جامعة المنارة

كلية الهندسة

قسم الهندسة المعلوماتية

مقرر برمجة1

الجلسة الثانية

المصفوفات والبحث الثنائي

الفصل الثاني 2022-2023

المصفوفات

الجلسة الثانية

الغاية من الجلسة:

تهدف هذه الجلسة إلى تعميق ما تعلمه الطالب عن التعامل مع المصفوفات والثنائية وتطبيق بعض العمليات الحسابية عليها بالإضافة الى البحث الثنائي في المصفوفة الثنائية

المصفوفة الثنائية

لا تختلف المصفوفات الثنائية كثيراً عن المصفوفة أحادية الابعاد .تختلف فقط في العنونة فبدل من "index" واحد نستخدم اثنان واحد للصفوف وأخر للأعمدة مثال مصفوفة (3*2) أي مكونة من ثلاث صفوف وعموديين

العنونة تكون (رقم العمود ,رقم الصف) وان العنونة للاعمدة والصفوف تبدأ من الصفر وليس من الواحد أي لو أردنا الوصول إلى عنصر في الصف الثاني العمود الاول تكون عنونه (1,0) والمصفوفة الثنائية أيضاً هي مجموعة خاليا متتالية في الذاكرة تحجز لغرض خزن معلومات معينة في داخلها كأن نخزن في داخلها أرقام أو أحرف.

تبقى القيم مخزنة داخل المصفوفة حتى نغلق البرنامج إذا لم نغيرها داخل البرنامج .في تعريف المصفوفة يجب ان نذكر عدد المواقع التي نحتاجها في العمل في بداية البرنامج حتى يحجزها المترجم للمصفوفة ويخزن قيم أخرى داخل هذه المواقع تبقى محجوزة فقط لعناصر المصفوفة. كذلك تخزين هذه العناصر في الذاكرة يكون نفس طريقة تخزين المصفوفة الاحادية لكن هنا يخزن صف وبعده صف أخر بالتسلسل إلى أن تنتهي الصفوف.

تمرين محلولة

التمرين الأول : اكتب برنامج ينفذ المهام التالية:

1. التصريح عن مصفوفة ثنائية البعد بحجم 3*2 من الشكل

$$\begin{bmatrix} 5 & 1 & 3 \\ 0 & -6 & 9 \end{bmatrix}$$

2. طباعة عناصر العمود الثاني من المصفوفة
3. طباعة عناصر الصف الأول من المصفوفة
4. طباعة المصفوفة بحيث نطبع العناصر التي قيمتها أكبر من 4، ونطبع 0 فيما عدا ذلك، مع العلم أنه يجب طباعة المصفوفة بحيث تحافظ على شكلها "أي 3 عناصر في الصف الأول، ثم 3 عناصر في الصف الثاني"
5. تعديل قيم المصفوفة، في حال كان العنصر يساوي الصفر، نسند له قيمة 0، وفي حال كان زوجي نسند له قيمة 1، وفي حال كان فردي نسند له -1، ثم طباعة المصفوفة.

فكرة الحل :

بعد التصريح عن المصفوفة الثنائية نستطيع طباعة عناصر سطر معين او عمود معين وذلك بتثبت رقم هذا العمود أو سطر

وذلك بفرض شرط مساواة قيمة العمود او السطر للقيمة المطلوب طباعتها

في الطلب التالي يجب ان نحدد شرط قيمة العنصر في المصفوفة x[i][j] اكبر من قيمة محددة والعناصر التي تحقق هذا الشرط فقط تطبع

