جَامعة المَنارة
MANARA UNIVERSITY

المعلوماتية كلية الهندسة

مقرر مدخل إلى الخوارزميات والبرمجة
Introduction to Algorithms and Programming

ا. د. علي عمران سليمان

محاضرات الأسبوع العاشر

الفصل الثاني 2022-2023

المصفوفات Arrays

عناوين الفصل :

4-0- مقدمة.	4-1-4-3 توليد الأعداد العشوائية.
4-1- المصفوفات وحيدة البعد one. dimension array	4-1-4-4- لعبة النرد Dice-rolling
4-1-1-الإعلان عن النسقDeclaring Array	4-2- المصفوفات متعددة الأبعاد Multiple dimension arrays
4-1-2- إدخال و إخراج عناصر النسق.	4-2-1- الإعلان عن المصفوفة متعددة الأبعاد Declaring multiple dimension arrays
4-1-3- إعطاء قيم ابتدائية لعناصر النسق.	4-2-2- إدخال المصفوفة وإخراجها
4-1-4- تطبيقات على المصفوفات.	4-3- فرز النسق
4-1-4-1متوسط نسق	4-4- البحث ضمن النسق
4-1-4-2-المخطط البياني	

المحاضرة من المراجع :

- Deitel & Deitel, C++ How to Program, Pearson; 10th Edition (February 29, 2016)
- **د.علي سليمان, مدخل إلى الحاسوب والخوارزميات, جامعة تشرين 2006-2005**

3-4 فرز النسق.
Array sort

الفرز: وضع المعطيات وفق ترتيب معين تصاعدي أو تنازلي.

تعتبر عملية فرز المعطيات من أهم التطبيقات وتظهر في البحث (ما فائدة قاموس فيه ملايين الكلمات أو دليل هاتف غير مرتبين).

يوجد كم هائل من خوارزميات الترتيب ولايزال مجال الاجتهاد مفتوح لابتكار خوارزميات جديدة

الفرز الفقاعي Bubble sort

تعتمد على القيام بأكثر من مرور على عناصر النسق.

في كل مرور يتم مقارنة كل زوجين متتالين من عناصر النسق، إذا كان هذان الزوجان مرتبين تصاعدياً (أو متساويين) فإننا نبقيهما على حالهما، أما إذا كانا مرتبين تنازلياً فإننا نقوم بالمبادلة بينهما ضمن النسق (فرز فقاعي تصاعدي).

BUBBLE SORT ALGORITHM

BUBBLE SORT ALGORITHM

(* Algorithm to sort the list A[0],...A[n-1] in ascending order. *)

1. set scan equal to 0

2. while scan less or equal to n-1 do the following:

a. set pos equal to 0.

b. while pos less or equal to n-2 do the following:

b1. if A[pos] > A[pos+1] then do the following:

(* swap A[pos] ⟷ A[pos+1])

b11. set temp equal to A[pos]

b12. set A[pos] equal to A[pos+1]

b13 set A[pos+1] equal to temp

b2.set pos equal to pos+1

c. set scan equal to scan+1

```
void main()
{const int arraysize=5; int
a[arraysize]={10,8,6,4,2};
int hold,i;
cout<<"data items in original order:\n";
for(i=0;i<arraysize;i++)cout<<a[i]<<"  ";

for(int pass=0 ;pass<arraysize;pass++)
  for(i=0;i<arraysize-1;i++)
  if(a[i]>a[i+1])
   {hold=a[i];  a[i]=a[i+1]; a[i+1]=hold;}

cout<<"\ndata items in ascending order:\n";
for(i=0;i<arraysize;i++) cout<<a[i]<<"  ";
system ("pause");
}
```

تحقيق الخوارزمية -1-

ليكن لدينا النسق التالي:

25	20	16	12	7	5	2	1
A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]

