Advanced Programming

Structs, Unions, Pointers

Topics

- Structs
- Unions
- Pointers
  - Declaration
  - Operations
- Pointers and Arrays
- Dynamic Allocation

Structs

- Structs are user defined types where:
  - Has a name
  - May have fields with different types
  - Each field is referred to by its name

- struct name
```
{
      type1  field1;
      tyep2  field2;
}
```

Example

- Create a table where each row has the name, surname, ID , and phone number of a student.
- Read data into table
- Read a phone number and find the corresponding student

Unions

- User defined types with multiple fields
- Fields are accessed by name
- The difference with structs is that in unions the fields overlap

Example

- The list of employees in a company contains:
  - Name
  - Surname
  - Gender
  - Responsibility
  - If manager then office number
  - If engineer then project code

Pointers

- Pointers are variables to store address of a memory location.
- Each pointer has a type which shows the type of the location referred to by the pointer.
- Syntax
  - Type *pointerVariable;

Operation

- Operations on pointers
  - Assignment (address of a variable)
    - int v1, *pv1;
    - pv1 = &v1;
  - Access to the value of location
    - int v1, v2, *pv1;
    - v1 = 25;
    - pv1 = &v1;
    - v2 = *pv1 + 3;

Pointer Arithmetic

- The value of a pointer can be incremented to refer to the next location
- The value of a pointer can be decremented to refer to the previous location

Arrays and Pointers

- The name of an array is the pointer to the first location in the array
- The following expressions are equivalent
  - int A[10];
  - A[0] = 5; ➔ *A=5;
  - A[2] = 20; ➔ *(A+2)=20;

Example

- Read numbers into an array using pointers
- Read two strings, find the second one in the first string and replace its characters with ‘*’

Dynamic Memory Allocation

- Memory can be allocated during run time using malloc function
- Use free(void *) to return back the allocated memory
- void *malloc(int bytesRequested)

- Example
  - int *Loc;
  - Loc = (int *)malloc(4*sizeof(int));
  - (Loc+2) = 22; ➔ Loc[2]=22;

Pointer To Structs

- Pointers can be used with structs.
- To refer to the internal elements of a struct we use -> operator
- Example:
  - struct point

```
{ int x,y;} *p;
 p = (struct point *) malloc(sizeof (struct point));
 p->x=2;
 p->y=p->x;
```

Example

- Using arrays store polynomials
- Read two polynomials and add them into a third polynomial
- Solve the problem using dynamic allocation