الطلب الاخير يعتمد على اسناد قيم جديدة لعناصر المصفوفة من خلال المرور على جميع عناصر المصفوفة واختبارها فيما اذا كانت صفرية تستبدل بصفر واذا كانت زوجية أي الشرط بافي القسمة على 2 يساوي الصفر تستبدل ب 1 واذا كانت فردية اي باقي القسمة على 2 لايساوي الصفر تستبدل ب -1

```
#include <iostream>
using namespace std;
int main()
{
/*  تعليمة تعريف مصفوفة ثنائية البعد من سطرين وثلاثة اعمدة وقد تم تهيئتها
بالقيم المطلوبة في التمرين السابق  */
    int a[2][3] = {{5,1,3} , {0,-6,9}};
/*print the second col 1 طباعة العمود الثاني أي العمود ذي لبفهرس
8باعتبار يبدأ العد من العمود صفر*/
    cout<< "the second col : " << endl;
    for(int i = 0 ; i < 2 ; i++)
    {
        for(int j = 0 ; j < 3 ; j++)
        {
            //here we put our code
            if(j == 1){
                cout << a[i][j];
            }
            cout << "  ";
        }
        cout << endl;
        // لطباعة عناصرالصف سطر سطر وليس جميع الصفوف على سطر واحد
د المحافظة على العمو  */
    }
    //print the first row   i=0 طباعة السطر الاول ذي الفهارس صفر أي
    cout<< "the first row : " << endl;
 for(int i = 0 ; i < 2 ; i++)
    {
        for(int j = 0 ; j < 3 ; j++)
        {
            //here we put our code
            if(i == 0){
                cout << a[i][j];
            }
            //
            cout << "  ";
        }
        cout << endl;
    }
```

```
/*   طباعة جميع عناصر المصفوفة بشرط ان تحقق الشرط قيمة العنصر في المصفوفة
اكبر من اربعه والا سوف نطبع صفر   */
    //print the Items greater than 4
    cout<< "Items greater than 4 : " << endl;
    for(int i = 0 ; i < 2 ; i++)
    {
        for(int j = 0 ; j < 3 ; j++)
        {
            //here we put our code
            if(a[i][j] > 4)
            {
                cout<< a[i][j];
            }
            else
            {
                cout << "0";
            }
            //
            cout << "  ";
        }
        cout << endl;
    }
    /*update matrix تحديث قيم المصفوفة بحيث نسند القيمة صفر لعنصر
//  المصفوفة ذو القيمة صفر والقيمة
if else     واحد للعنصر الزوجي والقيمة -1 للعنصر الفردي وذلك باستخدام
تعليمة*/
    for(int i = 0 ; i < 2 ; i++)
    {
        for(int j = 0 ; j < 3 ; j++)
        {
            //here we put our code
            if(a[i][j] == 0 )
            {
                a[i][j] = 0;
            }else if (a[i][j] % 2 == 0 )
            {
                a[i][j] = 1;
            }else
            {
                a[i][j] = -1;
            }

            cout << "  ";
        }
        cout << endl;
    }
/*في الطلب السابق تم تغير القيم للمصفوفة في الذاكرة لكنها لم تظهر في
الخرج لذلك نحن بحاجة الى حلقتين وتعليمة الطباعة لطباعة عناصر
  المصفوفة الثنائية بعد التغير  */
    cout<< "show matrix: " << endl;
```

```
    for(int i = 0 ; i < 2 ; i++)
    {
        for(int j = 0 ; j < 3 ; j++)
        {
            //here we put our code
            cout<< a[i][j];
            //
            cout << "  ";
        }
        cout << endl;
    }
    return 0;
}
```

البحث الثنائي

إن كانت العناصر في المصفوفة مرتّبة ترتيبًا تصاعديًا، فيمكن تسريع عملية البحث، وذلك باستخدام طريقة البحث الثنائي Binary Search.

تبدأ عملية البحث الثنائي بالتحقق من القيمة الموجودة في منتصف المصفوفة (سنسمّي هذا الموقع location وقيمة العنصر الموجود في هذا الموقع a[location]). فإن كانت a[location]=target القيمة التي نبحث عنهاتتوقف عملية البحث مباشرة، وهذا نادر الحدوث.

إن معرفة قيمة العنصر الموجود في منتصف المصفوفة مفيد للغاية وسيساعد في توجيه عملية البحث نحو الطريق الصحيح. فإن كانت target < a[location] فإنّنا سنجزم بعدم إمكانية ظهور target في أي موقع أكبر من موقع location، وبهذا يمكن تجاهل النصف العلوي من المصفوفة في عمليات البحث التالية.

وإن كانت target > a[location] فإنّنا سنجزم بعدم إمكانية ظهور target في أي موقع أصغر من موقع location، وبهذا يمكن تجاهل النصف السفلي من المصفوفة في عمليات البحث التالية.