المسح الأول (scan = 0 ; pos = 0 .. N-2) مبين في الجدول

0	1	2	3	4	5	6	7		
25	20	16	12	7	5	2	1	scan=0;pos=0	نبادل 0 و1
20	25	16	12	7	5	2	1	scan=0;pos=1	نبادل 1 و2
20	16	25	12	7	5	2	1	scan=0;pos=2	نبادل 2 و3
20	16	12	25	7	5	2	1	scan=0;pos=3	نبادل 3و 4
20	16	12	7	25	5	2	1	scan=0;pos=4	نبادل 4 و5
20	16	12	7	5	25	2	1	scan=0;pos=5	نبادل 5 و6
20	16	12	7	5	2	25	1	scan=0;pos=6	نبادل 6 و7
20	16	12	7	5	2	1	25	scan=0;pos=7	نهاية المسح الأول

وهكذا نجد أنه في نهاية المسح الأول أصبح أكبر العناصر وهو 25 في موقعه الصحيح وقد قمنا بـ N-1=7 عمليات تبديل في الحالة الأسوأ حيث كان العنصر الأكبر في بداية النسق.

تحقيق الخوارزمية -2-

وتم تطويرها على مراخل فمثلاً إذا كانت التبديلات كثيره يتم معرفة موقع أكبر العناصر ومبادلته مع العنصر الآخير وتالي أكبرها ومبادلته مع ماقبل الآخير وهكذا BUBBLE B

اصبح لدينا النسق التالي:

20	16	12	7	5	2	1	25
A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]

المسح الثاني (scan = 1 ; pos = 0 .. N-2)

0	1	2	3	4	5	6	7		
20	16	12	7	5	2	1	25	scan=1;pos=1	نبادل 0 و1
16	20	12	7	5	2	1	25	scan=1;pos=1	نبادل 1و2
16	12	20	7	5	2	1	25	scan=1;pos=2	نبادل 2و3
16	12	7	20	5	2	1	25	Scan=1;pos=3	نبادل 3و4
16	12	7	5	20	2	1	25	scan=1;pos=4	نبادل 4و5
16	12	7	5	2	20	1	25	scan=1;pos=5	نبادل 5و6
16	12	7	5	2	1	20	25	scan=1;pos=6	لا نبادل 6و7
16	12	7	5	2	1	20	25	scan=1;pos=7	نهاية المسح الثاني

وهكذا نجد أنه في نهاية المسح الثاني أصبح ثاني أكبر العناصر وهو 20 في موقعه الصحيح وقد قمنا بـ N-3=6 عمليات تبديل في الحالة الأسوأ حيث كان العنصر في بداية النسق.

وسنتابع مع المسوحات الباقية الثالث والرابع ....حتى السابع لنصل إلى نسق مصنف تصاعدياً.

خوارزمية الترتيب بالتبديل
SWAP SORT ALGORITHM

SWAP SORT ALGORITHM

(* Algorithm to sort the list A[0],...A[n-1] in ascending order. *)

1. set first equal to 0

2. while (first less or equal to n-2) do the following:

a. set second equal to first +1 .

b. while second less or equal to n-1 do the following:

b1. if A[first] > A[second] then do the following:

(* swap A[first] ⟷ A[second])

b11. set temp equal to A[first]

b12. set A[first] equal to A[second]

b13. set A[second] equal to temp

b2.set second equal to second +1

c. set first equal to first +1

```
int main()
{const int arraysize=5; int
a[arraysize]={10,8,6,4,2};
int hold,i,f,s;
cout<< "data items in original order:\n";
for(i=0;i<arraysize;i++) cout<<a[i]<<" ";

for(f=0 ;f<arraysize-1;f++)
  for(s=f+1;s<arraysize;s++)
   if(a[f]>a[s])
    {hold=a[f];  a[f]=a[s]; a[s]=hold;}

cout<<"\ndata items in ascending order:\n";
for(i=0;i<arraysize;i++) cout<<a[i]<<"  ";
system ("pause");return 0;}
```

تحقيق الخوارزمية -1-

ليكن لدينا النسق التالي:

25	20	16	12	7	5	2	1
A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]

المسح الأول (first = 1 ; second = 2 .. N-1):

