**المحاضرة السادسة**

**قسم الحاسبات**

**المرحلة الاولى**

مدرسة المادة : وداد عبد الخضر ناصر

تكملة موضوع الدوال

**الدوال الخالية void function**

**عندما تكون الدالة من نوع void لا تقوم هذه الدالة باسترجاع اي قيمة الى كلمة الاستدعاء لذلك لا يجب كتابة جملة return بداخلها والمثال التالي يوضح دالة نوعها void**

**#include<iostream.h>**

**void printmessage()**

**{**

**cout<<" c++ programming " ;**

**}**

**main ( )**

**{**

**printmessage( ) ;**

**return 0;**

**}**

**ارسال وامرار المصفوفات الى الدوال passing array to function**

**EX:write a program to enter integer array named A[6] in main program and then use the function to find the sum of it and print the sum in main program .**

**#include<iostream.h> int sum (int a[6]) { int i,s; s=0;**

الدالة والفرعية

نلاحظ ان جملة ال return s ستقوم باخراج قيمة ال s بعد انتهاء تنفيذ الدالة زالفرعية وعودتها الى كلمة الاستدعاء وتقوم بوضع هذه القيمة في اسم الدالة في كلمة الاستدعاء في الدالة الرئيسية ثم تنقل الى المتغير k ثم نكمل تنفيذ الجمل التي تليها في الدالة الرئيسية

**for(i=0;i<6; i++)**

**s=s+a[i];**

**return s;**

**}**

**main()**

**{ int i,k ,a[6];**

**for(i=0 ; i<6 ; i++)**

**{cout<<" enter the number"<< "\n";**

**cin>>a[i];**

**}**

كلمة استدعاء الدالة الفرعية

**k=sum(a)**

**cout<<"the sum ="<<k<<"\n";**

**return 0;**

**}**

**EX: write a program that use function to read integer array B[5] and find the sum of only even numbers and print the sum in main program . #include < iostream.h> int even( ) { int i, B[5] ; i=0; while(i<=4) { cin>>B{i]; if(B[i] %2= =0 ) s=s+B[i] ; i=i+1; } return s; }**

**main() {int k;**

**k=even( ) ;**

**cout<<" the sum= " << k ;**

**return 0;**

**}**

**EX:Write a program to enter the number and then test it if the number primary or not by using a function and print the message show if the number is primary or not in it .**

**#include<iostream.h>**

**int primaryornot(int x)**

**int i, k; {**

**k=1;**

**for(i=2 ;i<=x-1 ; i++)**

**if (x % i = = 0 )**

**{k=0;**

جملة ال if جملة واحدة داخل ال for

**break;**

**}**

**if(k==0)**

**cout<< x<< "is not primary "<<"\n" ;**

**else**

**cout<< x <<" is primary "<<"\n" ;**

**return 0;**

**}**

**main ()**

**{ int d ;**

**cin>> d ;**

**primaryornot(d) ;**

**return 0;**

**}**

**فكرة الاعداد الاولية ان العدد الاولي هو العدد الذي يقبل القسمة على واحد وعلى نفسة فقط و بدون باقي بينما باقي الارقام ايضا تقبل القسمة على نفسها وعلى واحد وبدون باقي وايضا تقبل القسمة على بعض الارقام وبدون باقي فتسمى هذه الاعداد بالاعداد الغير اولية فلكي نعرف ان العدد اولي او لا يكون كالاتتي انه عندما ندخل رقم فاننا نعد من واحد الى هذا الرقم المدخل فعلى سبيل المثال عندما ندخل الرقم 5 فنعد من واحد الى خمسة اي (1,2,3,4,5 ) ثم نترك الواحد والخمسة لان كل الارقام تقبل القسمة على واحد ووعلى نفسها وبدون باقي فيترك الواحد والخمسة في مثالنا فقط ناخذ (2,3,4 )ونقوم بقسمة العدد المدخل على كل منهم0 فقيمة العدد المدخل والذي هو في مثالنا العدد خمسة فاذا واحد من هذه الاعداد(2,3,4)عندما نقسم العدد خمسة عليةيكون بدون باقي فان العدد خمسة هو عدد غير اولي اما اذا كل الاعداد (2,3,4 ) عندما نقسم الخمسة عليهم يكون هناك باقي فان العدد هو اولي وهكذا بالنسبة الى بقية الارقام عندما نريد ان نعرف هل هي اولية او لا 0**

**EX: Write a program to enterr integer array a[3][3] in main program and then use function to find the square of element in main diagonal and then print the result array in it .**

