CSC 215
Structures

Dr. Achraf El Allali

Basic Structure

- A Structure is a collection of related data items, possibly of different types.
- A structure type in C is called struct.
- A struct can be composed of data of different types.

Structures

- A Structure holds data that belongs together
- Examples:
  - Student record: student id, name, major, gender, ..
  - Bank account: account number, name, balance, ..
  - Date: year, month, day

Structures

- Individual components of a struct type are called members (or fields).
- Members can be of different types (simple, array or struct).
- Complex data structures can be formed by defining arrays of structs.

Struct basics

- Definition of a structure

```
struct <struct-type>{
  <type> <identifier_list>;
  <type> <identifier_list>;
  ...
  } ;
```

Each identifier defines a *member* of the structure

- Example

```
struct studentRec {
int student_idno;
char student_name[20];
int age;
 };
```

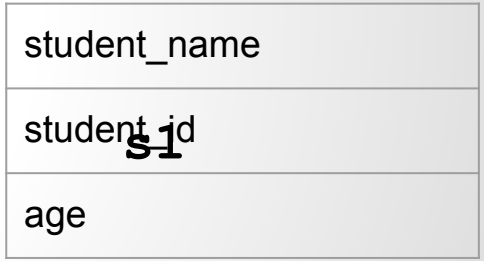
Struct basics

- Declaration of a variable of struct type:

```
struct <struct-type> <identifier_list>;
```

- Example:

```
struct studentRec s1, s2;
```



s1 and **s2** are variables of **StudentRec** type.

Struct basics

- Declaration of a variable of struct type:

```
struct studentRec {
    int student_idno;
    char student_name[20];
    int age;
 } s1, s2;
```

Struct basics

- A variable of a structure type can be also initialized by any the following methods:

```
struct date {
int day, month , year ;
} birth_date = {31 , 12 , 1988};
struct date newyear={1, 1};
struct date republic = {29 , 10 , 1922};
```

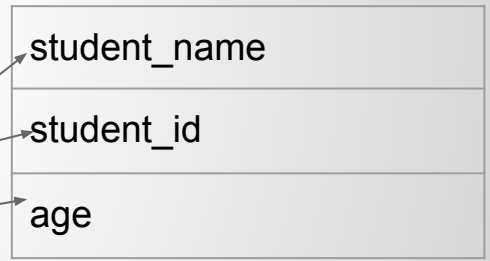
Struct basics

- The members of a struct type variable are accessed with the dot (.) operator

  <struct-variable>.<member_name>;

- Example:

```
strcpy(s1.student_name, "Mohamed Ali");
s1.studentid = 43321313;
s1.age = 20;
printf("The student name is %s", s1.student_name);
```



Struct basics

```
struct date American, revolution = {4, 7, 1776};
American = revolution;
```

- Assigns 4 to American.day, 7 to American. month, and 1776 to American.year

Declaring Structure Variables

```
struct s1 { char c ; int i ; } u ;
struct s2 { char c ; int i ;} v ;
struct s3 { char c; int i ; } w ;
struct s4 { char c; int i ; } x ;
struct s4 y ;
```

The types of u , v , w , and x are all different, but the types of x and y are the same.

Nested Structures

```
struct Client
{
    char name[21];
    char gender;
    int age;
    char address[21];
};
```

```
struct BankAccount
{
    char name[21];
    int accNum[20];
    double balance;
    struct Client aHolder;
    };
```

Nested Structures

- We can define the Client inside the BankAccount
- The Client is not visible outside the BankAccount which makes its name optional.

```
struct BankAccount{
        char name[21];
        int accNum[20];
        double balance;
        struct{
        char name[21];
        char gender;
        int age;
        char address[21];} aHolder;  };
```

Pointers to Structure

- Created the same way we create a pointer to any simple data type.

  struct date *cDatePtr, cDate;

- We can make **cDatePtr** point to **cDate** by:

  cDatePtr = &cDate

Pointers to Structure

- The pointer variable **cDatePtr** can now be used to access the member variables of Date using the dot operator as:

```
(*cDatePtr).year
(*cDatePtr).month
(*cDatePtr).day
```

- The parentheses are necessary because the precedence of the dot operator (.) is higher than that of the dereferencing operator(*).

Pointers to Structure

- Pointers are so commonly used with structures.
- C provides a special operator, **->** called the structure pointer or arrow operator, for accessing members of a structure variable pointed by a pointer.
- The general form for the use of the operator **->**

pointer-name -> member-name

cDatePtr-> year

cDatePtr-> month

cDatePtr-> day

Array of Structures

- Can create an array of sturctures

  struct studentRec studentRecords[500];

- studentRecords is an array containing 500 elements of the type studentRec.
- Member variable inside studentRecords can be accessed using the array subscript and dot operator.

  studentRecords[10].name = “Mohammed”;

Example

```
#include <stdio.h>
struct Employee {/* declare a global structure type */
     int idNum; double payRate; double hours; };
double calcNet(struct Employee *); /* function prototype */
int main() {
     struct Employee emp = {6787, 8.93, 40.5};
     double netPay;
     netPay = calcNet(&emp); /* pass an address*/
     printf("The net pay for employee %d is $%6.2f\n", emp.idNum, netPay);
     return 0; }
/* pt is a pointer to a structure of Employee type */
double calcNet(struct Employee *pt) {
     return(pt->payRate * pt->hours);
     }
```