المحاضرة الثامنة

البرمجة 1

(*Programming Language 1*)

إعداد

الدكتور المهندس فراس الزين

الكلمات المفتاحية

البيانات , هيكل , تركيب , المصفوفات , متداخل .

Data , structure , struct , array , combine .

الفصل
10

هياكل البيانات
Data Structures

مقدمة:

تمثل الدوال والمصفوفات طرقاً من طرق هيكلة البرمجة، حيث أنها تساعد في تنظيم وتسهيل كتابة البرنامج.
كما تمثل الهياكل طريقة أخرى من طرق هيكلة البرمجة.

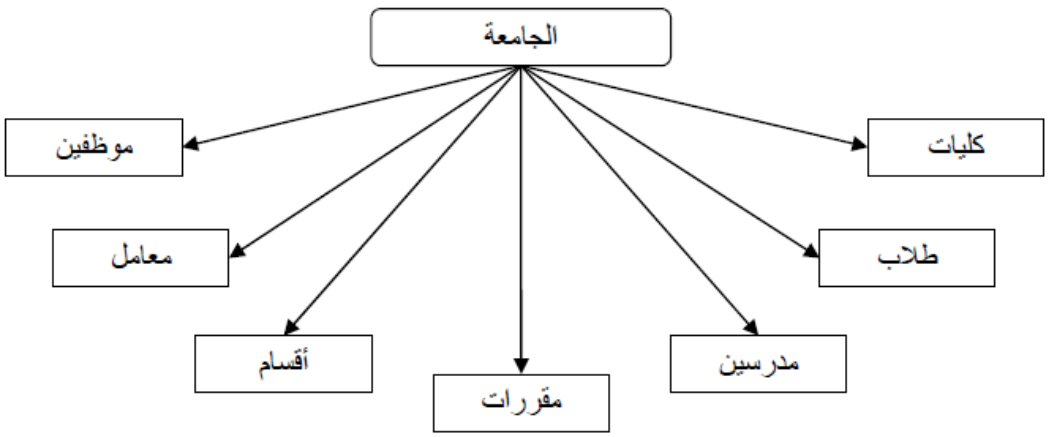
تعريفها:

يطلق عليها هياكل أو تراكيب أو سجلات وهي تعني التعامل مع مجموعة من البيانات كوحدة واحدة .
أو هي مجموعة من المتغيرات والصفات والخصائص "الدوال" التي تندرج تحت بناء واحد "هيكل".

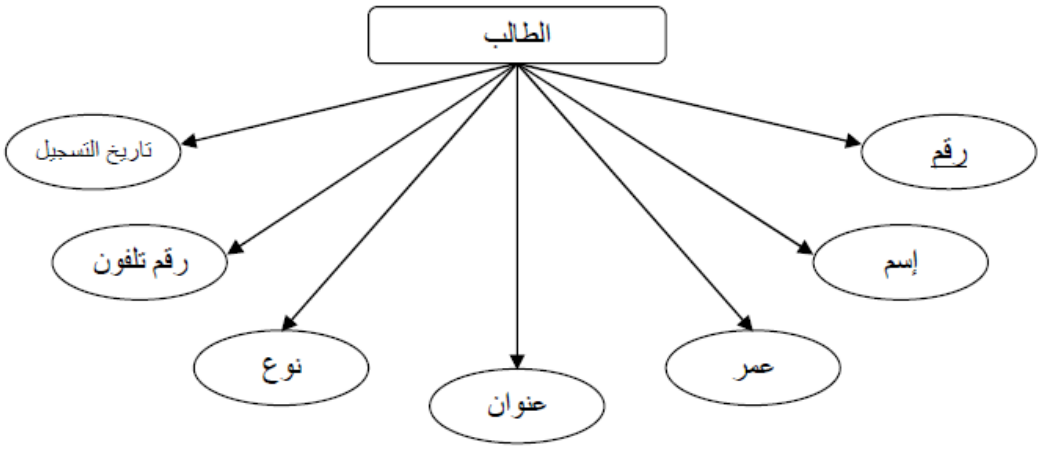
الهيكل التنظيمي:

فكرة هياكل البيانات تشبه الهياكل التنظيمية لأي مؤسسة أو شركة، فمثلاً الجامعة تحتوي على كليات ومقررات ومدرسين وطلاب، لكن كلمة "كلية تقنية المعلومات" أو "مادة C++" مبهمة إذا ذكرت بدون الإشارة إلى انتمائها فيجب الإشارة إليها والى الجهة التي تتبعها مثل "جامعة سبأ - كلية تقنية المعلومات" أو "جامعة سبأ - تقنية معلومات - مادة C++".