**#include< iostream .h>**

**#include< math .h>**

**int diagonal (int a[3][3]) { int i , j ; for(i=0:i<3 ; i++) for(j=0 ; j<3 ; j++) if(i==j) a[i][j]= pow(a[i][j], 2)**

**for(i=0 ; i<3 ; i++ ) { for(j=0 ; j<3 ; j++) cout<< a[i][j] ; cout <<"\n" ; }**

**return 0;**

**}**

**main()**

**{ int i,j ,a[3][3] ; for(i=0 ; i<3 ; i++)**

**for(j=0 ; j<3 ; j++)**

**cin>> a[i][j] ;**

**diagonal(a);**

**return 0;**

**}**

**ex:Writ e a program to enter integer array a[5] in main program and then use two function.The first function use to sort array ascending and print the result array in it . The second function use to sort the array descending and print the result array in it .**

**#include<iostream.h>**

**int sortasc( int a[5] )**

الدالة الاولى

**{ int i ,j, t;**

**fot(i=0 ; i<=3 ; i++)**

**forj(j=i+1; j<=4 ;j++)**

**if (a[i]>a[j] )**

**{ t= a[i] ;**

**a[i]= a[j] ;**

**a[j] = t ;**

**}**

**ملاحظة: لايمكن ان تكون ال value بعد ال كلمة ال return عبارة عن مصفوفة لان بعد ال return تكون قيمة واحدة والمصفوفة تحتو ي على عدة قيم مخزونة في مواقع لذلك لا تستخدم ال return لاخراج المصفوفة بعد تغييرقيمها من الدالة الفرعية الى كلمة استدعائها لان المصفوفة بمجرد تغيير قيمها مثلا في الدالة الفرعية والعودة الى كلمة استدعائها فان القيم القديمة تلغى ونحتفظ بالقيم الجديدة لذلك الدالة الرئيسية ستتعرف على القيم الجديده للمصفوفة بعد الخروج من الدالة الفرعية الى كلمة استدعائها في الدالة الرئيسية لتكملة تنفيذ الجمل وهذا فقط يحدث مع المصفوفات(اي ان الدالة لا تستخدم ال return لاخراج المصفوفة الناتجة فيها لانها ستكون معروفة مباشرة في الدالة الرئيسية)**

**for(i=0 ; i<=4 ; i++)**

**cout<< a[i]<<"\n" ;**

**return 0;**

**}**

الدالة الثانية

**int sortdes( int a[5] )**

**{ int i ,j, t;**

**fot(i=0 ; i<=3 ; i++)**

**forj(j=i+1; j<=4 ;j++)**

**if (a[i]< a[j] )**

**{ t= a[i] ;**

**a[i]= a[j] ;**

**a[j] = t ;**

**}**

**ان كلمة الاسدعاء sortasc(a) تقوم باستدعاء الدالة الاولى وسوف نرسل لها المصفوفة a الغير المرتبة والتي ادخلناها في الدالة الرئيسية حيث تقوم الدالة الاولى بيرتيب المصفوفة المرسلة لها ترتيب تصاعديا وعندما تنتهي تنفيذ الدالة الاولى ترجع الى كلمة استدعائها في الدالة الرئيسية لتكمل تنفيذ الجمل التي تلي كلمة الاستدعاء وبعد عودتها الى كلمة الاستدعاءنكون قد حصلنا على مصفوفة مرتبة تصاعديا والمصفوفة القديمة الغيت اي هنا تغيرت المصفوفة من مصفوفة غير مرتبة الى مصفوفة مرتبة تصاعيا فان الدالة الرئيسية ايضا سوف تتعامل مع هذه المصفوفة المرتبة لان المصفوفة القديمة تغيرت واصبحت مرتية تصاعديا لذلك الدالة الرئيسية دائما تاخذ التغير الجديد الذي يطرا على المصفوفة ثم بعد الانتقال الى تنفيذ الجملة التي تلي كلمة استدعاء الدالة الاولى والتي هي ايضا كلمة استدعاء ولكن للدالة الثانية عن طريق كلمة الاستدعاءsortdes(a) فهنا سوف نرسل لهذه الدالة المصفوفة a المرتبة تصاعديا وليست المصفوفة الاولى التي ادخلناها لان المصفوفة الاولى تغيرت بعد اخالها الى الدالة الاولى واصبحت مرتية تصاعديا فبعد رجوعها الى الدالة الرئيسية ناخذ التغير الجديد الذي طرا على المصفوفة ثم تقوم الدالة الثانية باعادة ترتيب المصفوفة التي ادخلت لها ترتيب تنازليا اي ان التفيير النهائي الذي سيحدث على المصفوفة انها ستكون مرتبة تنازليا لان بعد انتهاء تنفيذ الدالة الثانية وعودتها الى كلمة الاستدعاء في الدالة الرئيسية سنكون قد حصلنا على مصفوفة مرتبة تنازليا**

