LSN 3
C Concepts for OS Programming

ECT362 Operating Systems
LSN 3 – C Programming (Review)

• Numerical operations
  – Punctuators ( (), {}, )
  – Precedence and Association
  – Mixed mode expressions

• Data Types
  – Standard
    • Modifiers
  – Abstract data types (Classes)
  – Coercion
  – Data type casting

<table>
<thead>
<tr>
<th>Operators</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>( )</td>
<td>left to right</td>
</tr>
<tr>
<td>* / %</td>
<td>left to right</td>
</tr>
<tr>
<td>+ -</td>
<td>left to right</td>
</tr>
<tr>
<td>=</td>
<td>right to left</td>
</tr>
</tbody>
</table>
• The array name, when referenced, evaluates to the base memory address of the array

```cpp
int nums[2] = {5, 4};
```

- `cout << nums[0];` outputs `5`
- `cout << nums;` outputs `300`

• Remember that `char` arrays will contain an extra null character at the end of the string
LSN 3 – Pointers (Review)

Pointers Notation:

```c
char name[20];
char *name_ptr;

name_ptr = name;
name_ptr = "Bob";

cout << name_ptr → Bob
cout << *name_ptr → B
```
LSN 3 – Functions (Review)

- A function is called by using function’s name and passing data as arguments
  \[ \text{function-name} \ (\text{data passed to function}); \]
  - This identifies the called function
  - This passes data to the function

- Pass by value
  - A function receives only a copy of the data passed in as the argument

- Pass by reference
  - A function can directly return only one value, although it can receive many values
LSN 3 – C-Style File I/O (Review)

- Include `<cstdio>` library
- File declaration
  
  ```
  FILE *file_variable;  // pointer to file
  ```
- Files must be opened prior to use and closed after use
  
  - `fopen` returns a pointer to file structure for desired file
    ```
    file_variable = fopen( "data.txt", mode);
    ```
  - `fclose`
    ```
    status = fclose( file_variable);
    ```
- String I/O
  
  - `fgets(line, size, file_variable)`
  - `fputs(line, file_variable)`
LSN 3 – C-Style File I/O (Review)

- Extracting individual elements from a string according to a legend

```c
sscanf( string, control_string, other_arguments );
```

- Control string
  - Letters, numbers, symbols, spaces, tabs, newlines, ...
  - Conversion characters:
    - `c` = character
    - `d, i` = decimal integer
    - `f` = floating point number
    - `s` = string

String of data to be examined
List of formatted data expected to be read
Addresses of storage Arguments for data read
LSN 3 – In Class Example

- Example program that:
  - Reads in contacts.txt formatted as:
    
    first_name tab last_name tab email_address tab phone_number
  - Stores the file values into individual arrays for processing
  - Contains a function for reading from user new contact information and adding it to the local arrays
  - Must have design for complete program written out
LSN 3 – Dynamic Memory Management

• **Malloc()**
  - Grabs a continuous portion of memory
    ```c
    void *malloc(size_t number_of_bytes)
    ```
  - Cast memory returned according to desired storage variable’s data type
    ```c
    char *name;
    name = (char *)malloc(100);
    ```

• **Calloc()**
  - Dynamically allocate memory that is already initialized to 0
    ```c
    void * calloc(size_t num_elem, size_t elem_size_in_bytes)
    ```
LSN 3 – Dynamic Memory Management

- **Use** `free()` **to make dynamically allocated memory available when it is no longer needed**
  - Requires a pointer to the memory being freed

- **new / delete**
  - Explicitly allocate / deallocate memory for a pointer
    ```c
    int *ptr = new float(3.14159);
    delete ptr;
    ```
  - Returns the address of a block of unallocated memory
  - Must include the new library (`#include <new>`) 
  - Can explicitly allocate memory for an array
    ```c
    int *array_ptr = new int[100];
    delete[] array_ptr;
    ```
LSN 3 – Standard C I/O

• **printf**

  ```
  int printf(const char *format, var1, var2, ...);
  ```

  Format identifiers are used to create output stream from specified variable list

• **scanf**

  ```
  int scanf(const char *format, arg1, arg2, ...);
  ```

  – Format identifier are used to define input stream
  – Arguments provide storage location of input values
LSN 3 – Data Structures

• Mechanism for storing data of varying types together under a single name
  – Uses the `struct` keyword to group multiple data elements together

```c
struct PersonalData {
    char name[100];
    int age;
    double height;
    double weight;
} myself;
.
.
.
PersonalData bob, tom, jim;
```

Data structure name

Data members of structure

Optional variable declaration of type `Date`
Accessing members of data structures

- Get and set the values of the members of a structure with member operator ‘.’

```c
structure_name.member_name
myself.age = 32;
```

Data structures and functions

- Whole structures can be passed as a parameter or returned one as a single value

```c
void new_function( PersonalData ); // prototype
```

Function is passed an instance of the PersonalData structure

The PersonalData structure must be accessible to all functions using it (global)
LSN 3 – Command Line Arguments

• Two arguments are used by main() to communicate with the operating system
  – `int argc`
    • Provides a count of the command line arguments
    • The value of `argc` is always 1 or more
  – `char *argv[]`
    • An array of strings
    • `argv[0]` always contains the name of the command itself
Traditionally the command line parameters are checked early in the program for completeness.
LSN 3 – Preparations for Next Lesson

• Prior to the next lesson create a program with the following attributes (you will use this program for the next assignment)
  – A program that automatically reads a text file of contacts in the format “first_name tab last_name tab email_address tab phone_number”.
  – Stores the values extracted from the file locally to an array of structures named ContactInfo
  – Contains a function for reading from user new contact information and adding it to the local array of structures
  – When program finishes it will append added information to contacts.txt
LSN 3 – Homework

• References
  – ECT361 notes
  – Help within MSVS for specific function syntax descriptions (MSDN Library)
    – http://cplusplus.about.com/od/beginnerctutorial/l/blcplustut.htm