تمرين 1 :

اكتب برنامج بلغة C + + يقوم بحساب المقدار $n! + m!$ حيث m, n عددان طبيعيان مدخلان ...وذلك باستخدام الدوال.

تمهيد :

سنقوم بتعريف دالة تقوم بحساب العاملي لعدد طبيعي ثم سنقوم بكتابة برنامج يقوم بحساب العاملي لقيمتين مدخلتين باستخدام دالة العاملي التي عرفناها سابقاً ومن ثم طباعة مجموعهما.

هناك أكثر من طريقة لكتابة الدالة المطلوبة هنا سنشرح طريقتان لكتابة دالة العاملي :

الطريقة الأولى :

قمنا بكتابتها بالاستعانة بحلقة for.

الطريقة الثانية :

قمنا بتعريف دالة أسميناها fact وعرفنا الوسيط الصحيح x داخلها ووضعنا شرط بأنه إذا كانت $x = 0$ فإن القيمة المعادة هي 1 ((نعلم أن $0! = 1$)) وإذا كان الشرط غير محقق فعندها يتم إعادة$x * fact\ (x - 1)$

وذلك اعتماداً على مبدأ كل دالة تستدعي دالة.

لنأخذ مثال يوضح الطريقة الثانية : لنفرض أن $x = 3$

فعندها يكون الشرط الذي وضعناه غير محقق فتكون القيمة المعادة هي $3 * fact\ (2)$

هنا نلاحظ أنه تم استدعاء الدالة fact من جديد من أجل القيمة 2

وبعد المرور بالشرط من جديد نلاحظ أنه سيتم إعادة $3 * 2 * fact\ (1)$

نلاحظ أنه تم استدعاء الدالة fact من جديد من أجل القيمة 1

وبعد المرور بالشرط من جديد نلاحظ أنه سيتم إعادة $3 * 2 * 1 * fact\ (0)$

هنا نلاحظ أنه تم استدعاء الدالة fact من جديد من أجل القيمة 0

وبعد المرور بالشرط من جديد (نلاحظ أنه محقق فيتم إعادة القيمة 1 من أجل القيمة 0)

نلاحظ أنه سيتم إعادة $3 * 2 * 1 * 1$ أي ستكون القيمة المعادة 6 وهي ناتج 3!

الحل :

```
#include <iostream.h>
long int fact (int x) {
long int r=1 ;
for (int i=2;i<=x;i++)
r=r*i;
return r;
}
```

((طريقة 1))

```
long int fact (int x) {
if (x==0)
return 1;
else
return (x*fact (x-1));
}
```

((طريقة 2))

```
int main () {
int n,m;
do {
cin>>n;
cin>>m;
} while (n<0||m<0);
long int y=fact (n);   // y=n!
long int z=fact (m);  // z=m!
cout<<y+z;
return 0;
}
```

الدالة لا تختبر الوسيط فيما إذا كان صحيح أم لا إنما البرنامج هو من يقوم بذلك.

وكما نلاحظ هنا أن الاختبار قد تم قبل استدعاء الدالة.

تمرين 2 :

اكتب صفاً بلغة C++ يقوم بحساب العاملي لعدد طبيعي , ثم استخدم هذا الصف في برنامج يقوم بحساب المقدار n!+m! حيث n,m عددان طبيعيان مدخلان.

الحل :

```
#include <iostream.h>
class Factorial {
public :
int x;
long int fact () {
if (x==0)
return 1;
else
return (x*fact (x-1));
}
Factorial (int a) {
x=a; }
};
int main () {
int n,m;
do {
cin>>n;
cin>>m;
```

يمكن ألا نضع وسطاء ويمكن أن نضع والسبب يعود إلى أن الوسيط معرف من قبل الدالة

نعتبر العدد المراد حساب العاملي له معطى بياني وكلما أردنا حساب عاملي لعدد ننشئ له object خاص به

باني لكي نبني object داخله

```
} while (n<0||m<0);
Factorial a(n);
Factorial b(m);
cout<<a.fact ()+b.fact ();
return 0;
}
```

لو لم نضع باني في الصف السابق فكيف ستصبح آلية الحل ؟

سيصبح الحل كما يلي :

```
#include <iostream.h>
class Factorial {
public :
int x;
long int fact () {
if (x==0)
return 1;
else
return (x*fact (x-1));
}
```

⟶

```
};
int main () {
int n,m;
do {
```

```
Factorial () {
x=0;
}
```

هذا هو الباني الافتراضي الذي تقوم ببنائه الدالة ولكن لا يمكننا رؤيته وهذا الباني مهمته الوحيدة هي حجز مكان في الذاكرة

```
cin>>n>>m;
} while (n<0||m<0);
Factorial a ( );
Factorial b ( );
a.x=n;
b.x=m;
cout<<a.fact()+b.fact();
return 0;
}
```

الوسيط الافتراضي هنا هو 0 لأن لم ننشأ باني فكما ذكرنا سابقاً تم إنشاء باني افتراضي

أصبحت قيمة x داخل الـ a,b objects هي 0 ولكن هذه القيمة لا نريدها

لذا أسندنا القيمة n للـ x داخل a

والقيمة m للـ x داخل b

تمرين 3 :

اكتب برنامج بلغة C++ يقوم بحساب المقدار $x^n + y^n$ حيث x, y, m, n أعداد طبيعية مُدخلة وذلك باستخدام الدوال .

