CSI 33 LECTURE NOTES (Ojakian)

Topic 3: How C++ and Python differ: C++ Pointers, Memory, etc.

OUTLINE

(References: Ch 4.2, 8, 10)

1. Python and C++ differences on memory
2. C++ Pointers

1. Python versus C++ memory addresses

   (a) C++ and Python:
   variable (name we use) *associated to* memory address (location in computer) *which contains* data

   i. C++ access memory address: &

   ii. Python access memory address: `id` and see **namespace dictionary** with `locals()` or `globals()`

   iii. Can use "cstdint" library and line like following to get C++ address as int:
   `uintptr_t addr = reinterpret_cast<uintptr_t>(&j);`

   iv.
   PROBLEM 1.

   A. *Write a program in each language finding the memory address of variables and data at those addresses.*

   B. *Find the differences between the memory addresses (recall hexadecimal as necessary); change the types and see how that changes the differences.*

   C. *Given the differences, what happens if the data can't fit in C++? Experiment to find out. See table on page 265 of textbook.*

   (b) Standard variable assignment:

   i. C++: Right side data is put into the left side memory location.
   Issue: Can go out of range (because fixed amount of memory set aside for it)

   ii. Python: Left side variable is associated to new memory location which contains right side data (i.e. namespace dictionary has its value updated to reference the right side)
   Note: right side either:

   A. constructs new object, or

   B. already constructed, so left side assigned same memory address as right side

   iii.
   PROBLEM 2. *Modify the last programs (both languages) to make some assignments and see what happens to the memory addresses.*
   Observe what happens in Python to its namespace dictionary

   (c) Variables

  i. C++: Variable associated to the same fixed memory address throughout its lifetime (except “pointer variables”)
  ii. Python: Variable can be associated to different addresses during their lifetime
  iii. **PROBLEM 3.** *Look at the last program, and note the constant C++ addresses and the changing Python ones.*

2. <u>Arrays</u>

   (a) Used as an underlying data structure in Python and C++

   (b) Array (one useful definition): A collection of objects of the same size stored in a continguous manner in the memory of the computer.

   (c) Underlying access to an array:

      i. You have its: first address (also called: foundation address, or base address)
      ii. You know how large every item in the array is.
      iii. Thus you can access an array item from its *key* in a *Random Access* manner.
      iv. Random Access (from Wikipedia): “*is the ability to access an arbitrary element of a sequence in equal time or any datum from a population of addressable elements roughly as easily and efficiently as any other, no matter how many elements may be in the set.*”
      v. Contrast Random Access to Linear Search

      PROBLEM 4. *Play with Topic3 ArrayVectorList program.*

      PROBLEM 5. *Suppose a C++ array of integers has a first address of 2000 (in decimal). Suppose there are 50 items in the array. Answer the following questions:*

      A. How many bytes of memory are used by the array?
      B. What is the address of the first item in the array?
      C. What is the address of the following items in the array: 2nd item, 20th item, last item?
      D. The second item occupies which bytes in the memory?
      E. If a new item is added to the end of the array, which bytes of memory will it occupy? (this has a reasonable answer and a … more reasonable answer)

   (d) Python versus C++ use of arrays:

      i. C++: Each item in the array is a data item of some size (so need same type for each item)
      ii. Python: Each item in the array is a memory address for some object (so items can be any type)

3. C++ Pointers

   (a) Declare with * *in front of variable*

   (b) Pointer variable has data which is a memory address for the given type

   (c) Access data at pointer by * in front - called *dereferencing*

   (d) Two typical ways to use:

      i. Set to address of some data

      ii. Allocate new memory (using `new`)

   (e)

   PROBLEM 6. *Write a program with pointers, assigned to addresses, and using data pointed to.*

   (f) `new` and `delete`

   PROBLEM 7. *See the TwoWays example program*

   Moral: With a pointer, either: 1) set its address to already declared ordinary variable, or 2) allocate space.

   ***PROBLEM* 8.** *Write a program that allocates an exponentially growing amount of memory with new statements, without any delete statements, and see what happens ...*

4. Dynamic Arrays

   (a) Declare pointer to individual type (ex: for array of ints, declare: int *A)

   (b) To allocate space for array use new with array size (ex: A = new int[5])

   PROBLEM 9. *See the dynamic array program.*

5. C++ Functions - pointers and pass by reference

   (a) Three ways to pass arguments:

      i. By value

      ii. By reference

      iii. As a pointer

   (b) Use of `const` here (and elsewhere)

   (c)

   PROBLEM 10. *See the ThreeWays example program examples.*

6. C++ Classes - Destructors, Copy Constructor

   PROBLEM 11. *Examine class Simple.*

   PROBLEM 12. *Write a function that takes a class by value and by reference to see difference. Try using* `const`*.*