- local variables
- parameters
- return value
- dynamic link
- static link
- return address

3

FORTRAN Activation Record

- **Allocated statically**
- **Needs only**
 - local variables
 - parameters
 - return address

4

Static and Dynamic Links

- **Static Link**
 - activation record of enclosing scope
 - needed to access non-local variables
- **Dynamic Link**
 - activation record of caller
 - needed to pop the activation record
 - needed for finding exception handler

5

Displays

- **Alternative (old) implementation for accessing static information**
- **Instead of static link**
- **Maintained outside stack**
- **Contains pointers to activation records of all active nesting levels**
- **Old values are spilled onto stack**
- **No traversal of static links**

6

Implementation of Functional Languages

- **Activation record must be on heap**
 - if function returns a local function
 - if a function escapes its scope
- **Optimized implementations**
 - on heap if absolutely necessary
 - in registers if possible
 - otherwise on stack
- **Garbage collection**

7

Implementation on RISC Processors

- **Activation record in registers**
- **On SPARC: 6 parameters in registers**
- **On MIPS: 4 parameters in registers**
- **If registers are exhausted, spill an old activation record onto stack**
- **On SPARC: 8 register windows, spilling needed after 8 calls**
- **Inefficient for recursion**

8

Exception Handling

- **Alternative mechanism for returning from function**

- **In C++**

```
int main () {
    try { foo (42); }
    catch (int x) { ... }
}
int foo (int i) throw (int) {
    throw 17;
}
```

9

Exception Handling in Java

```
class Ex extends Exception { ... }  
void Main (String[] args) {  
    try { foo(42); }  
    catch (Ex e) { ... }  
    finally { ... }  
}  
int foo (int i) throws Ex {  
    if (somethingTerribleHappens())  
        throw new Ex();  
}
```

10

Exception Handling

- Handler searched for along call chain
- Jumps out across multiple functions
- May require destructor calls in intermediate scopes
- Conceptually
 - Return address is continuation param.
 - Exception is error continuation
 - Continuation is function to compute the rest of the program

11
