College of computer science & mathematics

Dep. Of Computer Science

DATA STRUCTURE
هیاکل بیانات

Lecture 7 :
Struct & Recursion

Prepared & Presented by

Mohammed B. Omar

2023 -2024

Struct

Problem with Array

- **Although arrays greatly improved our ability to store data, there is one major drawback to their use ... each element (each box) in an array must be of the same data type.**

- **It is often desirable to group data of different types and work with that grouped data as one entity. We now have the power to accomplish this grouping with a new data type called a structure.**

- **Structure is a collection of variables of different data types under a single name. It is similar to a class in that, both holds a collection of data of different data types.**

Problem Without Using Structure

- **It is also possible to create our own data types.**
- **A user defined data type is called a structure, class etc.**
- **A structure can contain both built-in data types and another structure.**
- **The concept of structure is pretty much the same as arrays except that in an array, all the data is of the same types but in a structure, the data can be of different types.**

What a Structure is?

- **For Example: You want to store some information about a person: his/her name, citizenship number and salary. You can easily create different variables name, citNo, salary to store these information separately.**
- **However, in the future, you would want to store information about multiple persons. Now, we'd need to create different variables for each information per person: name1, citNo1, salary1, name2, citNo2, salary2**
- **We can easily visualize how big and messy the code would look(ضخامة وفوضى الكود). Also, since no relation between the variables (information) would exist, it's going to be a daunting task**

(لا توجد علاقة بين المتغيرات فستكون مهمة شاقة وصعبة).

- **A better approach will be to have a collection of all related information under a single name Person, and use it for every person. Now, the code looks much cleaner, readable and efficient as well.**
- **This collection of all related information under a single name Person is a structure.**

What a Structure is?

“A structure is a collection of variables under a single name. These variables can be of different types, and each has a name that is used to select it from the structure”

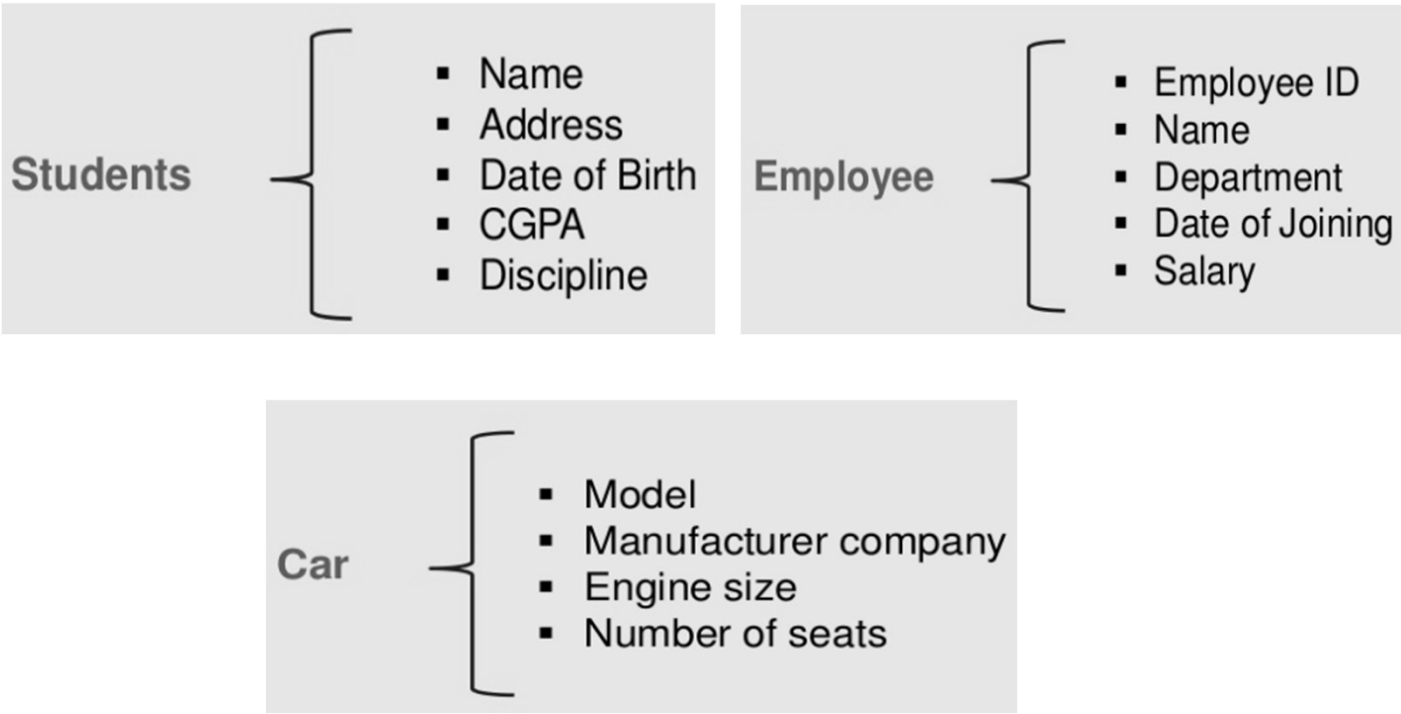
- There is always a requirement in most of our data processing applications that the relevant data should be grouped and handled as a group
- هناك دائمًا متطلب في معظم تطبيقات معالجة البيانات لدينا مفاده أن البيانات ذات الصلة يجب تجميعها ومعالجتها كمجموعة)

