Automatic Memory Allocation and Activation Record

Different Memory Allocation Space

- A C/C++ program may use different areas of memory to allocate memory space to store its required data or program instructions (code).
- For a C or C++ program four memory segments (storages) might be used:
  - Code segment
    - The space used to store program instructions
  - Data segment
    - The space used to store global and static variables
    - This segment can be further classified into read-only area and read-write area.
  - Stack
    - Area of the memory that contains the program stack
    - A Last in First Out (LIFO) structure.
    - Automatic variables (local variables and function arguments) are stored in this area.
  - Heap
    - An area of the memory used for dynamic (runtime) allocation
    - Will be discussed later in details

Activation Record

- At this point of the course we only focus on three ways of memory allocation:
    1. Global declaration
    2. Local variables declared in the body of the functions
    3. Argument declaration (formal argument of functions).
- The last two ways are also known as **automatic** declaration, because variables of these forms will be allocated, when a function is active and will be de-allocated when the function dies:

```
const double PI = 3.1415;  // PI is a global constant - not automatic
double square (int x)      // x a function argument and automatic var.
{
    double result;         // result is a local and automatic variable
     result = x * x;
     return result;
}
```

- **There other ways of memory allocation in C. We will discuss them in future lectures**

Activation Record (AR)

- This is an important topic in this course, which is not covered in detail in textbooks.
  - Study the lecture notes and documents posted on D2L.
- AR is a graphical presentation of variables of an **active function**.
- What is an Active Function?
  - A function is considered active if:
    - its code is currently being executed.
    - Or, if it is waiting for a function call to return.

Activation Record (continued)

- In a C/C++ program, several function can be active at the same time:

```
int main(void) {
  int x = 40;
  // point 1
  fun1(x);
   // point 5
  fun2();
  return 0;
}
```

```
void fun1( )
{
 int x =  50;
 // point 2
 fun3();
// point 4
 ...
}
```

```
void fun2()
{
 int  y = 60

 // point 6
 ...
}
```

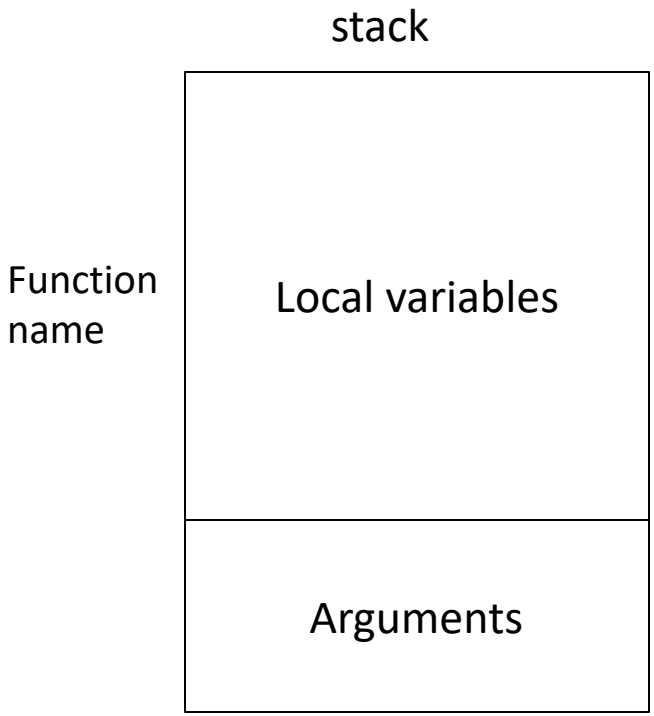
```
void fun3()
{
 int z = 70
// point 3
...

}
```

- At point 1: only main is active.
- At point 2: main and fun1 are active.
- At point 3: three functions are active: main, fun1, and fun3.
- At point 4: only two functions main and fun1 are active.
- At point 5: only main is active
- At point 6: two functions, main and fun2 are active

Activation Record

- A useful method to understand how functions work together is to draw an activation record diagram (AR diagrams).
- An AR diagram shows at a specific point in time which functions are active
- Activation record of a function is divided in two regions:
  - Upper region for function local variables
  - Lower region for function arguments.
- The activation records should be drawn on the stack memory space.
- each variable will be shown with a rectangle box and should have either a value or a question-mark (?), if its value is unknown.
- The activation records (ARs) should be drawn in the order that they appear from bottom to top. \main should be at the bottom of the stack, and the AR for the currently executing function should be at the top.
- Global declarations and other types of memory allocation will appear on separates area of memory that will be discussed later.