0	1	2	3	4	5	6	7		
25	20	16	12	7	5	2	1	first=0;second=1	نبادل 0 و1
20	25	16	12	7	5	2	1	first=0;second=2	نبادل 1و2
16	25	20	12	7	5	2	1	first=0;second=3	نبادل 1و3
12	25	20	16	7	5	2	1	first=0;second=4	نبادل 1و4
7	25	20	16	12	5	2	1	first=0;second=5	نبادل 1و5
5	25	20	16	12	7	2	1	first=0;second=6	نبادل 1و6
2	25	20	16	12	7	5	1	first=0;second=7	نبادل 1و7
1	25	20	16	12	7	5	2	first=1;second=8	نهاية المسح الأول

وهكذا نجد أنه في نهاية المسح الأول أصبح أصغر العناصر وهو 0 في موقعه الصحيح وقد قمنا بـ N-1=7 من عمليات المبادلة في الحالة الأسوأ حيث كان العنصر الأصغر في نهاية النسق.

تحقيق الخوارزمية -2-

وتم تطويرها على مراخل فمثلاً إذا كانت التبديلات كثيره يتم معرفة موقع أكبر العناصر ومبادلته مع العنصر الآخير وتالي أكبرها ومبادلته مع ماقبل الآخير وهكذا BUBBLE B

اصبح لدينا النسق التالي:

20	16	12	7	5	2	1	25
A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]

المسح الثاني (first = 1 ; second = 2 .. N-1)

0	1	2	3	4	5	6	7		
1	25	20	16	12	7	5	2	first=1;second=2	نبادل 1و2
1	20	25	16	12	7	5	2	first=2;second=3	نبادل 2و3
1	16	25	20	12	7	5	2	first=2;second=4	نبادل 2و4
1	12	25	20	16	7	5	2	first=2;second=5	نبادل 2و5
1	7	25	20	16	12	5	2	first=2;second=6	نبادل 2و6
1	5	25	20	16	12	7	2	first=2;second=7	نبادل 2و7
1	2	25	20	16	12	7	5	first=2;second=8	نهاية المسح الثاني

وهكذا نجد أنه في نهاية المسح الأول أصبح ثاني أصغر العناصر وهو 2 في موقعه الصحيح وقد قمنا بـ N-2=6 من عمليات المبادلة.

وسنتابع مع المسوحات الباقية الثالث والرابع ....حتى السابع

4-4- البحث ضمن النسق.
Search of an array

يوجد خوارزميتن للبحث هما الخطية والثنائية.

مسلمة البحث:يجب معرفة أين وعن ماذا نبحث (المعطيات والمفتاح) .

مفهوم البحث: يتم التعامل مع كميات كبيرة من البيانات (المعطيات-Data) المرتبة وفق ما يعرف بالسجلات (Records) حيث يحتوي السجل الواحد على جملة من البيانات المرتبطة التي تمثل معلومات عن كينونة ما (Entity) يتعامل معها البرنامج حيث تعمل كنموذج حاسوبي لكائنات حقيقية وتشكل هذه البيانات ما نطلق عليه اسم حقول (Fields) السجل وكمثال لتكن جملة البيانات التالية التي تمثل معلومات عن كينونة أحد الطلاب في برنامج يخدم شعبة الامتحانات في الكلية.
لكل طالب رمز جامعي يميزه عن غيره وهو مفتاح البحث ومنه نصل لكل معلوماته.

خوارزمية البحث الخطي (التتابعي)

تقوم خوارزمية البحث الخطي على فكرة مسح عناصر النسق بالتتابع حتى الوصول إلى العنصر المطلوب أو الوصول حتى نهاية النسق حيث تتم مقارنة القيمة المراد إيجادها مع عناصر النسق واحدا تلو الآخر (بالتتابع)

LINEAR SEARCH ALGORITHM

جَامعة المَنارة
MANARA UNIVERSITY

برنامج البحث الخطي

LINEAR SEARCH ALGORITHM
(*Algorithm to search the list A[0],.....,A[n-1] for Item , Found is set to true and LOC is set to the position of Item if the search is successful ; otherwise , Found is set to false.*)

