PRINCIPLES OF PROGRAMMING LANGUAGES

Variables - Scope & Lifetime

Prafullata Kiran Auradkar
Computer Science and Engineering

PRINCIPLES OF PROGRAMMING LANGUAGES

Variables - Scope & Lifetime

Prafullata Kiran Auradkar
Computer Science and Engineering

Storage Bindings and Lifetime

- Allocation and Deallocation
- Categories based on storage bindings and lifetime
  - Static
  - Stack-dynamic
  - Explicit Heap-dynamic
  - Implicit Heap-dynamic

Storage: Static Variables

- Efficient - as
  - Implements direct addressing
- Storage allocation is fixed
  - Adds in inflexibility
- No support for Recursive computations
- But brings history-sensitivity
- Static in a Class ! (Class with a State)

- Static in a Function ! (Function with State)

global variables (C,C++)
internal static variables (C, C++)
external static variables (C, C++)
static fields of a class (C++, Java)

Storage: Stack-Dynamic

- All local variables in C, C++, Java
- Allocation and Delallocation – on entry & exit to the block
- Indirect addressing – time consuming
- Support for Recursive computations

local variables (C,C++,Java)
Life time and scope – within block

Storage: Explicit Heap Dynamic variables

- Allocated and de-allocated by explicit directives,
  - Malloc(), free()
  - New(), delete()
- take effect during execution
- Referenced only through pointers or references

- dynamic objects in C++ (via new and delete),
- all objects in Java

Storage: Implicit Heap Dynamic variables

- Allocation and de-allocation caused by assignment statements
- Flexibility is high
- Inefficient, because all **attributes** are dynamic
- Error detection at compile time ?

- all variables in APL; all strings and arrays in Perl and JavaScript
- Python and Java objects

Deallocation of Heap Variables

- Explicit – with free() or delete()
- Implicit – **Garbage collector**

Think about this!!!

- What would happen if free() is not called for heap dynamic variables in **C**?
- Why python objects need not be freed?
- Is it the same with C++ and Java Objects?

THANK YOU

Prafullata Kiran Auradkar
Computer Science and Engineering

prafullatak@pes.edu