- In structure, we introduce a new data type

Some More Info

- A structure can contain any data type including **array** and **another structure** as well.
- Each variable declared inside structure is called **member of structure**.
- A structure may itself contain structures.
- A structure can be assigned to, as well as passed to, and returned from functions. (يمكن تعيين وظائف للهيكل، وكذلك تمرير ها إليه، وإعادته منها.)
- We declare a structure using the keyword **(struct).**

What a Structure is?



Steps to Create Structure

- Declare Structure
- Initialize Members of Structure
- Access Structure Elements

Declaration of a Structure

- Declare Structure
  - **struct** keyword is used for creating a structure.

- Structure Declaration Ways
  - By **struct** keyword
  - By declaring variable at the time of defining structure.

- طرق إعلان الهيكل
- باستخدام الكلمة الأساسية struct
- بإعلان المتغير في وقت تعريف الهيكل.

Declaration of a Structure

- The structure is declared by using the keyword **struct** followed by structure name, also called a tag. Then the structure members (variables) are defined with their type and variable names inside the open and close braces { and }.

- Finally, the closed braces end with a semicolon denoted as **;** following the statement. The above structure declaration is also called a Structure Specifier.

Declaration of a Structure

- Structures are syntactically declare with:
  - Keyword **struct**
  - Followed by the name of the structure
  - The data, contained in the structure, is defined in the curly braces(الاقواس المتعرجة)

- All the variables that we have been using can be part of structure

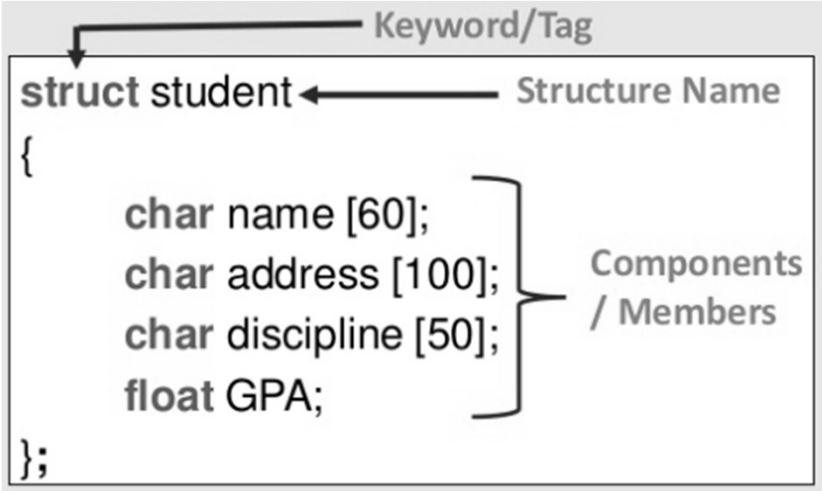
Declaration of a Structure

Syntax:

```
struct structure name
{
    member_type1   member_name1;
    member_type2   member_name2;
    member_type3   member_name3;
    .
    .
    .
};
```

approach 1(Declaration of a Structure)

```
struct student
  {
     char name [60];
     char address [100];
     char discipline [50];
     float GPA;
};
```



Declaration of a Structure

- **Student** is called the **structure tag**, and is your brand new data type, like int, double or char.
- **name, address, discipline,** and **GPA are structure members.**
- **Note**: Memory is not allocated at the time of its declaration. Memory is allocated when we declare structure variable.

