BITS Pilani

Lecture 16
Procedure Activations

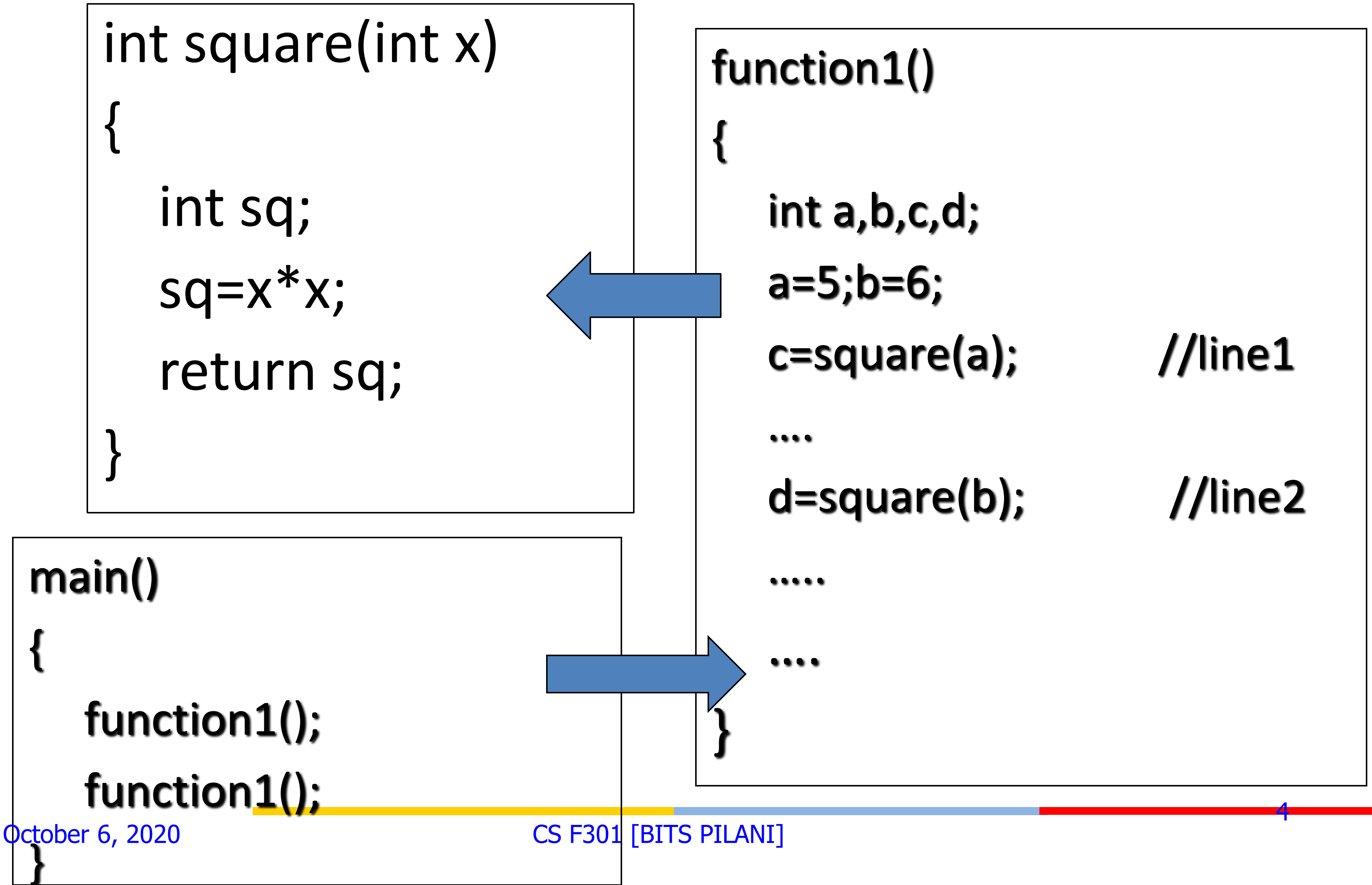
Elements of a procedure or subprogram

- Name for the procedure
- Body of code describing declarations and statements
- Formal parameters
- Result type

Function definition and call

- A **function definition** construct has a specific name and formal parameters to communicate with the calling procedure.
- A **function call** is a construct that uses name of existing procedures/ functions through the actual parameters.

Understanding flow of execution control: An example of procedure calls



```
int square(int x)
{
    int sq;
    sq=x*x;
    return sq;
}
```

```
function1()
{
    int a,b,c,d;
    a=5;b=6;
    c=square(a);        //line1
    ....
    d=square(b);        //line2
    .....
    ....
}
```

```
main()
{
    function1();
    function1();
}
```

Flow of Execution control

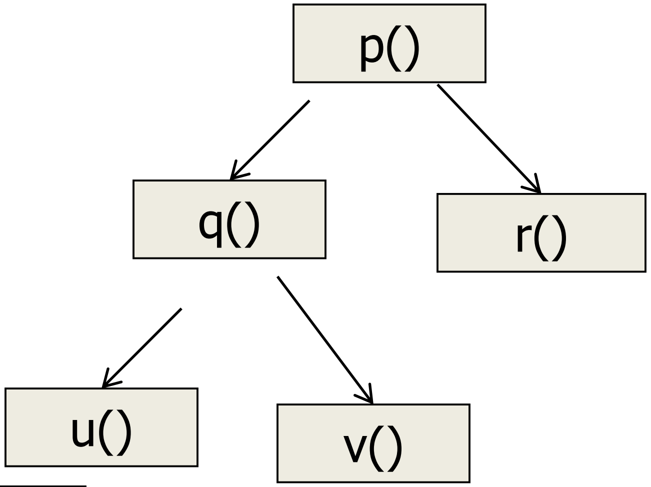
```
p()
{
        statements of p;
        q( );
        r( );
        remaining statements of p
}
```