**for(i=0 ; i<=4 ; i++)**

**cout<< a[i]<<"\n" ;**

**return 0;**

**}**

**main()**

**{ int a[5]**

**for(i=0; i<=4 ; i++)**

**cin >> a[i];**

كلمة استدعاء الدالة الاولى

**sortasc( a) ;**

**sortdes(a);**

**return0;**

**}**

**4-write a program to enter 10 integer numbers and find the sum of positive numbers only in main program. Ues the function to test if the number is positive or not .**

**#include<iostream.h>**

**int positive(in t x )**

**{ int k;**

**if(x> 0)**

**k=1;**

**else**

**k=0;**

**return k;**

**}**

**main ()**

**{int s,f,x,I;**

**s=0;**

**for(i=0 ; i<=9; i++)**

**{ cin>> x;**

**f=positive(x)**

**if( f= =1)**

**s=s+x;**

**}**

**cout<<"the sum "<<s;**

**return 0;**

**}**

**Write aprogram to enter A[5] in main program and use a function to square the element in the array and print the result array in main program**

**#include<iostream.h>**

**#include<math.h>**

**Int sqrarray(int a[5])**

**{ int i;**

**For(i=0 ;i<5 ;i++)**

**a[i] = squre(a[i],2) ;**

**Return 0 ;**

**}**

**عندما نقوم الدالة الرئيسية باستدعاء الدالة الفرعية عن طريق كلمة الاستدعاء sqrarray(a) سوف يتم ارسال المصفوفة التي تم ادخالها في الدالة الرئيسية الى الدالة الفرعية بعد ذلك يتم تنفيذ الدالة الفرعية حيث تقوم هذه الدالة بتربيع قيم المصفوفة المدخلة اي اصبحت لدينا مصفوفة بقيم جديدة وهي المربعة فعند اتتهاء تنفيذ الدالة الفرعية ورجوعها الى كلمة استدعائها فيى الدالة الرئيسية لتكمل تنفيذ الجمل التي تليها فان الدالة الرئيسية ستتعرف على المصفوفة الجديدة المربعة لذلك تقوم الدالة الرئيسية بطباعة المصفوفة المربعة ونلاحظ ان الدالة الرئيسية تعرفت على المصفوفة الجديدة بدون اخراجها بال return كما ذكرنا ذلك سابقا**

**Main() int**

**{ int i;**

**For(i=0; i<5 ;i++)**

**Cin >>a[i];**

**sqrarray(a);**

**For(i=0 ;i<5 ;i++)**

**Cout<<a[i] ;**

**Return 0;**

**}**

H.w

1- write a program that use function to find the sum of numbers in rang (12-24) and print the sum in main program .

2- write a program to enter integer array A[10] in main program and then use the function to find the average of it and print the average in main program .

3- write aprogram to enter integer array B[10] in main program and use two functions.

the first function use to find the minimum number and print it in main program

the second function use to find the maximum number and print it in main program.

4-write a program to enter 10 integer numbers and find the sum of even numbers only in main program. Ues the function to test if the number is even or not .

5- write a program to enter integer array A[10] in main program and then use a function to add 2 for every even number in it , and then print the result array in main program .

**موقع او تواجد المتغيرات في cope of variable**

سنوضح هنا كيف يتم تعريف المتغيرات من حيث موقغها وتواجدها في البرنامج

1**- المتغيرات الموضعية او المحلية Local Variable**

وهي المتغيرات التي يتم الاعلان عنها داخل الدالة الرئيسية main () او داخل الدوال الفرعية وتسمى بالموضعية او المحلية لانها تكون غير معروفة خارج الدالة المعرفة ضمنها , ومن الجدير بالذكر ان المتغير الموضعي او المحلي تنشا له قيمة وكيان عند البدء بتنفيذ الدالة التي ينتمي لها فقط وتنتهي هذه القيمة او تلغى عند الانتهاء من تنفيذ تللك الدالة 0

امثلة توضح استخدام المتغيرات الموضعية

اكتب برنامج يستخدم دالتين احدهما تقوم بايجاد مساحة المستطيل والاخرى ايجاد محيط المستطيل 0

#include<iostream .h>

int rec1( )

{int x, y ,t;

x=4;y=6 ;

t=x\*y ;

cout<< t

return 0;

}

int rec2( )

{ int x,y,t ;

x=4 ; y=6;

t=(x+y)\*2;

cout<<t;

return 0;

}

main()

{ rec1();

rec2();

return 0;

}

نلاحظ ان المتغيرات (x,y,t ) المستخدمة في الدا لة االاولى والثانية محلية اي خاصة بتللك الدالة اي ان x,y,t في الدالة الثانية هي ليست نفسها في الدالة الاولى لذلك ادخلناها مرة اخرى في الدالة الثانية لان المتغيرات المحلية بمجرد انتهاء تنفيذ الدالة تلغى القيم بداخلها 0

