بسم الله الرحمن الرحيم

Object Oriented Programming

البرمجة الشيئية

عدد الساعات: ٢ نظري + ٢ عملي

المتطلبات السابقة: ٢١١حسب

أستاذة/المادة: م. نجلاء حسن

Lecture 5

-الكائنات الثابتة والدوال الثابتة
-استخدام مصفوفة من الكائنات
– الدوال الخطية

المعاملات الافتراضية

- قيم افتراضية تعطى للمعاملات في حال استدعاء الدالة مع عدم تمرير قيم للمعاملات فان المترجم ينظر لقيم المعاملات الافتراضية.
- توضع في جملة الاعلان لا التعريف.
- تعطى قيما من اليمين لليسار الافضلية لليمين.

```
Class  account
{
    public:
    double balance ,rate;
    account(double bal=0.0 , double r=0.5);
    ....
};
account::account(double bal , double r)
{  balance=bal ;    rate=r ;   }
```

مثال:

```
account c1();            // account c1(0.0 , 0.5);
account  c1(500.0);    // account  c1(500.0 , 0.5);
```

الدوال الأعضاء الثابتة:

- يمكن جعل دالة عضو بأحد التصنيفات ثابتة إذا كنا نريد ضمان عدم تعديل تلك الدالة لأي من عناصر البيانات المعرفة بهذا الصنف.
- يتم تعريف الدالة العضو الثابتة بوضع الكلمة الأساسية const بعد الاعلان عن الدالة مباشرة وقبل جسم الدالة.
- في حالة تعريف الدالة خارج حدود الصنف، يجب استخدام كلمة const اثناء الاعلان واثناء التعريف.

مثال

```
void show( ) const;
};
void rectangle:: show( ) const
{ cout<<x<<"\t"<<y<<endl; }
```

الكائنات الثابتةConstant Objects

- يمكن جعل كائن تابع لصنف ما ثابتًا إذا كنا نريد ضمان عدم تغيير البيانات في الكائن ، وهذا يعني امكانية استخدام الدوال الأعضاء ذات النوع const فقط مع هذا الكائن.(لماذا؟؟؟: لأن هذه الدوال لاتقوم بتعديل عناصر بيانات الكائن)

مثال

لنفرض أننا نريد انشاء كائن يدعى rec1(10,2) سيكون من الجيد ضمان عدم تغيير قيمة هذا الكائن لتحقيق هذا نكتب العبارة:

const rectangle rec1(10,2);

المصفوفات Array:

تذكير

جاءت المصفوفات لحل مشكلة الحاجة لإدخال عدد كبير من البيانات، فبدلاً من استخدام عدد كبير من المتغيرات لحفظ البيانات يتم استخدام المصفوفة التي تستطيع الاحتفاظ بالبيانات كمتغير واحد.

تعريفها:
مجموعة من المواقع المتجاورة في الذاكرة ولها نفس نوع البيانات وتستخدم لخزن البيانات.

	x
3	int
2	int
1	int
0	int
index	

أنواع المصفوفات Array types:

١. مصفوفات أحادية البعد Single Dimensional.
٢. مصفوفات متعددة الأبعاد Multi Dimensional.

١. مصفوفة أحادية:

```
Data_Type Array_name [ Array_Size ];
```

مثال:

```
int x[5];
```

إدخال البيانات للمصفوفة (الإسناد) [1]:

تذكير

الطريقة الأولى:

تعريف مصفوفة وإسناد كل بياناتها في نفس الوقت:

```
1. int x[5] = { 1, 7, 10, 2, 5};
```

بقية الخانات ستكون صفرية:

```
2. int y[10] = {3, 5};
```

كل الخانات ستكون أصفار:

```
3. int z[4] = {0};
```

الطريقة الثانية:

تعريف مصفوفة:

```
1. int a[3];
```

إسناد قيمة للخانة الأولى في المصفوفة:

```
2. a[0] = 1;
```

إسناد قيمة للخانة الثانية في المصفوفة:

```
3. a[1] = 7;
```

إسناد قيمة للخانة الثالثة في المصفوفة:

```
4. a[2] = 10;
```

10
7
1

تذكير

ويمكن إدخال البيانات إلى المصفوفة أثناء تشغيل البرنامج عن طريق (cin) :

إدخال جميع قيم المصفوفة باستخدام دالة For

```
1. int a[2];
2. for ( int i=0; i<=2; i++)
3. {
4.     cin >> a[i];
5. }
```

وبالمثل عملية الإخراج:

طباعة جميع قيم المصفوفة باستخدام دالة For

```
1. for ( int i=0; i<=2; i++)
2. {
3.     cout << a[i] << endl;
4. }
```

تذكير

المصفوفة متعددة الأبعاد:

- ثنائية البعد:

```
Data_Type Array_name [ x ][ y ];
```

- ثلاثية البعد:

```
Data_Type Array_name [ x ][ y ][ z ];
```

- رباعية البعد:

```
Data_Type Array_name name [ x ][ y ][ z ][ t ];
```

وهكذا يمكن إضافة أبعاد بحسب الحاجة.

