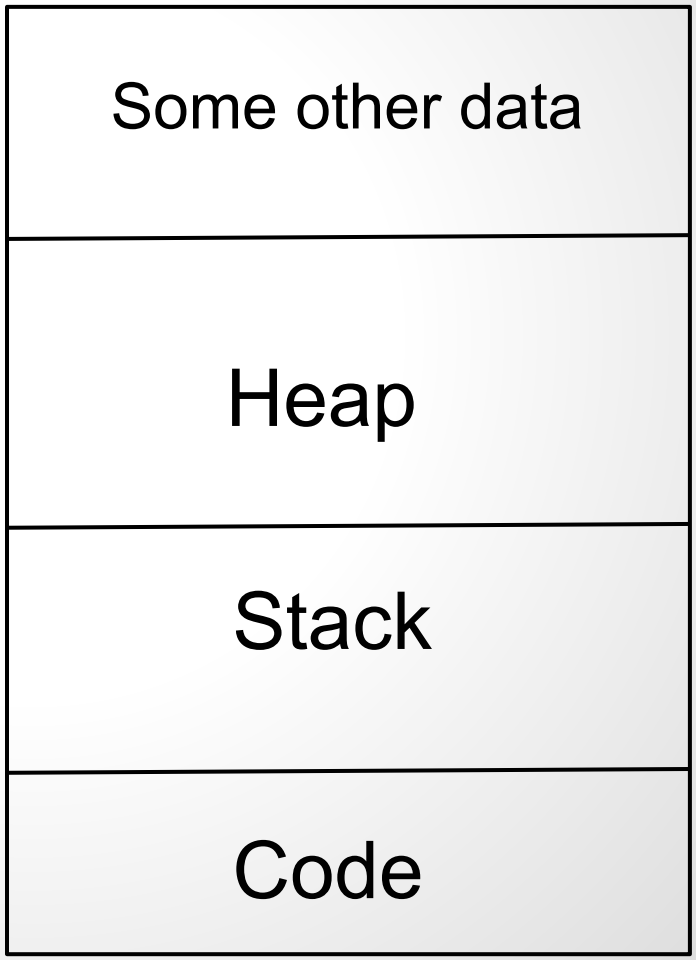
SENG 265-Lab 09

szehtabi@uvic.ca

Memory

- Stack vs Heap
- Dynamic allocation



Dynamic allocation

- If size unknown at compile time
- ptr = malloc (s)
  - s: number of bytes of heap memory to be allocated
  - Returns an address to the memory
  - void*: requires casting when assigned to a variable in stack (ptr in this case)

- Always check the return value:
  - If allocation fails, malloc() returns NULL

Dynamic allocation (cont.)

- #include <stdlib.h>
- sizeof()
  - Example: sizeof(int)
- realloc(ptr, size)
  - changes the size and copies from the original ptr
- free(ptr)
  - frees the allocated memory given the pointer
- Things to watch out for:
  - NULL pointers
  - Memory leaks

Structs

- Similar to class
- How to define it (before main function):

```
typedef struct mystruct{
    int myVar;
} mystruct_t;
```

- How to use it:
  - Very much like using any other type in C:

```
mystruct_t nstruct;
```

C reminder

- To compile:
  - gcc myprog.c -o myprog
- To run:
  - ./myprog
- To open a file:
  - File* fopen(char* fname, char* mode)
- To read from a file:
  - char* fgets(char* str, int max, File* f)
  - Returns NULL when reaches the end of file
- To tokenize a string:
  - char* strtok(char* str, char* delim)

Switching back from python

- #include<> instead of import
- Don't forget variable types
- Don't forget semicolons
- Don't forget curly brackets

Exercise

- Tokenize the 'data.txt' file and store it in an array of structs and then print out every other persons information.