The Calm before the Midterm

Comp-206 : Introduction to Software Systems
Lecture 12

Alexandre Denault
Computer Science
McGill University
Fall 2006
The midterm

- will be held next class, Thursday, in Macdonald Engineering Building 280
- starts at 16:05 sharp, so be there on time.
- will count towards 20% of your grade.
- has 15 questions.
- has four sections
  - Operating systems
  - The Shell and Shell Scripting
  - The Python Language
  - The C Programming Language
Assignment 2
- Out: October 24th
- Due: November 14th

Assignment 3
- Out: November 14th
- Due: December 5th
Lecture 12 is about making sure you are comfortable with the material.

We will do some review on last lecture.

Then, we will move on to practice questions for the midterm.
Structures are a data type composed of several other data types.
  - Think of it as a container, a variable that has variables inside it.

You can define new structures using the `struct` keyword.

```c
struct course {
    int number_of_students;
    char[100] name_professor;
    char[100] location_building;
    int location_room;
}
```
Coercion or Type-Casting

- **Coercion**: forcing one variable of one type to be another type.
- **Sometimes, type-casting is implicit**:
  - `int a = 2;`
  - `float b = a; // b = 2.0`
- **Most of the time, it's safer to specify it**:
  - `float a = 3.1415;`
  - `int b = (int)a; // b = 3`
- **When in doubt, type cast**:
  - `int a = 2;`
  - `float b = 3 / a; // b = 1.0`
  - `float c = 3 / (float)a; // c = 1.5`
Enumerated types: contain a list of constants that can be addressed in integer values.

- `enum days {monday, tuesday, wednesday, thursday, friday, saturday, sunday};`

As with arrays first enumerated name has index value 0.

- So monday has value 0, tuesday 1, ...

We can also override the 0 start value:

- `enum days {monday = 1, tuesday, wednesday, thursday, friday, saturday, sunday};`

Or simply assign different numerical values:

- `enum days {monday = 10, tuesday = 20, wednesday = 30, thursday = 40, friday = 50, saturday = 60, sunday = 0};`
What are pointers?

- A pointer is a variable which contains the address in memory of another variable.
  - Think of it as an integer variable that points to a block of memory.
- We can have a pointer to any variable type.
### Pointer operations, simplified

<table>
<thead>
<tr>
<th></th>
<th>content</th>
<th>address of</th>
</tr>
</thead>
<tbody>
<tr>
<td>int a</td>
<td>a</td>
<td>&amp;a</td>
</tr>
<tr>
<td>int *a</td>
<td>*a</td>
<td>a</td>
</tr>
</tbody>
</table>
Dynamic Memory Allocation

- The malloc() function allocates a block of memory and returns a pointer to that allocated memory.
  - `void *malloc(size_t size);`
- The size of the block must be specified.
- That block memory is not initialized.
  - It will contain whatever is currently in memory.
- Be careful not to access memory outside what you allocated.
  - Nothing will prevents you from accessing outside that block of memory.
Using the blocks of memory

- Both `malloc` and `calloc` return a void pointer (`void *`).
- In C, you use a `void*` when return a generic pointer.
- This generic block of memory must be cast before it can be used.
  ```c
  int *a = (int *) malloc( sizeof(int) * 40 );
  ```
- The `sizeof()` function simplifies the allocation of memory by calculating the size of the provided data type.
Deallocating Memory

- The free() function releases the specified memory space.
  - void free(void *ptr);
- The specified memory must have been returned by a previous call to malloc(), calloc() or realloc().
  - Otherwise, undefined behavior occurs.
- Not releasing memory after finishing with it can create memory leaks.
  - This can be an especially serious problem if you continually allocate memory.
On to the review ...