**المتغيرات العامة Global Variable**

ان هذه المتغيرات تكون قيمها معروفة في البرنامج من اوله الى اخره ويمكن استخدامها في اي جزء منه اي في اي دالة وكما تحتفظ بقيمها اثناء تنفيذ البرنامج 0 وتعرف هذه المتغيرات خارج حدود الدوال ومن المفضل تعريفها في بداية البرنامج اي بعد عبارات تحميل الفايلات الرئسيه 0

\*\*نفس المثال السابق الذي يستخدم دالتين الاولى ايجاد مساحة المستطيل والثانية محيطه ولكن باستخدام المتغيرلت العامة 0

#include<iostream.h>

int x,y

x,y هي متغيرات عامة فعرفت خارج حدود اي دالة فتكون معروفة في كل البرنامج فلا نحتاج الى تعريفها مرة اخرى عندما نستخدمها في اي دالة وتكون محتفضة بقيمها الحاليه

int rec1()

{int t;

x=4 ; y=6;

عند الخروج من الدالة تبقى قيمة الx هي4 وقيمة ال y هي 6 لانها متغيرات عامة تحتفظ بقيمها النهائية فعندما نستخدمها في اي مكان اخر في البرنامج تبقى على قيمتها التي تحتويها

t=x\* y;

cout<<t;

return0;

}

ال x هنا قيمتها ايضا 4 وال y هنا ايضا قيمته 6 لانها نفس ال x و ال y المستخدمة في الدالة الاولى لذلك لم نقم بادخالها مرة اخرة لانها متغيرات عامة تبقى محتفظة بقيمها النهائية

int rec2()

{int t ;

t=(x+y)\*2 ;

cout<< t;

return 0;

}

main()

{ rec1();

rec2() ;

return 0;

}

\*\*\*برنامج اخر يبين استخدام دالة لايجاد مساحة دائرة حيث نقوم بادخال نصف قطر الدائرة بالبرنامج الرئيسي ونستخدمه نفسه في الدالة بدون تعريفه مرة اخرى او ادخال قيمته مره اخرى0 بما ان نصف القطر لا نريد تعريفة مره اخرى في اي جزء من البرنامج ويكون محتفظ بقيمته اينما يستخدم يجب ان يعرف كعام Global) ) اي يكون البرنامج كالاتي 0

# قانون مساحة الدائرة هو النسبة الثابتة في نصف القطرتربيع

#include<iostream.h>

متغير عام

int rad ;

int areacricle( )

float area;

ال red هنا قيمتة 10 ولا نحتاج الى ادخال قيمته مرة اخر او الى تعريفه لانه معرف في البداية

area=3.14\*red \*red ;

cout<< "area="<< area <<"\n" ;

return 0

}

main()

لا نحتاج الى تعريفه لانه معرف في البداية

{red=10;

areacricle()

return 0;

}

**EX**: Writ a program to enter array A[5] in main program and then use two functions .the first function find the minimum number in it and the second function find the maximum number in same array .and print the minimum and maximum in main program .

#include<iostream.h>

int A[5];

int minfun()

{int i ,min ;

for(i=0 ; i<5 ; i++)

if( i==1)

min=A[i];

جملة ال if هي جملة واحدة داخل ال for

else

if (A[i]< min )

min=A[i];

return min ;

}

int maxfun()

{int i, max;

for(i=0 ; i<5 ; i++)

if( i==1)

max=A[i];

جملة ال if هي جملة واحدة داخل ال for

else

if (A[i]> max )

max=A[i];

return max ;

}

main ()

{int I ;

for(i=0; i< 5 ; i++)

{cout<<"enter the number"<<endl;

cin>> A[i] ;

}

minfun();

maxfun();

cout<<"the minimum no ="<<mimfun() <<"\n" ;

cout<<" the maximum no ="<<maxfun() <<"\n" ;

return 0 ;

}

نلاحظ ان المصفوفة تم تعريفها في البداية لذلك هي اصبحت عامة اي يمكن استخدامها في اي جزء في البرنامج بدون تعريفها مرة اخرى او ادخالها مرة اخرى لانها تبقى محتفظة بقيمها الحالية اثناء تنفيذ البرنامج فعندما ادخلنا المصفوفة في البرنامج الرئيسي واستدعينا الدالة الاولى فان الدالة الاولى وكذلك الثانية سوف تتعرف على هذه المصفوفة ومع قيمها لذلك لا نحتاج الى تعريف المصفوفة مرة ثانية او ادخال قيمها مرة اخرى في الدالتين لانها عامة