1. set Found equal to false.
2. set LOC equal to 0.
3. while LOC<= n-1 and not Found do the following :
a. if Item = A[LOC]then
b. set Found equal to true.
Else
c. increase LOC by 1.

```
void main(void)
{const int n=10;
 int A[n]={7,3,1,0,21,5,17,99,2,-5};
 for (int i=0;i<n;i++) cout<<setw(4)<<A[i];
cout<<endl;
for(int i=0;i<n;i++)cout<<setw(4)<<i;cout<<endl;
 int found=0,  loc=0, item;
 cout<<"Input Item to search for: "; cin>>item;

while ( (loc < n) && (!found) )
    {if (item== A[loc]) found = 1; else loc++;}

if(found)cout<<item<<" was found at location "
<<loc<<endl;
else  cout<<"item was not found!\n";
    system ("pause");
}
```

طريقة البحث الثنائي
Binary search

تطبق على المعطيات المصنفة فقط.

تقوم هذه الخوارزمية بمقارنة قيمة العنصر الواقع في منتصف النسق مع القيمة المفتاح التي نبحث عنها، فإذا تساوت القيمتان فهذا يعني أنه تم إيجاد القيمة التي نبحث عنها وإذا لم تحدث المساواة فيتم إعادة صياغة المسألة لتصبح مكافئة لمسألة إيجاد عنصر ضمن أحد نصفي النسق السابق، أي إذا كانت القيمة المفتاح تقل عن قيمة العنصر الواقع في منتصف النسق فإن القيمة التي نبحث عنها تقع في النصف الأول من النسق وإلا فإنها تقع في النصف الثاني منه وذلك على افتراض أن عناصر النسق مرتبة تصاعدياً.
في كل عملية لم نجد العنصر يتم ترك نصف المعطيات التي نبجث ضمنها.

الخوارزمية بلغة الشفرة الزائفة

بفرض أن اللائحة مؤلفة من A[0] وحتى A[n-1] حيث n هو عدد العناصر.

A[0]	A[2]	A[3]	...	A[mid]	...	A[n-2]	A[n-1]

1- نوجد العنصر الأوسط في لائحة العناصر A[mid] والذي يقسم اللائحة إلى لائحتين جزئيتين مرتبتين.
2- نقارن هذا العنصر مع المفتاح Key الذي نبحث عنه وعندئذ سنواجه الحالات الثلاث التالية:
أ - Key = A[mid] وعندها يكون البحث قد انتهى والعنصر تم إيجاده.
ب- Key < A[mid] وعندها قد يكون العنصر واقعا حتماً في النصف الأول من اللائحة.
ج- Key > A[mid] وعندها قد يكون العنصر واقعا حتماً في النصف الثاني من اللائحة.
3- إذا لم يتم الوصول للعنصر ولم تكن لائحة العناصر التي نتعامل معها فارغة نكرر الخطوات السابقة على اللائحة الفرعية التي نتوقع أن تحوي العنصر (النصف الأول أو الثاني) وذلك تبعا لنتيجة المقارنة السابقة.

pseudocode Binary search

كود البحث الثنائي

BINARY SEARCH ALGORITHM

*(*Algorithm to search the list A[0],.....A[n-1] for Item using a binary search. Found is set to true and Mid is set to the position of Item if the search is successful; otherwise Found is set to false.*)*

1. set Found equal to false.
2. set First equal to 0.
3. set Last equal to n-1.
4. while First <= Last and not Found do the following :
   - a.calculate Mid = (First +Last) div 2.
   - b. if Key >A[Mid] then
     - b1. set First equal to Mid +1.
   - c. else if Key < A[Mid] then
     - c1.set last =Mid-1
   - d.else set Found equal to true.

```
main()
{       const int n=8;   int target;
int a[n]={22,33,44,55,66,77,88,99};
cout<<"a array :";
for(int i=0;i<8;i++)cout<<a[i]<<" ";cout<<endl;
do { cout<<"enter target,0 to end:";
cin>>target;
int loc,left=0, right=n-1, found=0;
while(!found && left<=right){
   loc=(left+right)/2;  found=(a[loc]==target);
    if(a[loc]<target) left=loc+1;
    else right=loc-1; }
if(found)cout<<target<<"is at:"<<loc<<"\n";
if(!found)cout<<target<<"is not found.\n";
}while(target!=0);  }
```

