DYNAMIC MEMORY ALLOCATION-CALLOC & MALLOC

Asst. Prof. Anup Adhikari
Faculty of Information Technology
Gandaki University
BIT

DYNAMIC MEMORY ALLOCATION

- We often don't know at compile time how much data space a program will need:
  - e.g. may not know how large an array will be needed to hold input data
- Solution 1: Use arrays of maximum possible size that we would encounter
  - Very wasteful, as we require lots and lots of memory that we may never use
  - Total amount of memory required by the program may exceed acceptable limit
- Solution 2: Dynamic memory
- We can create arrays (or other data structures) "on the fly", asking for memory as we need it and releasing it for possible reuse when finished
- Pointers are used to keep track of dynamic memory

MEMORY ALLOCATION: MALLOC()

- Additional memory to create dynamic data structures must be requested from the system.
- The function malloc() allocates a requested number of bytes from the system to the program
- Returns a generic pointer to the allocated memory
- The program can cast this pointer to any desired data type

USING MALLOC()

- Must request a specific number of bytes from malloc()
- Use sizeof operator to get bytes required by data type
- Must cast returned pointer to correct type

```
int *xPtr; /* Allocate space for 100 int values
*/
xPtr = (int *) malloc(100*sizeof(int));
*xPtr = 3; /* Sets first int to 3 */
xPtr[99] = 2; /* Sets 100th int to 2 */
*(xPtr + 99) = 2; /* Same thing as above */
```

SIZEOF() OPERATOR

- Determines the size in bytes of a data type
- When an array name is specified as the argument, returns the number of bytes in the array
- Typically:
  - sizeof(int) = 4
  - sizeof(float) = 4
  - sizeof(char) = 1
  - sizeof(double) = 8
- Later, we will use sizeof() to determine the size of more complex data types-e.g. structures

THE ANATOMY OF A MALLOC() CALL

Pointer to desired type

No. of elements to be allocated

number of bytes per element

```
➤ xPtr = (int *) malloc( 100*sizeof(int) );
```

Cast the pointer returned by malloc to desired pointer type

Total Allocated Bytes

MALLOC() —— SEE INTO THE MEMORY

- xPtr = (int *) malloc(100 *sizeof(int));

- Fill the table if
  - xPtr[1] = 10
  - xPtr[0] = 300
  - xPtr[98] = 900
  - xPtr[99] = 800

Table 1

	Address	Value
xPtr	3000	7000
	.....	....
*xPtr, xPtr[0]	7000	....
*(xPtr + 1), xPtr[1]	7004	....
	....	...
*(xPtr + 98), xPtr[98]	7392	.....
*(xPtr + 99), xPtr[99]	7396	....

ANOTHER DYNAMIC MEMORY ALLOCATION–CALLOC

- xPtr = (int *) calloc(100, sizeof(int));

- What would be the difference?
  - Note: two parameters
  - Main difference between malloc() and calloc() is that calloc() initializes the allocated memory to all zeros.

DEALLOCATING MEMORY WITH FREE()

- When a dynamically allocated data structure is no longer needed, it can be given back to the system for reallocation, using the free() function
  - int *xPtr;
  - xPtr = (int *)malloc(100 * sizeof(int));
  - free(xPtr);
- After free(), the values pointed to by xPtr are no longer valid
  - Using xPtr without reinitializing may result in strange problems