شرح وتوضيح

عند تعريف بنية (Structure Declaration): لا يتم تخصيص أي مساحة في الذاكرة عند تعريف هيكل أو بنية.
أي أن مجرد تعريف نوع جديد من البيانات باستخدام struct لا يعني تخصيص ذاكرة مباشرةً لهذا النوع. تعريف البنية فقط يحدد "قالب" لشكل البيانات التي قد تستخدمه لاحقًا.

```
struct Person {
char name[50];
int age;
};
```

في هذا المثال، لم يتم تخصيص أي ذاكرة بعد. تم فقط إنشاء قالب Person الذي يحتوي على name و age.

عند تعريف متغير من نوع البنية (Structure Variable): يتم تخصيص الذاكرة عند تعريف متغير من نوع البنية. عند تعريف متغير بناءً على هذا النوع، تقوم اللغة بتخصيص مساحة في الذاكرة بناءً على الحقول المحددة في البنية.

```
struct Person person1;
```

في هذا السطر، تم الآن تخصيص الذاكرة في الذاكرة لـ person1 بحجم يناسب name و age المذكورين في البنية Person.

Approach 1(Declaration of a Structure)

```
struct student
{
        string name;
        string address;
        string discipline;
        float GPA;
};
```

```
struct car
{
        string model;
        string company;
        string engineSize;
        int price;
};
```

```
struct employee
{
        string employeeID;
        string name;
        string department;
        float salary;
};
```

Declaring Variables of Type struct

- The most efficient method of dealing with structure variables is to define the structure **globally**.

- This tells "the whole world", namely main and any functions in the program, that a new data type exists.

To declare a structure globally, place it BEFORE int main().

- The structure variables can then be defined locally in main, for example.

Declaring Variables of Type struct

```
. . .
struct STUDENT_TYPE
{
string name, street, city, state, zipcode;
int age;
double ID_num;
double grade;
} ;
int main()
{
// declare two variables of the new type
STUDENT_TYPE student1, student2;
. . .
}
```

Approach 2(Declaring Variables of Type struct)

```
struct STUDENT_TYPE
{
string name, street, city, state, zipcode;
int age;
double ID_num;
double grade;
}
student1, student2;
```

Accessing Structure Members

- To access any member of a structure, we use the member access operator (.).
- The member access operator is coded as a period between the structure variable name and the structure member that we wish to access.
- Remember we would use **struct** keyword to define variables of structure type.

Accessing Structure Members

- Suppose, you want to access age of structure variable student1 and assign it 50 to it. we can perform this task by using following code below:

```
student1.age = 50;
```

- Tasking input as:

```
cin >> student1.age;
```

Example 1 C++ Program to assign data to members of a structure variable and display it.

Solution-1/3

```
#include <iostream>
using namespace std;
struct Person
{
char name[50];
int age;
float salary;
 };
```

Declare a Structure of Person

Solution-2/3

```
int main()
{
   Person p1;
   cout << "Enter Full name: ";
   cin.get(p1.name, 50);
   cout << "Enter age: ";
   cin >> p1.age;
   cout << "Enter salary: ";
   cin >> p1.salary;
 //Displaying user Enter information
  cout << "\nDisplaying Information." << endl;
  cout << "Name: " << p1.name << endl;
  cout <<"Age: " << p1.age << endl;
  cout << "Salary: " << p1.salary;
  return 0;
}
```

Structure Variable Creating an object p1 of Person type like other Data type

Use Member Access Operator

Output of the Previous Program

```
Enter Full name: Adil Aslam
Enter age: 20
Enter salary: 1000$

Displaying Information.
Name: Adil Aslam
Age: 20
Salary: 1000$
```

Initializing Structures

Like normal variable structures can be initialized at the time of declaration. Initialization of structure is almost similar to initializing array. The structure object is followed by equal sign and the list of values enclosed in braces and each value is separated with comma.