وبالتالي عند عمل أي مشروع برمجي يجب تحويل هيكل المنظمة إلى هيكل بيانات بحسب الاحتياج، فيجب هيكلة ما لا يمكن أن تستمر المنظمة في عملها بدونه وعدم هيكلة الأجزاء غير الأساسية في المنظمة، ولكن هذا ليس شرطاً فللمبرمج الحرية في تحديد ما يجب أن يهيكله وما لا يجب بحسب المشروع الذي ينوي بناءه.
فإذا كان مشروعه نظام جامعة متكامل فمن الضروري أن يحتوي النظام على كل أجزاء الجامعة الرئيسية أما إذا كان برنامج بصمة الكترونية فإنه يحتاج لمعرفة أسماء الموظفين فقط.



وكل جزء من أجزاء المنظمة يحتوي على خصائص فالكلية لديها "اسم" و "أقصى عدد طلاب" و "أقل عدد" والطالب يحتوي على "رقم" و"اسم" و "عمر" و "عنوان" .. الخ.

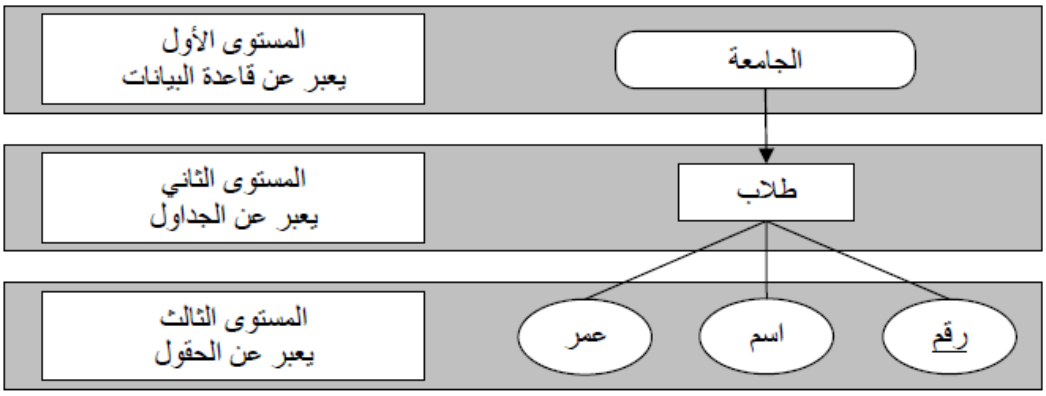


وهناك علاقة بين أجزاء المنظمة فهناك علاقة بين الكليات والطلاب من نوع (واحد الى كثير) فالطالب يدرس في كلية واحدة بينما الكلية يدرس فيها أكثر من طالب (راجع مقرر قواعد البيانات) .

وقد تتطلب العلاقات غير الواضحة أو علاقات كثير لكثير إنشاء كيان وهمي مثل (مقررات الكلية) لربط المقرر بالكلية.

في الحياة الواقعية تفرض طبيعة العمل نفسها على طبيعة العلاقات بين مكونات المنظمة وما يهم المبرمج أن يعكس طبيعة هذه العلاقة في برنامجه بالشكل الأنسب، وهو ليس بهذه السهولة فأي مشروع برمجة يحتاج إلى تحليل وتصميم ثم برمجة، وكل وظيفة من هذه الوظائف تحتاج الى متخصصين:

المحلل : يجمع البيانات ، ويربط المعلومات مع بعض، ولديه خبرة في العلاقات العامة.
المصمم : يأخذ نتائج التحليل ويحولها الى شكل يعبر عن الهيكل ، ولديه خبرة في برامج التصميم.
المبرمج : يحول تصميم الهيكل الى واجهات وأوامر وأكواد.



التركيب Struct:

بعد المقدمة النظرية ننتقل إلى الجزء العملي من الهياكل:

الصيغة العامة Public formula:

```
1.  struct struct_name
2.  {
3.      DataType member1;
4.      DataType member2;
5.      DataType member3;
6.  }
```

ملاحظات:

١. يكتب التركيب قبل الدالة الرئيسية:

مثال لكيان طالب:

```
1.  struct students
2.  {
3.      int num;
4.      char name[20];
5.      int age;
6.      string address;
7.      double phone;
8.  };
9.  void main()
10. {
11.
12. }
```

رقم
اسم
عمر
عنوان
تلفون

٢. يصبح التركيب نوعاً من أنواع البيانات مثله مثل int أو char ... الخ.

