

# تعلم لغة سي++ مع روح سامية : الدرس العشرون

بمعنا : المعالجة الأولية و نتعلم في هذا الدرس ما يلي :

١. معنى المعالجة الأولية.
٢. أمر تضمن.
٣. أمر تعريف و إلغاء تعريف.
٤. أمر إذا تم تعريف و إذا لم يتم تعريف.

البرنامج :

افتح برنامج code::blocks ثم أنشئ ملفين جديدين و احفظهما أولا في نفس المجلد ، احفظ الأول باسم "person\_class.hpp" و الثاني باسم "preprocessor-directives.cpp" و تأكد مرة أخرى أنهما في نفس المجلد ، سنبدأ بالكتابة في الملف "person\_class.hpp" أولا :

```
/*=====
    فئة الشخص
=====*/

#ifndef PERSON_CLASS    //لم يتم تعريف فئة الشخص ؟
#define PERSON_CLASS    //عرف ثابتا يحدد فئة الشخص

#include <iostream>
using namespace std;

//فئة الشخص
class Person {
public :
    //طباعة معلومات الشخص
    void print_info() {
        cout << "Hi. I am " << name
            << " and I am " << age << " years old" << endl;
    }
    //دالة البناء
    Person (string n, int a) {
        name=n;
        age=a;
    }
private :
    string name;    //الاسم
    int age;        //العمر
};
#endif    //PERSON_CLASS
```

لنكتب في الملف الثاني "preprocessor-directives.cpp" ما يلي :

```
/*=====
    المعالجة الأولية
=====*/
/*=====*/
#include <iostream>
#include "class.hpp"
using namespace std;
/*=====*/

//الدالة الرئيسية
int main() {
```

```

Person p("ro7 samiah", 15);
p.print_info();
cout << endl << endl << endl;
return 0; //إنهاء البرنامج
}

```

قم بتشغيل البرنامج بزر (F9) من لوحة المفاتيح أو من القوائم العلوية بالضغط على خيار build→build and run ، سيطبع البرنامج على الشاشة باللغة الانجليزية عبارة "مرحبا ، أنا روح سامية و عمري ١٥ سنة" ، بعدها عبارات أخرى ، برنامج code::blocks هو الذي وضعها فلا تفكر فيها كثيرا ، ستجدها بعد نهاية كل برنامج تكتبه و تشغله بـ code::blocks.

Hi, I am ro7 samiah and I am 15 years old

المخرج

الشرح :

يقوم برنامج الدرس بما يلي :

- ١- يعلن متغيراً من نوع Person معرفه p و يعطي دالة البناء الاسم "ro7 samiah" و العمر ١٥ .
  - ٢- يستدعي دالة print\_info من الكائن p.
  - ٣- يطبع ثلاثة أسطر ليفصل ما يطبعه برنامج الدرس عما يطبعه برنامج Code::Blocks بعد انتهاء البرنامج.
  - ٤- ينهي البرنامج.
- لننظر إلى البرنامج سطرا سطرا و لنبدأ بالملف "person\_class.hpp" :
- ١-

```

/*=====
    فئة الشخص
=====*/

```

تعليق يبين أن هذا الملف يحوي فئة الشخص.

٢-

لم يتم تعريف فئة الشخص ؟//  
**#ifndef PERSON\_CLASS** ، يكون شرط العبارة صائبا إذا تم تعريف الثابت **PERSON\_CLASS** قبل هذا السطر بأمر **التعريف defined** ، أي أن شرط العبارة صائب إذا تم قبل هذا السطر كتابة السطر :

**#define PERSON\_CLASS**

٣-

عرف ثابتا يحدد فئة الشخص//  
**#define PERSON\_CLASS**

**تعريف الثابت PERSON\_CLASS.**

-٤

```
#include <iostream>
```

تضمين الملف iostream في البرنامج ، الملف iostream يحوي إعلان أوامر الإدخال و الإخراج  
cin و cout بالإضافة إلى إعلان نوع سلسلة الرموز string.