يمكن تهيئة هياكل المتغيرات العادية في وقت الإعلان عنها. تتشابه تهيئة الهيكل تقريبًا مع تهيئة المصفوفة. يتبع كائن الهيكل علامة المساواة وقائمة القيم محاطة بأقواس، ويتم فصل كل قيمة بفاصلة.

Example

```
Person p1={"Adil Aslam", 20, 1000}
```

Initializing Structures

- **(1st Way)**

```
struct student
{
string name;
string address;
string discipline;
float GPA;
};
student std1 = {"ADIL", "PAK", "BSCS",3.5};
```

Initializing Structures

(2nd Way)

```
struct student
{
string name;
string address;
string discipline;
float GPA;
};
```

```
std1.name = “ADIL";
std1.discipline " PAK";
std1.salary = "BSCS";
std1.GPA = 3.5;
```

Example(Initializing Structures)

```
#include<iostream>
using namespace std;
struct Student
{
string Name;
string Address;
string Discipline;
float GPA;
};
int main()
{
Student S = {"Adil Aslam", "PAK", "BSCS", 3.5};
cout << "nStudent Name : " << S.Name;
cout << "nStudent Address : " << S.Address;
cout << "nStudent Discipline : " << S. Discipline;
cout << "nStudent GPA : " << S. GPA;
 }
```

Output of the Previous Program

```
Student Name : Adil Aslam
Student Address : PAK
Student Discipline : BSCS
Student GPA : 3.5
```

Structure Variable in Assignment Statement

S1 = S2;

- The statement assigns the value of each member of S2 to the corresponding member of S1.

- Note that one structure variable can be assigned to another only when they are of the same structure type, otherwise complier will give an error.

لاحظ أنه لا يمكن تعيين متغير هيكلي واحد إلى آخر إلا عندما يكونان من نفس نوع الهيكل، وإلا فسيعطي المترجم خطأ.

Limitation with Structures are:

- S1 + S 2;
- S1 - S 2;
- S1 * S 2;
- S1 / S 2;

- S1 = S 2;

Problem Statement • Write a program which declares two variables of a structure and copies the contents of first variable into the second variable.

ينسخ محتويات المتغير الأول في المتغير الثاني

```
#include <iostream>
using namespace std;
struct employee
  {
     int id;
     string name;
     double salary;
     string address;
};
```

```
employee emp1= {1, "Adil", 20000, "Pak_FSD"};
 employee emp2 = emp1;
 cout << "Employee Information:" << endl;
 cout << "-----------------------------------" << endl;
 cout << "ID:\t\t" << emp1.id << endl;
 cout << "Name:\t\t" << emp1.name << endl;
 cout << "Salary:\t\t" << emp1.salary<< endl;
 cout << "Address:\t" <<emp1.address << endl;
 cout<<" _____"<<endl;
```

```
   cout << "Employee 2:" << endl;
   cout << "-----------------------------" << endl;
   cout << "ID:\t\t" << emp2.id << endl;
   cout << "Name:\t\t" << emp2.name << endl;
   cout << "Salary:\t\t" << emp2.salary<< endl;
   cout << "Address:\t" << emp2.address << endl;
return 0;
}
```

Output of the Previous Program

```
Employee 1:

--------------------------------

ID:             1
Name:           Adil
Salary:         20000
Address:        Pak_FSD

_________________________
Employee 2:

--------------------------------

ID:             1
Name:           Adil
Salary:         20000
Address:        Pak_FSD
```

Your Task

Write a C++ Program to Store Information (name, Stage and marks) of a Student Using Structure and Display store information.

Recursion

Recursive Solutions

- Recursion **breaks a problem into smaller identical problems** – mirror images so to speak.
- By continuing to do this, eventually the new problem will be so small that its solution will be either obvious or known – this is known as the **base case** or **degenerate case**.
- Recursion is an alternative to iteration.
- Some recursive solutions are inefficient and impractical and iteration is better.