وبطبيعة الحال فإنّ نصف المواقع في المصفوفة ستستبعد من عملية البحث في كلا الحالتين.

بعد ذلك تتحقّق عملية البحث الثنائي من القيمة الموجودة في منتصف الجزء الذي يتوقّع أن تكون القيمة المطلوبة موجودة فيه، وبذلك يمكن استبعاد نصف المواقع المتبقية من عمليات البحث التالية. تستمر عملية البحث الثنائي بهذا الأسلوب إلى أن تعثر على العنصر المطلوب أو إلى حين الفراغ من جميع المواقع الموجودة في المصفوفة والتي يمكن أن تحتوي على قيمة target

التمرين الثاني : ابحث عن العدد المدخل من قبل المستخدم في المصفوفة أحادية البعد باستخدام البحث الثنائي

ملاحظة يجب ان تكون العناصر مرتبة تصاعديا قبل البحث بامكانك استخدام الترتيب الفقاعي اذا لم تكن مرتبة مسبقا

الطريقة الأولى : نستخدم حلقة while طالما أن شرط الحلقة محقق (!found && left<=right)

```
#include <iostream>
using namespace std;
int main()
{
const int n=8;
//القيمة التي سوف نحبث عنها
int target;
// تعريف مصفوفة احادية ذات الحجم عرف سابقا وتهيئتها بقيم معينة
int a[n]={22,33,44,55,66,77,88,99};
//ادخل العدد الذي تبحث عنه
cin>>target;
// location المؤشر الاوسط  العنصر الذي ستبحث عنه في المصفوفة  يدل على موقع
// left  يدل على موقع العنصر الاول في المصفوفة
// right  يدل على موقع العنصر الاخير في  المصفوفة

 int location,left=0, right=n-1; int found=0;
//طالما القيمة
while(!found && left<=right){
// البدء بالبحث من منتصف المصفوفة  location  من النوع الصحيح فيهمل الفواصل
%/
location=(left+right)/2;
 //  اذا ( تحقق شرط )location  تصبح  يساوي القيمة المدخلة found واحد
found=(a[location]==target);
//  أكبر من القيمة المراد البحث عنها location اذا كانت قيمة المؤشر الاوسط
//
if(a[location]<target)
// نغير المؤشر الايسر ليصبح بعد منتصف المصفوفة بمقدار واحد أي يصبح في النصف
//الايمن من المصفوفة
left=location+1;
//   نغير المؤشر الايمن ليصبح قبل منتصف المصفوفة بمقدار واحد أي يصبح في
//النصف الايسر من المصفوفة
else right=location-1; }
// اذا كانت قيمة  found  واحد نفذ
if(found)cout<<target<<"is at:"<<location<<"\n";
//  اذا كانت قيمة  found  لا تساوي واحد نفذ
if(!found)cout<<target<<"is not found.\n";
  return 0;}
```

طريقة ثانية للحل :

باستخدام حلقة whileمحققة دائما شرط الخروج في حال تم ايجاد العنصر باستخدام تعليمة break

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

```
int main() {

int n=8;
int target;
cin>>target;
int low=0,high=n-1,mid;
int arr[n]={12,24,36,44,55,66,77,89};

while(true)
{
 mid=(low+high)/2;

 if (target == arr[mid])
  {
    cout<<"target = "<<target <<" is found at index "<<mid<<"\n";
    break;

  }
  else if (target>arr[mid]&&mid!=high)// لم و المصفوفة في دليل أعلى الى الوصول عند نهائية حلقة في يدخل لا كي
 يجد الحل بعد
  {
   low=mid+1;
  }
  else if (target<arr[mid]&&mid!=low) لم و المصفوفة في دليل أصغر الى الوصول عند نهائية حلقة في يدخل لا كي
 يجد الحل بعد
   {
   high=mid-1;
   }
  else // المصفوفة في فيه نبحث الذي العنصر فيها نجد لا التي الحالة أجل من
   {
    cout<<"target = "<<target<<" is not found at any index \n";
    break;
   }

}
   return 0;
}
```