الحل :

```
#include <iostream.h>
long int power (int a,int b) {
long int r=1;
for (int i=1;i<=b;i++)
r=r*a;
return r;
}
int main ( ) {
int x,y,m,n;
do {
```

b تمثل الأس و a تمثل الأساس

طريقة ((1))

```
if (b==0) return 1;
else
return (a*power(a,b-1));
}
```

طريقة ((2))

شرح الطريقة الثانية :

نعلم أن $a^b = a * a^{b-1}$

ومنه $power(a,b) = a * power(a,b-1)$

وهي نفس فكرة التمرين (كل دالة تستدعي دالة)

```
cin>>x>>n>>y>>m;

}

while (x<0||n<0||y<0||m<0);
long int z=power (x,n);

long int t=power (y,m);

cout<<z+t;

return 0;

}
```

تمرين 4 :

اكتب صفاً يقوم بحساب القوة لعددين طبيعيين مُدخلين , ثم باستخدام الصف السابق اكتب برنامج بلغة C++ يقوم بحساب المقدار $x^n + y^n$ حيث x, y, m, n أعدد طبيعية مُدخلة وذلك باستخدام الدوال .

الحل :

```
#include <iostream.h>

class Power {

public:

int a;

int b;

Power (int x,int y) {a=x,b=y;}

long P(){

long int r=1;

for (int i=1;i<=b;i++)

r*=a;  //r=r*a
```

```
return r;
}
};
int main () {
int x,n,y,m;
cin>>x>>n>>y>>m;
while (x<0||n<0||y<0||m<0);
Power H(x,n);
Power R(y,m);
cout<<H.P()+R.P();
return 0;
}
```

تمرين 5 :

اذكر مخرجات (نتائج) تنفيذ البرنامج التالي :

```
#include <iostream.h>
int main () {
int x=7; int y=15;
int z=x+y;
return 0;
}
```

الحل : النتيجة لا شيء لعدم وجود تعليمة cout فقط عرفنا متحولات ولم نخرجها.

تمرين 6 :

اذكر نتائج تنفيذ المقطع البرمجي التالي :

```
int r=0;
for (int i=11;i<1111;i++);
{r=r+i;
} cout<<r;
```

نلاحظ وجود ; مباشرة بعد for أي لا يوجد تعليمات داخل حلقة for

الحل :

سيتم طباعة 1111 وذلك لأن أخر قيمة يأخذها i هي 1111 ومن ثم يجدها لا تحقق الشرط فيتم الخروج من الحلقة.

ملاحظة هامة :

<< عندما نكتب صفاً فإننا نعرف نوع جديد ولكي نستخدم متحولات من نوع object يجب علينا أن نكتب باني للصف , وإذا لم نكتب باني فإن البرنامج يقوم بشكل أوتوماتيكي ببناء باني افتراضي >>

تمرين 7 :

اكتب مخرجات البرنامج التالي :

```
#include <iostream.h>
int main () {
int x=11.11;
double y=11;
cout<<x+y;
return 0;
}
```

نلاحظ هنا أننا أسندنا 11.11 لـ x بالرغم أنه متحول صحيح وبرغم ذلك لا نواجه خطأ عند تنفيذ البرنامج والسبب أن المترجم يرى فقط القسم الصحيح ويتجاهل ما بعد الفاصلة

الحل :

نتيجة التنفيذ هي 22

تمرين 8 :

اكتب مخرجات البرنامج التالي :

```
#include <iostream.h>
int main () {
int r=1;
for (int j=1;j<11;j++);
r=r+(r/2);
cout<<r;
return 0;
}
```

نلاحظ وجود ; مباشرة بعد for أي لا يوجد تعليمات داخل حلقة for

إن ناتج هذه التعليمة هو 3/2=1.5 ولكن r عرفناها كعدد صحيح فيتم أخذ القسم الصحيح (1) وإهمال ما بعد الفاصلة

نتيجة التنفيذ هي 1

تمرين 9 :

عند ترجمة البرنامج التالي نحصل على عدة أخطاء اذكرها مع التعليل :

```
#include <iostream.h>
int main {
int x=3.14
double y=x+'x';
cout<<x+y;
retnru 0;
}
```

الخطأ الأول : نقص ( )

الخطأ الثاني : نقص الفاصلة

الخطأ الثالث : لا يمكن جمع عدد مع محرف

الخطأ الرابع : خطأ في كتابة return

ملاحظات امتحانية :

- أي حل صحيح بطريقة غير مطلوبة هو حل مرفوض في الامتحان.
- عندما يُطلب منا تعريف ما في الامتحان فإنه ليس بالضرورة كتابته حرفياً بل يكفي كتابته بشكل مختصر ووافي.

فمثلاً لو طلب كتابة تعريف الصف فإنه يكفي قول هو آلية برمجية تسمح بتعريف نوع جديد.

نموذج أسئلة الامتحانات لمقرر البرمجة والخوارزميات

1) عرف كلاً مما يلي

2) اكتب برنامج برنامج يقوم بـ ..... باستخدام الطريقة .....

3) اكتب صفاً يقوم بـ ..... ثم استخدم هذا الصف بكتابة برنامج يقوم بـ .....

4) ما نتائج تنفيذ المقطع / البرنامج .....أو ما الأخطاء.....

تم بعونه تعالى كتابة ملحق لمقرر البرمجة والخوارزميات يحوي حل التمارين غير المحلولة في المحاضرات يمكنكم الحصول عليه من مكتبة خبراء الطباعة أو تحميله من موقعنا www.syriamath.net

...انتهت المحاضرة (11) ...