

Brief Intro to C (I)

C Overview

```
/**
 * File: File Name followed by a short description of the program
 * Author: Your Name
 * Date: MM/DD/YYYY
 */

#include <stdio.h>
#include <stdlib.h>

/* A short description of the function, if necessary
 */
int main (int args, char *argv[]) {

    int courseNum = 333;

    printf("Welcome to CS %d! \n", courseNum);

    // Integer types
    char    c1;
    unsigned char c2;
    short int    integer1;
    int          integer2;
    long int     integer3;

    // Float types
    float  f1;
    double f2;

    // Pointer types
    unsigned char *ptr;
    ptr = (unsigned char *)&c1; // the symbol & means "address of"
    printf("The pointer points to an address with %u in it.\n", *ptr);

    // malloc: initialize pointers with memory
    ptr = (unsigned char *) malloc(sizeof(unsigned char));
    ptr = (unsigned char *) malloc(sizeof(*ptr));

    // free: return memory
    free(ptr);

    return 0;
}
```

- Two types of comments:
 - Multiple lines: `/* ... */`
 - Single line: `//`
- Execution begins at the start of the main function, which takes two arguments: `int argc, char *argv[]`. Return type can be `int`, used for error control.
- C is a typed language. To declare a variable, you need to specify its type. Basic types include `char`, `int`, `float`, `double`
- Decoration before types: `short`, `long` (set the size of a type); `unsigned` (set the range of a type)
- Pointer is a special type of C. It stores the address of a variable. You can declare any type of pointers. E.g., to declare an unsigned char pointer, use `unsigned char *ptr`;
- To fetch the address of a variable of a variable, use `&`. (e.g., `&c1`) The symbol `&` means “address of”
- Type casting in C is to use `()` enclosing the type you want to cast to and place it in front of the variable you want to cast. `ptr = (unsigned char *)&c1`
- Explicit memory management: allocate (`malloc`) a block of memory from the heap, and deallocate (`free`) the piece of memory when you done with it. Otherwise, you program will have memory leak. *Q: Do you find a memory leak in this program?*
- To compile and run your C program:
 - use `gcc -o` to compile: `gcc overview.c -o test` It generates an executable named `test`
 - run the executable: `./test`
- `sizeof()`: return the size of a variable/type