**EX:** Write a program to enter array a[5] in main program and then use function to square the element in it and print the array in main program

#include<iostream.h>

#include<math.h>

int a[5];

int powarray( )

{ int i ;

for(i=0 ;i< 5 ;i++)

a[i]= pow(a[i],2) ;

هنا لا يمكن ان نقول return(a[]) وذللك لان القيمة بعد ال return هي مصفوفة ونوع القيمة التي تنتجها الدالة هي int لذلك لايمكن اخراج مصفوفة لانه يتعارض مع نوع القيمة الموجودة في راس الدالة وهذا يحدث فقط مع المصفوفة لذلك البرنامج الرئيسي سوف يعرف المصفوفة بعد تربيع عناصرها وبدون الحاجة الى اخراجها لانها معرفة بشكل عام ) global) اوحتى اذ لم تكن معرفة بشكل عام ايضا يتعرف عليها كما ذكرنا سابقا

return 0;

}

main()

{int i, a[5];(

for(i=0 ; i<5 ; i++)

cin>> a[i];

powarray( );

for(i=0 ;i<5 ; i++)

طباعة المصفوفة التي تغيرت قيمها داخل الدالة وهنا سوف يعرفها البرنامج بدون اخراجها ب ال return

cout<<a[i];

return 0;

}

**EX:** Write a program to enter array a[5] in main program and then use two functions, the first function use to square the element in it and the second use to print the result array in it.

هنا سوف نقوم بادخال المصفوفة في البرنامج الرئيسي ثم نستدعي الدالة الاولى لكي تقوم بتربيع كل عنصر قي المصفوفة ثم تقوم الدالة الثانية بطباعة المصفوفة الناتجة بعد التربيع لذلك يجب ان تكون المصفوفة معروفة ومحتفظة بقيمها الجديده لكي تسطيع الدالة الثانية التعرف عليها وطباعتة قيمها المربعة الجديدةا لذا يجب ان تعرف كمصفوفة عامة (globle )

#include<iostream.h>

#include<math.h>

double a[5];

void power()

{ int i;

for(i=0 ; i< 5; i++)

a[i]=pow(a[i] ,2);

}

void printarray ();

{int i;

for(i=0 ; i<5 ; i++)

cout<<a[i ]<<"\n";

return 0;

}

main()

{ int i;

for(i=0 ; i< 5; i++)

cin>> a[i] ;

power() ;

هنا لم نقم بادخال المصفوفة في كلمة الاستدعاء لان المصفوفة ستكون معرفة ومحتفظة بقيمها اينما تستخدم بالبرناج لانها عامة لذلك الدالتين سوف تعرفها بدون الحاجة الى ادخالها لهما

printarray ();

return 0;

}

**الارسال بواسطة المرجع والقيمة Passed by value and reference**

في جميع البرامج السابقة كنا نمرر نسخ من البرامترات( arguments ) الى الدوال وفي هذه الحالة فان اي تغيير نجريه على هذه البارامترات في داخل الدوال سوف لن يؤثر على القبم الحقيقية للمتغيرات المرسلة في كلمة الاستدعاء داخل الدالة الرئيسية لاننا لم نرسل المتغيرات الحقيقية ولكننا ارسلنا نسخ عنها (وبمعنى اخر فان التغيير الذي يتم على المتغيرات داخل الدالة سوف يكون غير مرئي من قبل البرنامج الرئيسي) تسمى هذه الحالة التمرير بواسطة القيمة (**passed by value** ) اي اننا مررنا او ارسلنا قيم المتغير فقط وليس القيمة مع مرجعها

وفي بعض الاحيان نحتاج ان نجري تغيير على القيمة الاصلية للمتغير المرسل داخل الدالة بحيث ان هذا التغيير على قيم البارامتر داخل الدالة سيعود هذا التغيير نفسة الى المتغير الاصلي في كلمة الاستدعاء في الدالة الرئيسيه او في اي دالة اخرى وهكذا يمكن طباعتة والاستفادة منه بعد تغيره في الجزء الذي استدعي فيه وتسمى هذه الحالة التمرير بواسطة القيمة والمرجع ) **passed by value and reference**) اي ان التغيير على الباراميتر داخل الدالة سيرجع الى المتغير الاصلي اي بهذه الحالة تسطيع الدالة اخراج اكثر من قيمة بداخلها الى كلمة الاستدعاء اي الى المتغيرات الموجودة في كلمة الاستدعاء بالاضافة الى اسم الدالة في كلمة الاستدعاء عن طريق ال return 0 حيث تقوم الدالة بارسال القيم الجديدة التي طرءت على المتغيرات المرسلة لها الى نفس هذه المتغيرات في كلمة الاستدعاء عن طريق وضع علامة الجمع المنطقي & قبل اسم االبارامتر في اول سطر لدالة اي في سطر تعريف الدالة بعد النوع البياني للبارامتر اي كالاتي0

