SOPE – Quick review of some C concepts

See examples in Moodle

- **printf() + scanf()** =>
  - *format specifiers*
    - %d, %i – integer
    - %c – char
    - %f – float
    - %f – double
    - %s – string
    - %zu – size_t
    - … *many others*
  - scanf("%d", &intValue); *but* printf("%d", intValue);
  - scanf("%c", &charValue);
  - scanf("%s", stringValue);
    - => *before* scanf()
    - char[MAX_STRLEN+1] stringValue; // define MAX_STRLEN as a constant
    - ***or***
    - char *stringValue;
    - stringValue=(char*) malloc((MAX_STRLEN+1)*sizeof(char));
    - … ; // use stringValue
    - free(stringValue);

- **int scanf("%s", stringValue);**
  - reads up to the 1st space, tab or newline (non included in stringValue); the rest remains in the input buffer
- **int sscanf(const char *s, const char *format, ...);**
  - similar to scanf(); reads from string **s**
- **int sprintf(char *s, const char *format, ...);**
  - similar to printf(); writes to string **s**

- **char * fgets (char * str, int num, FILE * stream);**
  - It stops when either **num-1** characters are read, the newline character is read, or the end-of-file is reached, whichever comes first.
  - NOTES:
    - => allocate space for **str**
    - depending on the number of input characters, the newline character may be appended or not to **str** ☹
    - if more than **n-1** characters are typed, the exceeding characters remain in the input buffer
  - alternatives (*look for usage example at Open Group web pages*):
    - ssize_t **getdelim**(char **restrict lineptr, size_t *restrict n, int delimiter, FILE *restrict stream);
    - ssize_t **getline**(char **restrict lineptr, size_t *restrict n, FILE *restrict stream);
      - *note:* restrict *is a keyword (only in C) mainly used in pointer declarations that has to do with code optimization*
- **Arrays & Pointers**
  - int a[10];
  - int * aPtr; => aPtr = (int *) malloc(10*sizeof(int));
  - aPtr = a; is valid
    - a[2] ⇔ *(aPtr+2)

- **Arrays of strings** (3 alternative ways for allocating space)
  - char names[MAX_NAMES][MAX_LEN+1];
  - char *names[MAX_NAMES];
    - =>
    - for (i=0; i <MAX_NAMES; i++) names[i]=(char *) malloc(...);
    - ~~names[0]="Ana";~~ strcpy(names[0],"Ana");
    - How to free memory?
  - char **names:
    - =>
    - names = How to allocate memory?
    - for (i=0; i<MAX_NAMES; i++) names[i]=(char *) malloc(...);
    - How to free memory?

- **Pointers to functions**
  - void **func1**(int n)
    { … }

    void func2 (int x, void (***f**)(int))
    { …;
      f(x);
      …
    }
    …

  - // "playing" with pointers

    void (*funcPtr) (int); // what is funcPtr ?

    funcPtr = func1;
    funcPtr(10);
    …
    func2(25, func1);
    …