- إن الاستدعاء يقسم المشكلة إلى مشاكل أصغر متطابقة – صور معكوسة إذا جاز التعبير.
- ومن خلال الاستمرار في القيام بذلك، ستصبح المشكلة الجديدة في النهاية صغيرة جدًا بحيث يكون حلها واضحًا أو معروفًا – وهذا ما يُعرف بالحالة الأساسية أو الحالة المتدهورة.
- الاستدعاء هو بديل للتكرار.
- بعض الحلول الاستدعاء غير فعّالة وغير عملية والتكرار أفضل.

Recursive Solutions (Cont'd)

- Recursion sometimes provides elegantly simple solutions to very complex problems.
- A binary search is a good example of a problem solving approach that can be naturally accomplished using recursion.
- This is a divide and conquer approach.

- يوفر الاستدعاء أحيانًا حلولاً بسيطة وأنيقة لمشاكل معقدة للغاية.
- يعد البحث الثنائي مثالاً جيدًا لنهج حل المشكلات الذي يمكن إنجازه بشكل طبيعي باستخدام الاستدعاء.

Observations with Respect to a Recursive Method

1. One of the actions of the method is to invoke (call) itself.
2. Each call passes a smaller version of the same problem, i.e., the new problem is identical in nature but smaller is size.
3. The method must contain a test for the base case since that case is handled differently from the others. When you reach the base case the recursive calls stop.
4. The manner in which the size of the problem diminishes must ensure that the base case is reached.

- أحد إجراءات الطريقة هو استدعاء (استدعاء) نفسها.
- تمرّر كل عملية استدعاء نسخة أصغر من نفس المشكلة، أي أن المشكلة الجديدة متطابقة في طبيعتها ولكنها أصغر حجمًا.
- يجب أن تحتوي الطريقة على اختبار للحالة الأساسية نظرًا لأن هذه الحالة يتم التعامل معها بشكل مختلف عن الحالات الأخرى. عند الوصول إلى الحالة الأساسية، تتوقف الاستدعاءات المتكررة.
- يجب أن تضمن الطريقة التي يتم بها تقليل حجم المشكلة الوصول إلى الحالة الأساسية.

Recursion vs. Iteration

- Repetition
  - Iteration: **explicit loop** حلقة صريحة
  - Recursion: **repeated function calls** تستدعي وظيفة معينة متكررة
- Termination
  - Iteration: **loop condition fails**
  - Recursion: **base case recognized** التعرف على الحالة الاساسية
- Both can have infinite loops
- Storage
  - recursion use **stack** to give answer. Recursive programs typically **use a large amount of computer memory** and **the greater the recursion, the more memory used**.
  - A **recursively** written method can be simpler, but will usually run slower and use more storage than an equivalent **iterative version**
    - يمكن أن تكون الطريقة المكتوبة بشكل متكرر أبسط، ولكنها عادةً ما تعمل بشكل أبطأ وتستخدم مساحة تخزين أكبر من الإصدار التكراري المكافئ
- Balance
  - Choice between performance (iteration) and good software engineering (recursion)

    الاختيار بين الأداء (التكرار) والهندسة البرمجية الجيدة (للاستدعاء)

A Recursive Valued Method: The Factorial of n

- Problem
  - Compute the factorial of an integer n

- An iterative definition of factorial(n)

  factorial(n) = n * (n–1) * (n–2) * … * 1
  for any integer n > 0

  factorial(0) = 1

A Recursive Valued Method: The Factorial of n

- A recursive definition of factorial(n)