**type function- name (int & x,int & y)**

ستقوم علامة الجمع المنطقي & باخراج القيم الجديدة التي اصبحت في ال x وال y الى المتغيرات التي تقابلها في كلمة الاستدعاء بالاضافة الى قدرة الدالة باخراج القيمة الجديدة التي تكونت بداخلها الى اسم الدالة في كلمة الاستدعاء لكن عن طريق جملة ال return وبذلك تسطيع الدالة باخراج اكثر من قيمة الى كلمة الاستدعاء اما الى المتغيرات الموجودة في كلمة الاستدعاء عن طريق علامة & او الى اسم الدالة عن طريق جملة ال retune ونلاحظ ان ال return تاتي بعدها قيمة واحدة اي انها ترجع فقط قيمة واحدة الى كلمة الاستدعاء وتضعها في اسم الدالة لذلك عندما نريد اخراج اكثر من قيمة نستفاد من علامة & بالاضافة الى retune 0

مخطط توضيحي

راس الدالة (السطر التعريفي لدالة ) **int sum(int &x , int& y**)

كلمة الاستدعاء **sum ( a , b**) ان الاسهمين المتقابلين يعني ارسال واستقبال معنى ذلك ان كلمة الاستدعاء ترسل قيم ال a و ال b الى x وال y في الدالة ثم تقوم الدالة بارجاع القيم الجديدة ل x و y التي تكونت في داخل الدالة بعد الانتهاء من تنفيذ الدالة الى a و b في كلمة الاستدعاء لذا تلغى منها للقيم القديمة وتصبح بها القيم الجديدة التي تكونت داخل الدالة اي هنا الدالة قامت باخراج قيمتين الى خارجها ولكن ليس في اسمها وانما في المتغيرات الموجودة في الاقواس بعد اسم الدالة في كلمة الاستدعاء 0

**المثال التالي يوضح طريقة passed by reference**

هنا سوف نقوم بادخال ثلاث قيم الى الدالة من كلمة الاستدعاء في البرنامج الرئيسي ثم تقوم الدالة بمضاعفة هذه الارقام ثم طباعتها في البرنامج الرئيسي اي نحتاج الى اخراج ثلاث قيم بعد مضاعفتهم الى البرنامج الرئيسي لكي يتم طبعهم فهنا يجب استخدام علامة & لان ال return تقوم فقط باخراج قيمة واحدة الى اسم الدالة في كلمة الاستدعاء في البرنامج الرئيسي

#include<iostream.h>

void duplicate(int& a,int& b,int& c)

{ a\*=2;

b\*=2;

c\*=2;

}

int main()

{ int x,y, z;

x=1 ; y=3; z= 7;

duplicate(x,y,z);

cout<<"x="<< x <<"y="<< y<<"z="<< z;

return 0;

}

توضيح : هذا البرنامج يحتوي الدالة duplicate وضيفتها مضاعفة القيم التي ترسل اليها 0

الشي الجديد وهو اضافة علامة الجمع المنطقي في سطر تعريف الدالة(راس الدالة) بعد النوع البياني للبارامتر نفسه وهي دليل على ان المتغيرات a,b,c تم تمريرهم بواسطة المرجع

passed by reference

فعند استدعاء الدالة duplicate فان المتغيرات x وy وz ستعطى الى بارامترات الدالة والتي هي a و b و c على التوالي اي انه اصبح الان قيم ال a=1 و b=3 و c=7 وما قامت به الدالة هو مضاعفة قيم كل من a,b,c وعند انتهاء العمل في هذه الدالة تقوم العلامة & بالسماح بخروج قيمة a الجديدة والتي اصبحت تساوي 2 واسنادها اي وضعها في المتغير x في كلمة الاستدعاء في البرنامج الرئيسي وكذللك اعطاء قيمة b والتي اصبحت تساوي 9 الى المتغير y في كلمة الاستدعاء وكذلك قيمة c ستعود الى المتغير z اي ان قيم المتغيرات الموجودة في كلمة الاستدعاء ستتغير بعد استدعاء الدالة duplicate لان ارسالها تم بواسطة المرجع وفي حالة عدم وجود علامات & فان قيم البارامترات لن تعود الى المتغيرات في كلمةالاستدعاء في البرنامج الرئيسي وتكون نتيجة تنفيذ للبرنامج في هذه الحالة ان x=1,y=3,z=7

نلاحظ في البرنامج اعلاه ان الدالة نوعها void لانها لا تحتوي على جملة return لاننا اصلا لا نريد اخراج قيمة الى اسم الدالة في كلمة الاستدعاء في البرنامج الرئيسي0

