Topics

- Dynamic memory allocation
- Command line parameters
- More complex pointer usage
- Structures:
  - Syntax
  - Characteristics
  - Simple uses
  - Arrays of structures

Objectives

1) Be able to use dynamic memory allocation to support data structures of arbitrary size;
2) Be able to use command-line parameters in Unix systems;
3) Understand the syntax of structures and be able to use them in simple programs;
4) Be able to define self-referencing structures.
**cdecl tool**

**Commands:**
- declare <name> as <english>
- explain <gibberish>
- set or set options
- help, ?
- quit or exit

**english:**
- function [( <decl-list> )] returning <english>
- array [ <number> ] of <english>
  - [ const | volatile | noalias ] pointer to <english> <type>
- type:
  - [ <storage-class> | struct | union | enum ] <name>
- decllist: a comma separated list of <name>, <english> or <name> as <english>
- name: a C identifier
- gibberish: a C declaration, like 'int *x', or cast, like '(int *)x'
- storage-class: extern, static, auto, register
- C-type: int, char, float, double, or void
- modifier: short, long, signed, unsigned, const, volatile, or noalias

**Examples:**
- cdecl> declare p as pointer to int
  - int *p
- cdecl> explain int (*ipf[10])()
  - declare ipf as array 10 of pointer to function returning int

---

**Class problem 1**

- Design a C program to take information about part-time students enrolled in subject 31429 and print them on the screen. The information contains student number, name, and marks. Suppose that there are 2 students.

  (Solution is provided in two versions: using one-dimensional arrays and using structures. See the working code files stuRecord.c and stuRecord2.c, respectively.)
Arrays are collections of identical elements and array names are expressions which is evaluated to be the address of the first element in the array. Passing an array name to a function is passing an address to a function, call by reference is used.

Structures are collections of different things and structure name (struct tagName together) is same as any ordinary variable name. When a variable of a structure type is passed to a function, the function sees the copy of the variable value instead of the address of the variable. Call by value is used.

```c
#include <stdio.h>

int main() {
    struct student {
        int number;
        char name[20];
        double mark;
        char subject[20];
    };

    struct student stu1, stu2;
    printf("The address and size of stu1 is %p and %ld.\n", &stu1, sizeof(stu1));
    printf("The addresses of the individual elements are %p, %p, %p, and %p.\n", &stu1.number, stu1.name, &stu1.mark, stu1.subject);
    printf("The address and size of stu2 is %p and %ld.\n", &stu2, sizeof(stu2));
    printf("The addresses of the individual elements are %p, %p, %p, and %p.\n", &stu2.number, stu2.name, &stu2.mark, stu2.subject);
    return 0;
}
```
STORING STRUCTURES – Example 1 (cont.)

bash$ cc exStructStorage.c
bash$ a.out
The address and size of stu1 is ffbefab8 and 56.
The addresses of the individual elements are ffbefab8, ffbefabc, ffbefad0, and ffbefad8.
The address and size of stu2 is ffbefa80 and 56.
The addresses of the individual elements are ffbefa80, ffbefa84, ffbefa98, and ffbefaa0.
bash$
Note: The shadowed area is not used.

STORING STRUCTURES – Example 2 (exStructStorage2.c)

#include <stdio.h>
int main() {
    struct rational {
        int top;
        unsigned denom;
    };
    struct errortype {
        char item[25];
        double result;
        struct rational inval;
    } myErrorType;
    struct rational val,*ratbase;
    printf("The address and size of val is %p and %ld.\n",&val,sizeof(val));
    ratbase = (struct rational *) calloc(100,
        sizeof (struct rational));
    printf("The address and size of myErrorType is %p and %ld.\n", &myErrorType,sizeof(myErrorType));
    return 0;
}
STORING STRUCTURES – Example 2 (cont.)

bash$ cc exStructStorage2.c
bash$ a.out
The address and size of val is
    ffbefab8 and 8.
The address and size of myErrorType
    is ffbefac0 and 48.
The addresses of the individual
    elements are
    ffbefac0,
    ffbefae0, and
    ffbefae8.
bash$

Note: The shadowed area is not used.

Class problem 2

• Design a C program to take information about part-time students enrolled in three subjects and print them on the screen. The information contains student number, name, and marks (3 marks). Suppose that there are 2 students.

(Solution is provided in two versions. One uses call by value and the other uses call by reference. See working code file exStructFunc.c and exStructFunc1.c, respectively.)
**Class problem 2 – extension 1**

- Design a C program to take information about part-time students enrolled in three subjects and print them on the screen. The information contains student number, name, and marks (3 marks). Suppose that the number of student is determined at run time and is taken in from the keyboard.

  *(Dynamic memory allocation is involved. See working code file exStructFunc3.c)*

**Class problem 2 – extension 2**

- Design a C program to take information about part-time students enrolled in three subjects and print them on the screen. The information contains student number, name, and marks (3 marks). Suppose that the number of student is given when the program is run, i.e., at the command-line.

  *(Command-line parameters and dynamic memory allocation is involved. See working code file exStructFunc4.c)*