- Procedure calls are nested in time
- If procedure p calls procedure q and procedure r (refer example)
  - The execution of q starts, the execution of p suspends
  - The execution of r starts only after execution of q is over
  - The execution of p resumes when execution of r is over

Activation tree

- The activations of procedures during execution of the entire program is represented by a tree, called an activation tree.

```
p()
{
        statements of p;
        q( );
        r( );
        remaining statements of p
}
q()
{
                …
                u( );
                v( );
                …
}
```



Sequence of procedure calls corresponds to the preorder traversal of the activation tree

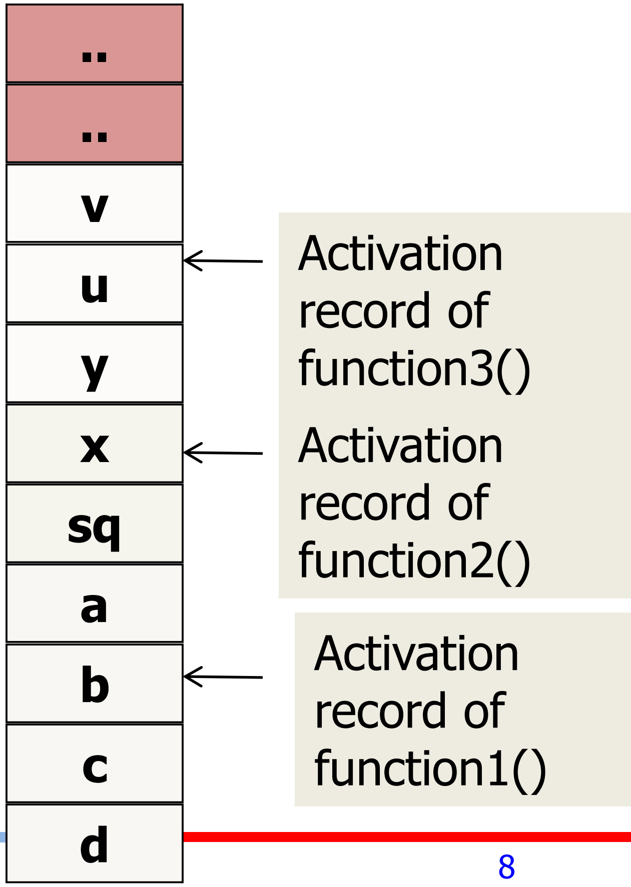
Procedure execution is in LIFO order

Run time Stack

- Each live activation of a procedure call is maintained by a data structure called activation record
- When a procedure is called, its activation record is pushed onto the stack

Activation Records

1. Each time a procedure or function is called, space for its local variables is pushed onto a stack
2. When the procedure terminates, the space is popped off the stack.
3. Non overlapping functions may share the stack space.
4. function calls keep the stack growing
5. Execution of the functions keeps the stack shrinking


..
..
v
u
y
x
sq
a
b
c
d
Activation record of function3()
Activation record of function2()
Activation record of function1()

Activation Records

Why Stack area?

1. Space is allocated at **stack** area of the memory at each function call
2. Space includes not just the local variables, it is needed for keeping the return values, function parameters, control link etc.
3. Size of activation record is fixed corresponding to one function call

Actual parameters
Returned values
Control links
Access link
Saved machine status
Local data
temporaries

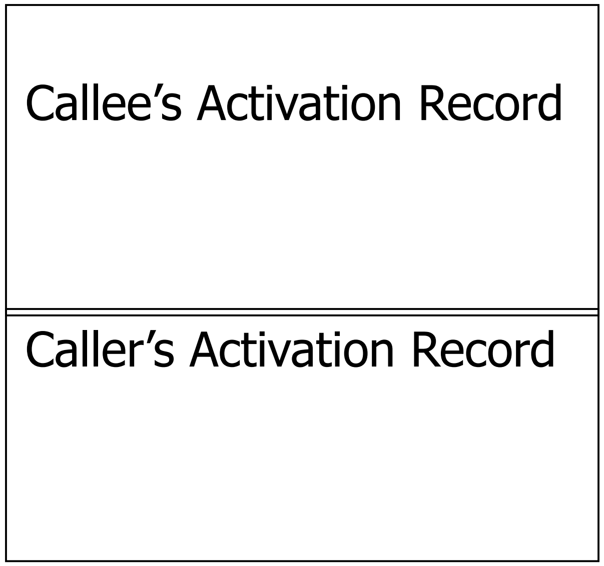
Calling Sequences

- Compiler code that allocates an activation record on the stack
- The calling sequences enter the fields of the activation records when a function is called

Communication of values between the caller and the callee

- Formal Parameters (part of callee's activation record)
- Actual parameters (part of caller's activation record)
- Return value (part of callee's activation record)

- **Above values are placed in the beginning of the callee's Activation Records**


Callee's Activation Record
Caller's Activation Record