**تغير للبرنامج السابق :** نفس البرنامج السابق لكن اذا طلب بالاضافة الى انه يتم طباعة القيم المدخلة لدالة بعد مضعافتها في البرنامج الرئيسي ايضا ان تقوم الدالة بجمع هذه القيم بعد مضاعفتها وطباعة المجموع ايضا في البرنامج الرئيسي

اي هنا نحتاج الى اخراج اربعة قيم من الدالة فاننا سوف نستخدم علامة & لاخراج القيم الثلاثة بعد مضاعفتها وممكن ان نستخدم ال retune لاخراج قيمة الجمع اي يكون البرنامج كالاتي

#include<iostream.h>

int duplicate(int& a,int& b,int& c)

{int s ;

a\*=2;

b\*=2;

c\*=2;

s=a+b+c;

return s

}

int main()

{ int x, y, z, k;

x=1 ; y=3; z= 7;

k= duplicate(x,y,z);

cout<<"x="<< x <<"y="<< y<<"z="<< z<<"\n";

cout<<"the sum ="<< k ;

return 0;

}

ممكن حل البرنامج اعلاه ايضا بدون استخدام return لاخراج قيمة الجمع وانما نستخدم فقط علامة & لاخراج كل القيم المراد اخراجها اي ان التغير على البرنامج اعلاه يكون كالاتي

int duplicate(int& a,int& b,int& c,int& s)

{

a\*=2;

b\*=2;

c\*=2;

s=a+b+c;

return 0;

}

int main()

{ int x,y,z, k;

x=1 ; y=3; z= 7;

duplicate(x,y,z,k);

cout<<"x="<< x <<"y="<< y<<"z="<< z<<"\n";

cout<<"the sum ="<< k ;

return 0;

}

نلاحظ انه اضفنا متغير k في كلمة الاستدعاء حتى تنتقل لة قيمة الجمع من المتغير s بعد الانتهاء من تنفيذ دالة duplicate لان k يقابل ال s وبهذه الحالة نستغني عن جملة ال return لاعادة قيمة الجمع

**EX:** Writa aprogram to enter the number in main program and then uses a function to find the number that come before it(previous ) and the number that come after it(next) and print them in main program

برنامج لادخال رقم في لبرنامج الرئيسي ثم ايجاد الرقم السابق والاحق باستخدام الدالة ثم طباعة الرفم السابق والاحق في البرنامج الرئيسي

بما اننا نريد اخراج قيمتين نستخدم علامة & لان ال return تسمح لنا باخراج قيمة واحدة

ممكن في هذا البرنامج اخراج القيمتين عن طريق علامة & اواخراج قيمة واحدة عن طريق & الى المتغير الذي يقابلها في كلمة الاستدعاء والاخرى عن طريق return الى اسم الدالة في كلمة الاستدعاء 0 ولكننا سوف نقوم في هذا البرنامج باخراج القيمتين عن طريق علامة & ولا نستخدم ال return

#include<iostream.h>

void prevnext(int x, int& prev, int& next)

{ prev=x-1;

next=x+1;

}

int main( )

{ int x,y,z;

x=100 ;

prevnext(x,y,z) ;

cout<<"previous ="<<y <<" , next =" <<z <<"\n" ;

return o;

}

نلاحظ ان ال x في كلمة الاستدعاء في البرنامج الرئيسي هو الرقم المرسل الى الدالة لايجاد الرقم السابق له واللاحق وان ال y, z سنضع بها مخرجات هذه الدالة التي هي الرقم السابق يوضع في y والرقم اللاحق يوضع في ال z

لذلك نلاحظ ان ال x في راس الدالة(تعريف الدالة) لا نحتاج الى تغيره وارجاعة الى البرنامج الرئيسي لذلك لم نضع له علامة & اي يكون فقط مدخل لدالة بينما الرقم السابق الممثل ب prev والرقم الاحق المثثل ب next سوف يتم ايجادهما في داخل الدالة ونحتاج اليهما خارج الدالة لذا وضعنا عند تعريف الدالة لهما علامات & قبلهما

هنا مادام نوع الدالة void لذلك عبارة ال retune لا تكتب 0

**Ex:** Write a program to enter array A[10] in main program and then call function to find the sum of even no and the sum of odd no . in it.and then print the their sum in main program .