Struct:

num	name	age	address	phone
4	20	4	8	4

= 44 byte

٣. يعرّف على التراكيب متغيرات:

```
students stud;
```

إسناد قيم للتركيب:

١. إسناد القيم دفعة واحدة:

```
students stud = {1, " ali ", 25, " sanaa ", 123.456};
```

٢. إسناد كل قيمة بشكل مستقل:

```
1.  students stud;
2.  stud.num = 1;
3.  stud.name = "ali";
4.  stud.age =  25;
5.  stud.address = "sanaa";
6.  stud.phone = 777777777;
```

٣. ويمكن استخدام الدالة cin لطلب البيانات من المستخدم (أثناء التشغيل):

```
1.  students stud;
2.  cout<< "Enter number: ":      cin >> stud.num;
3.  cout<< "Enter name: ":        cin >> stud.name;
4.  cout<< "Enter age: ":         cin >> stud.age;
5.  cout<< "Enter address: ":     cin >> stud.address;
6.  cout<< "Enter phone: ":       cin >> stud.phone;
```

يمكن تعريف نوع التركيب لمتغير بعد بناءه مباشرة:

```
1.  struct notebook
2.  {
3.      int num;
4.      char name[20];
5.      double phone;
6.  } note;
7.  void main()
8.  {
9.       note.num = 1;
10. }
```

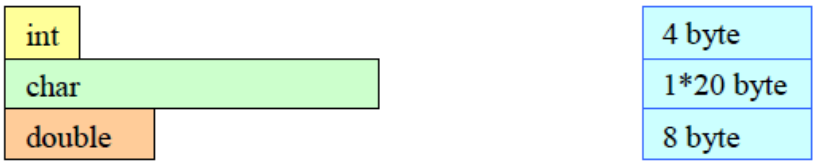
لاحظ المتغير note في السطر رقم (٦).

إسناد قيم للمتغير من نوع notebook

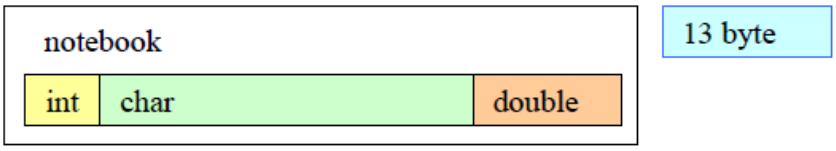
حجم التركيب:

يأخذ التركيب إجمالي حجم أنواع البيانات الداخلة في تركيبه:

حجم أنواع بيانات العادية:



حجم التركيب notebook (نوع بيانات مركب):



ملاحظة:
جاء التركيب لحل مشكلة المصفوفات في أنه يمكنه احتواء بيانات مختلفة الأنواع.

ملاحظات:
١) في الشكل التالي نلاحظ أن بيئة C++ تعطينا قائمة منسدلة بكل خصائص التركيب مما يسهل علينا كتابة برنامجنا.

```
#include <iostream>
#include <string>

using namespace std;

struct students{

    int num;
    char name[20];
    int age;
    string address;
    double phone;
}

void main(){

    students stud = {1, "ali", 25, "sanaa", 777777777};

    cout << stud.
}
```

٢) التركيب في الشكل أعلاه لا يستطيع خزن بيانات أكثر من طالب واحد، فعند إدخال بيانات طالب ثاني سيتم حذف بيانات الطالب السابق، وهذا يشبه إسناد القيم للمتغير، فكلما تسند قيمة جديدة للمتغير فإن القيم السابقة تحذف، ولحل هذه المشكلة نستخدم المصفوفات.