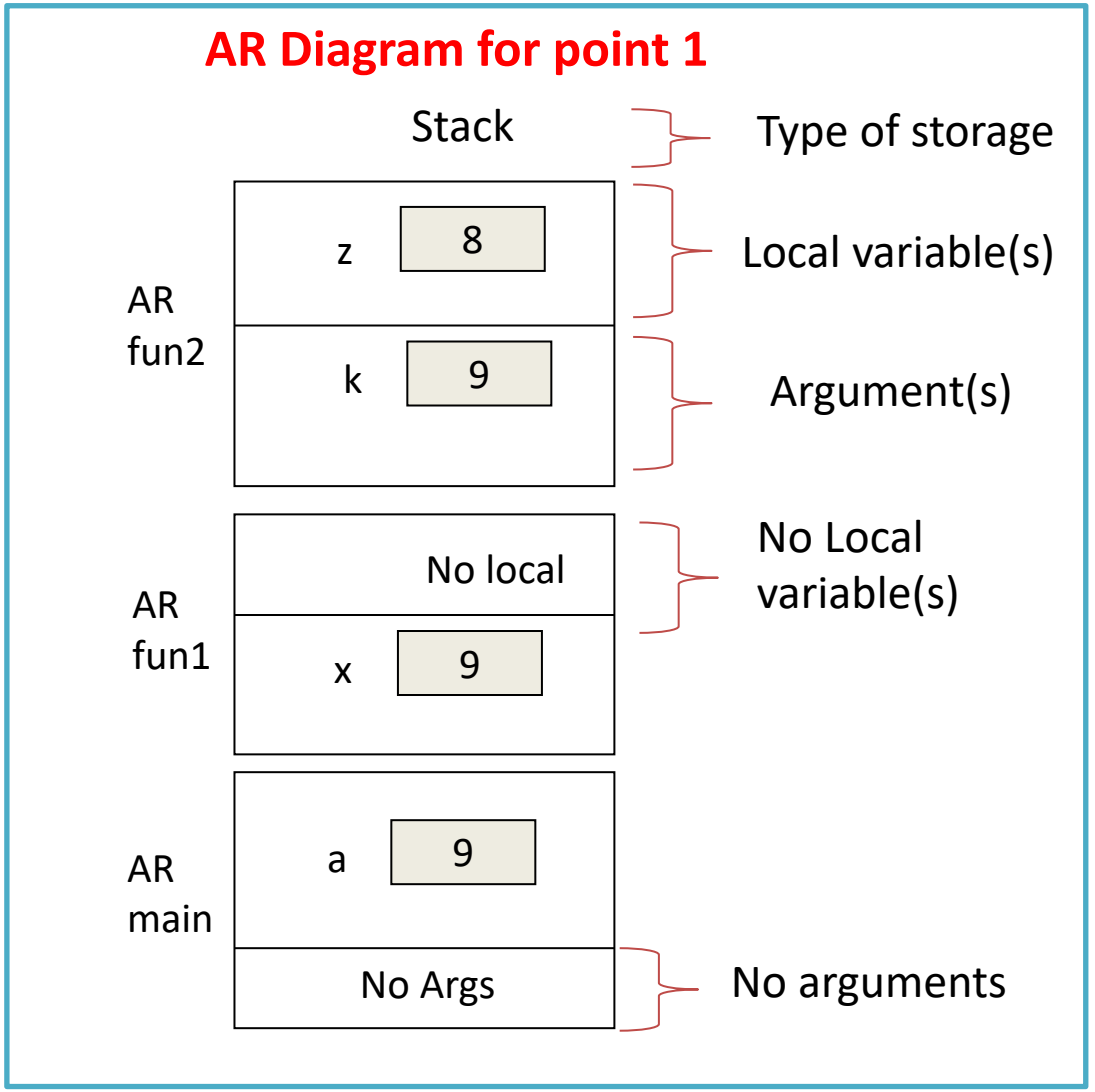


AR Conventions

```
int main()
{
    int a = 9;
    fun1(a);
    // point 2
    return 0;
}
```

```
Void fun1(int x )
{
    fun2 (x);
}
```

```
Void fun2(int k)
{
     int z =8
    // point 1
}
```



Question: What functions are active at point 2?

Answer: only one function, which is -- main