نتيجة برنامج البحث الثنائي

خرج البرنامج السابق.

```
a array : 22  33  44  55  66  77  88  99
enter target , 0 to end:55
55  is  at : 3
enter target , 0 to end : 6
6  is not  found.
enter target , 0 to end : 2
2  is not  found.
enter target , 0 to end : 99
99  is  at : 7
enter target , 0 to end :0
0  is not  found.
Press any key to continue
```

هذا البرنامج يسمح بالبحث عن القيمه الهدف target حتى يتم إدخال الرقم 0، في حال وجدت القيمة الهدف يطبع فهرسها وإلا تطبع رسالة تعلن عن عدم إيجادها.

في حال كان النسق مصنف تنازلياً إما أن يبدل شرط التحقق أو الانتقال للجزء اليميني بالانتقال للجزء اليساري

بعض الأسئلة المطلوب الإجابة عنها 1

6- 1- قام مدرس مادة البرمجيات بإجراء امتحان لطلاب الصف البالغ عددهم 11 طالباً وبعد أن قام بتصحيح الأوراق أراد أن يجري العمليات التالية على علامات الطلاب ضمن الصف والمطلوب:

1- إدخال قيم علامات الطلاب.

2- طباعة قيم علامات الطلاب في المادة.

3- إيجاد أعلى علامة وأدنى علامة وموقعهما.

4- حساب نسبة النجاح في المادة (علامة النجاح أكبر أو تساوي 60).

5-هناك معيار في تقييم نتائج مادة يعتبر أنه إذا كان أكثر من نصف الطلاب قد حصلوا على علامة أكبر من متوسط العلامات فإن النتائج منطقية. اختبر فيما إذا كانت علامات الطلاب في المادة منطقية.

6-1- هناك معيار أخر في تقييم نتائج مادة يعتبر أنه إذا كان مجموع انحرافات علامات الطلاب عن العلامة الوسطية موجباً فإن النتائج منطقية. اختبر فيما إذا كانت علامات الطلاب في المادة منطقية.

7- إنشاء نسخة من علامات الطلاب بحيث يتم استبدال كل علامة ناجحة بالقيمة 1 وكل علامة راسبة بالقيمة 0.

8-طور البرنامج السابق ليتعامل مع عدة مواد وعدة طلبة ويحسب متوسطات علامات المواد ومتوسطات علامات الطلبة

بعض الأسئلة المطلوب الإجابة عنها

8- يراد تخزين النتائج الامتحانية لثلاثين طالباً في 7 مواد مع معدلاتهم وأرقامهم الجامعية وأسماء المواد في مصفوفات بحيث يتم العرض وفق التالي السطر الأول أسماء المواد ويظهر الرقم الجامعي وكائنه في العمود الأول والأعمدة السبعة التالية تحوي درجات المواد والعمود الأخير يحوي المعدل والمطلوب:

1- تعريف مصفوفات أحادية البعد لكل من الرقم، اسم المقرر والمتوسط ، مصفوفة ثنائية البعد للدرجات.

2- إدخال القيم المطلوبة إلى المصفوفات الرقم الجامعي والعلامات.

3- حساب معدلات الطلاب وتخزينها في الأماكن الملائمة.

4- البحث عن رقم الطالب الذي حصل على أعلى معدل.

5- تحديد الطلاب غير الموفقين وهم الحاصلين في خمس مواد أوأكثر على علامة أقل من 60.

6- تجديد عدد الطلبة الحاصلين على معدل وفق المجالات التالية [0-1[و [1-1.5[و [1.5-2[و [2-3[و [3-4]