CSC180: Lecture 27

Wael Aboulsaadat

wael@cs.toronto.edu

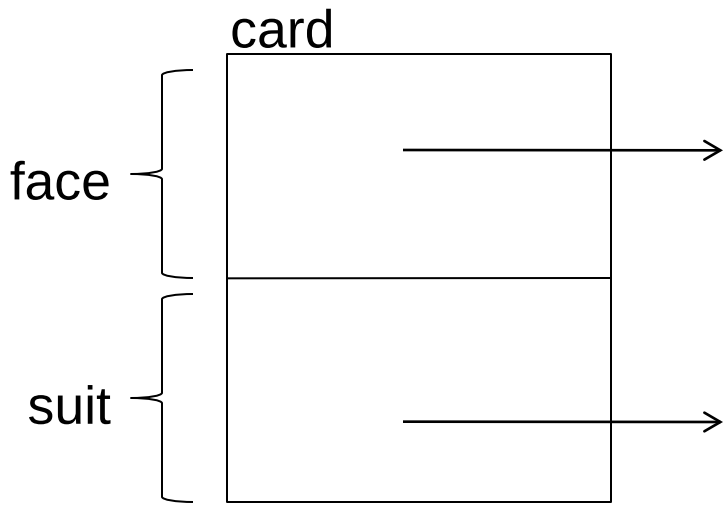
http://portal.utoronto.ca/

Acknowledgement: These slides are partially based on the slides supplied with Prof. Savitch book: Problem Solving with C

Structure

- Example

```
struct card {
   char *face;
   char *suit;
};
```



  - `card` is the structure name and is used to declare variables of the structure type
  - `card` contains two members of type `char *`
    - These members are `face` and `suit`

Structure Operations

- Valid Operations
  - Assigning a structure to a structure of the same type `=`
  - Taking the address of a structure: `&`
  - Accessing union members: `.`
  - Accessing union members using pointers: `->`
  - Using the `sizeof` operator to determine the size of a structure

```c
/*
   Using the structure member and
   structure pointer operators */
#include <stdio.h>

/* card structure definition */
struct card {
   char *face; /* define pointer face */
   char *suit; /* define pointer suit */
}; /* end structure card */

int main( void )
{
   struct card aCard; /* define one struct card variable */
   struct card *cardPtr; /* define a pointer to a struct card */

   /* place strings into aCard */
   aCard.face = "Ace";
   aCard.suit = "Spades";
```

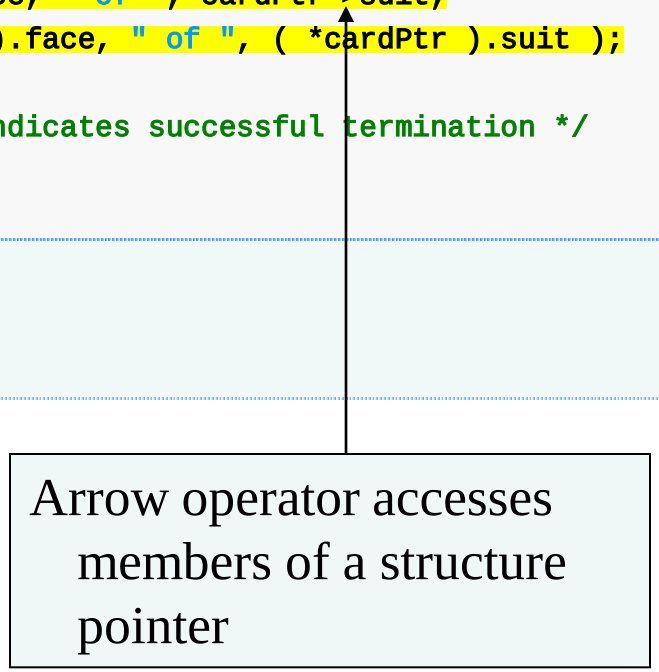
Structure definition

Structure definition must end with semicolon

Dot operator accesses members of a structure

```
20
21    cardPtr = &aCard; /* assign address of aCard to cardPtr */
22
23    printf( "%s%s%s\n%s%s%s\n%s%s%s\n", aCard.face, " of ", aCard.suit,
24       cardPtr->face, " of ", cardPtr->suit,
25       ( *cardPtr ).face, " of ", ( *cardPtr ).suit );
26
27    return 0; /* indicates successful termination */
28
29 } /* end main */
```

```
Ace of Spades
Ace of Spades
Ace of Spades
```