استخدام المصفوفات في التراكيب:
البرنامج التالي عبارة عن نظام لتسجيل بيانات الطلاب وعرضها بشكل منسق:

```
#include <iostream>
#include <string>

using namespace std;

struct students{
        int num;
        char name[20];
        int age;
        string address;
        double phone;
};
```

```
void main(){
    students   stud[10];
    for(int i=0; i<3; i++){
        stud[i].num = i;
        cout << "Student number: " << i+1 << endl;
        cout << "--------------------\n\n";

        cout << "Name: ";          cin >> stud[i].name;
        cout << "Age: ";           cin >> stud[i].age;
        cout << "Address: ";       cin >> stud[i].address;
        cout << "Phone: ";         cin >> stud[i].phone;
        system("cls");
    }

    cout << "Num\tName\tAge\tAddress\tPhone\n";
    cout << "----------------------------\n";

    for(i=0; i<3; i++){
        cout << i+1 << "\t";
        cout << stud[i].name << "\t";
        cout << stud[i].age  << "\t";
        cout << stud[i].address << "\t";
        cout << stud[i].phone  << "\n";
    }

}
```

تعريف مصفوفة من نوع التركيب "Stud"

مخرجات الشاشة:

```
Num     Name    Age     Address Phone
----------------------------
1       ali     21      sana'a  177
2       bader   22      aden    277
3       saeed   23      amran   377
Press any key to continue
```

التركيب المتداخل:

هو عبارة عن تركيب داخل تركيب. فقد نحتاج في برنامجنا السابق لتاريخ ميلاد الطالب ولكن لا يوجد في لغة C++ نوع بيانات للتاريخ، لذلك سنقوم بعمل تركيب لتعريف نوع بيانات جديد يمثل التاريخ بالطريقة التي تناسبنا:

- **إنشاء تركيب "تاريخ":**

```
1.  struct date {
2.          int day;
3.          int month;
4.          int year;
5.  };
6.  struct students{
7.          int num;
8.          char name[20];
9.          int age;
10.         string address;
11.         double phone;
12.         date birthdate;
13. };
```

ملاحظات:

- يجب أن يكون التركيب المضمّن قبل التركيب المتضمن له، فيجب كتابة تركيب التاريخ قبل تركيب الطلاب.
- يمكن بدلا من عمل تركيب للتاريخ تعريف متغير من نوع نصي string ولكن التركيب يوفر لنا سهولة في الوصول إلى اليوم أو الشهر أو السنة .

- **إدخال قيمة لخاصية في تركيب داخلي:**

```
1.  void main() {
2.      students stud;
3.      cin >> stud.birthdate.day;
4.      cin >> stud.birthdate.month;
5.      cin >> stud.birthdate.year;
6.      cin >> stud.address;
7.      cin >> stud.phone;
8.  }
```

- برنامج لإدخال و عرض بيانات الطلاب (مصفوفة تركيب):

```
#include <iostream>
#include <string>
using namespace std;

struct date{
        int day;
        int month;
        int year;
};
struct students{
        int num;
        char name[20];
        int age;
        string address;
        double phone;
        date birthdate;
};

void main(){
        students stud[10];

        for(int i=0; i<3; i++){
                stud[i].num = i;
                cout << "Student number: " << i+1 << endl;
                cout << "--------------------\n\n";

                cout << "Name:\t\t";          cin >> stud[i].name;
                cout << "Age:\t\t";           cin >> stud[i].age;
                cout << "Address:\t";         cin >> stud[i].address;
                cout << "Phone:\t\t";         cin >> stud[i].phone;
                cout << "Birth day:\t";       cin >> stud[i].birthdate.day;
                cout << "Birth month:\t";     cin >> stud[i].birthdate.month;
                cout << "Birth year:\t";      cin >> stud[i].birthdate.year;
                system("cls");
        }
```

```
        cout << "Num\tName\tAge\tAddress\tPhone\tBirthdate\n";
        cout << "---------------------------------------\n";
        for(i=0; i<3; i++){
                cout << i+1 << "\t";
                cout << stud[i].name << "\t";
                cout << stud[i].age << "\t";
                cout << stud[i].address << "\t";
                cout << stud[i].phone << "\t";
                cout << stud[i].birthdate.day << "/" ;
                cout << stud[i].birthdate.month << "/";
                cout << stud[i].birthdate.year << "\n";
        }
}
```

- شاشة إدخال البيانات:

```
Student number: 1
-----------------

Name:           ali
Age:            21
Address:        sana'a
Phone:          177
birth day:      1
birth month:    1
birth year:     1989_
```

- شاشة عرض النتائج:

```
Num     Name    Age     Address Phone   Birthdate
---------------------------------------
1       ali     21      sana'a  177     1/1/1989
2       badr    22      aden    277     2/2/1988
3       omar    23      amran   377     3/3/1987
Press any key to continue_
```

تمرين :

في المثال السابق أضف إمكانية للبرنامج يتيح للمستخدم أن يحدد عدد الطلاب الذين يريد أن يدخل بياناتهم.