مثال:

```
char ary[3][3] = {{'A','B','C'},{'D','E','F'},{'G','H','I'}};
```

ملء عناصر المصفوفة ذات البعدين

```
For ( int i=0 ; i<2 ; i++)
    for (int j=0 ; j<3 ; j++)
        cin> >x[i][j];
```

استخدام مصفوفة من الكائنات

- يتم الاعلان عن مصفوفة كائنات بنفس طريقة الاعلان عن مصفوفة أعداد صحيحة.

```
int a1, a2, a3=7, a[10];
   Employee e1,  e3(424, “mona”, “Cairo”,  ‘f’, 2000, 500, 300), e[10];
```

- يتم آلياً استدعاء دالة البناء التي تنشئ كائن فارغ عشرة مرات عند الاعلان عن مصفوفة كائنات مثل e[10] وذلك لانشاء مصفوفة من عشر كائنات من نوع الطبقة Employee.
- مفيدة عند الرغبة في معاملة عدد كبير من الكائنات التابعة لنفس الطبقة بطريقة مماثلة.(تتم بوضع الكائنات في مصفوفة واستعمال حلقة)

تابع:استخدام مصفوفة من الكائنات

- **ادخال بيانات كائن في المصفوفة:**

```
for (int i=0; i<n ;i++)
{
 cin >>e[i].idnumber>> e[i].address>> e[i].Gender;
}
```

- **استخراج بيانات كائن في المصفوفة:**

```
 cout<<e[i].idnumber<< e[i].address<< e[i].Gende;
```

مثال:

صممي برنامج بلغة C++ يحتوى على:

-صنف Eemployeeيحتوى علي :

الأعضاء البيانية العامة:رقم الموظفno ، الاسم name

الأعضاء الدالية : دالة البناء، دالة الهدم ، دالة compute يمرر اليها أساسي الراتبA ,العلاوات Bوالخصومات Cللموظف لتعود بالراتب الشهري

-في الدالة الرئيسية اشتقي مصفوفة كائنات E[6] من نوع Employee ثم املئيها بالبيانات , واحسبي الراتب الشهري لكل موظف. و طباعتها

```
# include<iostream.h>
#include<string.h>
class Employee
{
public:
 int no;
  char name[50];
Employee (){}
~Employee (){}
 float compute(float A,
float B, float C)
  {
        return (A+B-C);
  }
};
```

```
int main()
        {
        //const int n=5;
        Employee  e [3];
        float a,b,c;
        for (int i=0; i<3 ;i++)
        {
  cout<<"\n enter number&name\n";
        cin>>e[i].no>>e[i].name;
        cout<<"\n enter three number";
        cin>>a>>b>>c;

cout<<"\n"<<e[i].no<<e[i].name<<e[i].
compute(a,b,c);
        }
        return 0;
        }
```

مثال:

صممي برنامج بلغة C++ يحتوى على:

-صنف Distance الذي يمثل المسافة بالنظام الانجليزي في صورة قدم feet وبوصة inches.

يقوم البرنامج باستقبال المسافات (الكائنات)من المستخدم وبعد كل مسافة يتم ادخالها يسأل المستخدم اذا كان يرغب في ادخال المزيد من المسافات أم لا وبعد ذلك يتم طباعة جميع المسافات؟؟؟

دوال السطر القصيرة Inline Functions

- دالة السطر هي دالة تقليدية عادية أو عضو في صنف معين يتم احلال تعريفها (جسم الدالة) بالكامل في أي سطر من البرنامج يوجد به استدعاء لتلك الدالة مع العلم بأن وظيفة ومفهوم وتركيب ونتيجة الدالة لا تتأثر بكونها دالة سطر.

مفهوم دالة السطر

- عندما تكون الدالة قصيرة فان تنفيذ الأوامر والجمل الموجودة في جسم الدالة يستغرق زمن أقل من الزمن اللازم لاستدعاءها والرجوع بالنتيجة.
- تحديد دالة على أنها Inline لا تضيف عبئاً اضافياً على المبرمج ولكنها فقط عبارة نصيحة من المبرمج الى المترجم عن طريق وضع العامل "inline" قبل اسم الدالة.
- استخدام دوال السطر مع الدوال الأعضاء في صنف معين مهم جداً لأن معظمها تتكون من عدد قليل من السطور.

أنواع دالة السطر

- **دالة سطر ضمنية(implicit inline function):**

تتكون من سطر واحد ولا يذكر المبرمج صراحة أنها دالة سطر (لا يكتب الكلمة inline امام اسم الدالة).يعتبرها المترجم دالة سطر.

- **دالة سطر صريحة (explicit inline function):**

تتكون من سطر واحد أو عدة سطور ويذكر المبرمج صراحة أنها دالة سطر (يكتب الكلمة inline امام اسم الدالة).لا بد أن يعالجها المترجم كدالة سطر.

دالة السطر التقليدية

- يمكن أن توجد في الشكلين السابقين الذكر.

دالة سطر ضمنية

```
 int  max(int a,int b)
{
   if(a>b)return a  else return b;
}

inline float computezakah(float M,int nesba,int months)
{
float z;
If(m>nesba&& months>12)
z=2.5*m/100;
Else
    z=0;
return z;
}
```

دالة السطر العضو في صنف

نفس الاسلوب المتبع مع دالة السطر التقليدية

```
class mango_box {
    public:
          void eat_mango(int m);
           .....
};
inline void  mango_box :: eat_mango(int m)
{
.....  }
```

كل دالة معرفة داخل الطبقة تكون inline حتى لو لم نعلن عنها بـ inline

ملخص

قاعدة:

الدوال الأعضاء السطرية inline function هي المعرفة داخل كتلة الصنف ؛ اما الدوال الأعضاء غير السطرية non-inline function فهي معرفة خارج الصنف ومصرحة داخل كتلة الصنف

ولإجبار المترجم على التعامل مع أي دالة عضو على أنها دالة سطرية معرفة خارج كتلة الصنف فبإمكانك كتابة الكلمة المفتاحية inline ضمن تعريف الدالة.

تدريب2 :

اكتبي برنامج باستخدام الclass لحساب زكاة المال علما بأن زكاة المال تساوي 2.5% من المبلغ اذا بلغ النصاب ومر عليه سنه هجرية؟

z=(2.5*mony)/100

- حددي الأعضاء البيانية (عامه)والدالية
- اشتقي مصفوفة كائنات ;[4]z