Arrow operator accesses members of a structure pointer

Structures and Duplicate Names

- Member variable names duplicated between structure types are not a problem.

```
struct FertilizerStock
{
   double quantity;
   double nitrogen_content;
} super_grow;
```

```
struct CropYield
{
   int quantity;
   double size;
} apples;
```

- super_grow.quantity and apples.quantity are different variables stored in different locations

Structures as Arguments

- Structures can be arguments in function calls
  - Parameter can be call-by-value or call-by-reference

- Example:
  void print_employee (struct Employee employee)
  - Uses the structure type Employee we saw earlier as the type for a call-by-reference parameter

Structures as Arguments

```
struct Employee
 {
        char* strName;
        char* strAddress;
        int      Salary;
        int      SIN;
  };
…….
void print_employee (struct Employee employee)
{
        printf(“ Employee Name: %s”, employee.strName );
        printf(“ Employee Address: %s”, employee.strAddress );
        printf(“ Employee Salary: %d”, employee.Salary );
        printf(“ Employee SIN: %d”, employee.SIN );
}
```

Structures as Arguments

```
void print_employee (struct Employee employee)
{
        printf(“ Employee Name: %s”, employee.strName );
        printf(“ Employee Address: %s”, employee.strAddress );
        printf(“ Employee Salary: %d”, employee.Salary );
        printf(“ Employee SIN: %d”, employee.SIN );
}

void print_all_employees(struct Employee arrEmployees[], int nSize )
{
    for( nCount =0; nCount < nSize; nCount++ )
        print_employee( arrEmployees[nCount] );
}
```

Structures as Return Types

- Structures can be the type of a value returned by a function

- E.g.

```
struct Employee
{
        char* strName;
        char* strAddress;
        int      Salary;
        int      SIN;
};
```

```
struct Employee create_new_employee(  )
{
    struct Employee employee;

    employee.strName   = “”;
    employee.strAddress= “”;
    employee.Salary = -1;
    employee.SIN = -1;

    return employee;
}
```

Hierarchical Structures

- Structures can contain member variables that are also structures

```
struct Date
{
   int month;
   int day;
   int year;
};
```

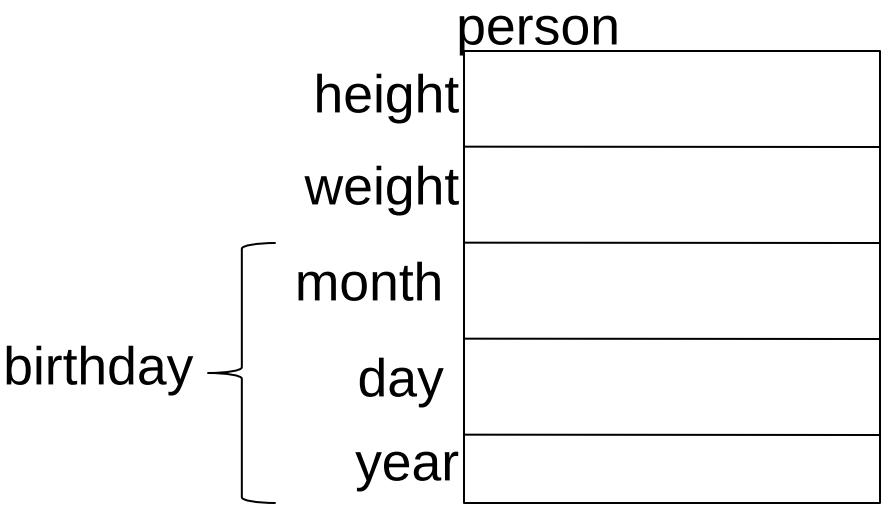
```
struct PersonInfo
{
    double height;
    int weight;
    struct Date birthday;
};
```

- struct PersonInfo contains a Date structure

Hierarchical Structures

```
struct Date
{
   int month;
   int day;
   int year;
};
```

```
struct PersonInfo
{
    double height;
    int weight;
    struct Date birthday;
} person;
```



Hierarchical Structures

```
struct Date
{
   int month;
   int day;
   int year;
};
```

```
struct PersonInfo
{
    double height;
    int weight;
    struct Date birthday;
};
```

```
void print_person_info( struct PersonInfo person )
{
    printf( " Height is %f \n", person.height);
    printf( " Weight is %d \n", person.weight);
    printf( " Birth Month is %d \n", person.birthday.month);
    printf( " Birth day is %d \n", person.birthday.day);
    printf( " Birth Year is %d \n", person.birthday.year);
}
```