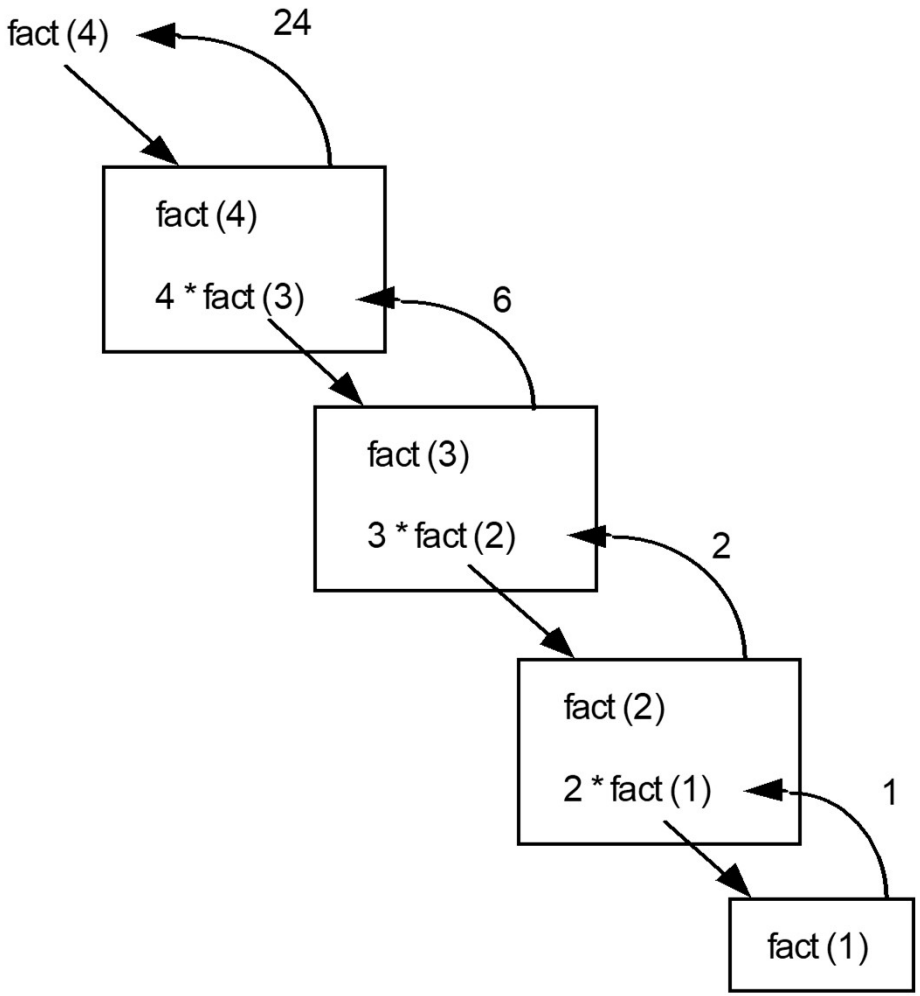
$$\text{factorial}(n) = \begin{cases} 1 & \text{if } n = 0 \\ n * \text{factorial}(n-1) & \text{if } n > 0 \end{cases}$$

- A recurrence relation
  - A mathematical formula that generates the terms in a sequence from previous terms

صيغة رياضية تولد المصطلحات في تسلسل من المصطلحات السابقة

  - Example

$$\text{factorial}(n) = n * [(n-1) * (n-2) * \ldots * 1]$$
$$= n * \text{factorial}(n-1)$$


fact (4)
24
fact (4)
4 * fact (3)
6
fact (3)
3 * fact (2)
2
fact (2)
2 * fact (1)
1
fact (1)

```
1   #include <iostream>
2
3   using namespace std;
4
5   int fact(int x){
6   //base case
7   if(x<=1)
8       return 1;
9   else
10      return x*fact(x-1);
11
12
13
14
15  }
16
17  int main()
18  {
19    cout<< fact(5);
20      return 0;
21  }
22
```

```
C:\Users\majid\Documents\codeblock\rec\bin\Debug\rec.exe
120
Process returned 0 (0x0)   execution time : 0.074 s
Press any key to continue.
```

main.cpp

```
1   #include <iostream>
2   using namespace std;
3
4   void fi(int n){
5
6   if(n<0)
7       return ;
8   else
9   for(int i=0;i<n;i++){
10      cout<<"*";
11  }
12  cout<<endl;
13  fi(n-1);
14
15  }
16
17  int main()
18  {
19      fi(5);
20
21        //return 0;
22  }
23
```

```
C:\Users\majid\Documents\codeblock\rec\bin\Debug\rec.exe
*****
****
***
**
*

Process returned 0 (0x0)   execution time : 0.086 s
Press any key to continue.
```

Thank You
&
Good luck