هنا نحتاج الى ادخال المصفوفة في البرنامج الرئيسي ثم استخدام الدالة لايجاد قيمتين لجمع واخراجهم بعد انتهاء من تنفيذ الدالة الى البرنامج الرئيسي 0

#include <iostream.h>

void evenoddsum(int A[10],int& s1,int & s2)

{int i;

s1=0 ; s2=0;

for(i=0;i<9 ;i++)

if(A[i] %2==0)

s1=s1+A[i];

جملة الif تعتبر جملة واحدة داخل ال for

else

s2=s2+A[i];

}

int main()

{int A[10],i,k1,k2;

for(i=0;i<9 ;i++)

cin >>A[i];

evenoddsum(A,k1,k2);

cout<<"the sum of even numbers="<<k1<<"\n";

cout<<"the sum of odd numbers="<<k2<<"\n";

return 0;

}

H.W

1- write a program to enter array A[10] in main program and use two function , the first function to find the sum of positive numbers in it and print it in main program . the second function use to find the sum of negative numbers in the same array and print the sum also in main program.

2- write a program to enter array B[8] in main program and then use a function to find the sum and average of it and then print the sum and average in main program .

**-2الدوال المبنية built in function :**

سناخذ الدوال الرياضيةmathematical function كمثال عن الدوال المبنية ولكي تستخدم هذه الدوال في البرنامج وتنفذ يجب استدعاءاو تحميل الفايل الرائسي math.h في البرنامج . سنتطرق الان الى بعض الانواع الشائعة من الدوال الرياضية .

**1-abs(x**) – The function abs() returns the absolute value of integer parameter.

Ex: d= abs(-2 )

d= 2

**2- fabs(x**) : returns the absolute value of a floating point number

T=fabs(-2.4)

T= 2.4

**3- acos(x)**  -The function acos() returns the arc cosine of the argument. The range of the argument is from -1 to 1. Any argument outside this range will result in error.

Ex: d=acos(-0.707 )

d=2.35604

**4-asin(x)**- The function asin() returns the arc sine of the argument. The range of the argument is from -1 to 1. Any argument outside the range will result in error.

D= asin(-0.707)

D= -0.785247

**5- atan(x)** : returns the arc tan of the argument. The range of the argument is from -1 to 1. Any argument outside the range will result in error.

D= aatan(-0.707)

**6-cos(x)** – The function cos() returns the cosine of the argument. The value of the argument must be in radians.

اي ان قيمة ال x يجب ان تضرب في 3.14 ثم يقسم على 180 لكي تكون في ال radians

F= cos(30 \* 3.14/180)

**7-sin(x)**- The function sin() returns the sine of the argument. The value of the argument must be in radians .

اي ان قيمة ال x يجب ان تضرب في 3.14 ثم يقسم على 180 لكي تكون في ال radians

F=sin(30 \* 3.14/180)

**8-tan (x**)- The function tan() returns the tangent of the argument. The value of the argument must be in radians .

اي ان قيمة ال x يجب ان تضرب في 3.14 ثم يقسم على 180 لكي تكون في ال radians

F=tan(45\* 3.14/180)

**9-fmod (x/y)**- The function fmod() returns the remainder of floating point division of the two arguments. Y can not be zero

R=fmod(4.5/1.2)

**10- pow(x,y)**- calculates x to the power of y.

Ex:b= pow(3,2)

b=9

**11 – sqrt (x)**- The function sqrt() the square root of the argument. An error occurs if the value of the argument is negative.

Ex: A=sqrt(2)

A=1.41421

**12-ceil(x)** : rounds up to a whole number

Ex :T=ceil(11.2)

T=12

**13: floor(x**) : rounds down to a whole number

Ex: y=floor(11.5)

Y=11

البرنامج التالي يوضح استخدام بعض الدوال الرياضي

#include<iostream.h>

#include<math.h>

 int main()

{      double a=-0.707;

      double a1=(45.0\*3.14)/180;

      double a2=2;

      b=acos(a);

      cout << "The arc cosine of " << a << "= " << b << endl;

      b=asin(a);

      cout << "The arc sin of" << a << "= " << b << endl;

      b=atan(a);

      cout << "The arc tangent of  " << a << " = " << b << endl;

b=ceil(a);

      cout << "The ceiling of " << a << " = " << b << endl;

      cout << "The floor of  " << a << " = " << floor(a) << endl;

      cout << the power of " << a2 << " = "  << pow(a2,2) << endl;

     cout << "The square root of " << a2 << "  =  " << sqrt(a2) << endl;

cout << "The cosine of " << a1 << " = " << cos(a1) << endl;

      }

**The output of this program after running it are**

The arc cosine of -0.707= 2.35604

The arc sin of-0.707= -0.785247

The arc tangent of  -0.707 = -0.615409

The ceiling of -0.707 = 0

The floor of  -0.707 = -1

the power of 2 = 4

The square root of 2.0  = 1.414

The cosine of 45 degrees = 0.707

EX: write a program to find the sqrt root of the numbers between (1 -10)

#include<iostream.h>

#include<math.h>

Main()

{ int i ;

Cout<<sqrt(i) <<"\n" ;

}