-٥

```
using namespace std;
```

استخدام مساحة الاسم std ، مساحة الاسم تمثل نطاقا يحوي متغيرات و باستخدامه تصبح هذه  
المتغيرات مثل أي متغيرات عمومية في البرنامج.

-٦

```
// فئة الشخص
```

```
class Person {
```

بداية تعريف فئة الشخص و تحوي اسم و عمر الشخص.

-٧

```
public :
```

```
// طباعة معلومات الشخص
```

```
void print_info() {
```

```
    cout << "Hi. I am " << name
```

```
        << " and I am " << age << " years old" << endl;
```

```
}
```

```
// دالة البناء
```

```
Person (string n, int a) {
```

```
    name=n;
```

```
    age=a;
```

```
}
```

تعريف الأعضاء العاميين ، عرفنا دالة طباعة معلومات الشخص و دالة البناء التي تبتدئ اسم و عمر  
الشخص.

-٨

```
private :
```

```
    string name; // الاسم
```

```
    int age; // العمر
```

إعلان الأعضاء الخاصيين ، أعلننا اسم و عمر الشخص.

-٩

```
};
```

نهاية تعريف فئة الشخص.

-١٠

```
#endif //PERSON_CLASS
```

إنهاء أمر إذا لم يتم تعريف الذي كتبناه في بداية الملف.

لننظر إلى الملف "preprocessor-directives.cpp" :

-١-

```
/*=====
المعالجة الأولية
=====*/
```

كما فعلنا في الدروس السابقة فهذا تعليق متعدد الأسطر كتبنا فيه عنوان الدرس لا أكثر.

-٢-

```
/*=====*/
#include <iostream>
```

تضمين الملف iostream في البرنامج.

-٣-

```
#include "class.hpp"
```

تضمين الملف class.hpp في البرنامج ، لاحظ أن الملف class.hpp هو الملف الأول الذي كتبناه.

-٤-

```
using namespace std;
/*=====*/
```

استخدام مساحة الاسم std ، مساحة الاسم تمثل نطاقا يحوي متغيرات و باستخدامه تصبح هذه المتغيرات مثل أي متغيرات عمومية في البرنامج.

-٥-

```
//الدالة الرئيسية
int main() {
```

بداية الدالة الرئيسية مع تعليق يبين ذلك.

-٦-

```
Person p("ro7 samiah", 15);
```

إعلان كائن من نوع Person معرفه p، أعطينا دالة البناء العبارة "ro7 samiah" و الرقم ١٥ ، لاحظ أننا عرفنا فئة Person في الملف class.hpp و ليس في هذا الملف.

-٧-

```
p.print_info();
```

استدعاء دالة طباعة البيانات من الكائن p.

-٨-

```
cout << endl << endl << endl;
return 0; //إنهاء البرنامج
}
```

قمنا بطباعة ثلاثة أسطر لفصل ما طبع برنامج الدرس عما يطبعه برنامج Code::Blocks في نهاية كل برنامج يتم تشغيله ، بعد ذلك أنهينا البرنامج بأمر return 0 و أغلقنا قوس الدالة الرئيسية.

## معنى المعالجة الأولية :

المعالجة الأولية هي أوامر نقوم بكتابتها يتم تنفيذها قبل تصريف البرنامج لتغيير شيء في شكله كما سنلاحظ إذا عاينا أوامر المعالجة الأولية ، كل أوامر المعالجة الأولية تكون في سطر لوحدها و يتم بدء هذا السطر برمز **المربع #** ، لا تتأثر أوامر المعالجة الأولية بالنطاق فيمكن حتى أن نجعلها تتداخل مع الأقواس.

### أمر تضمين :

أمر **تضمن** هو أمر للمعالجة الأولية يقوم بإضافة محتويات ملف بلغة سي++ في المكان الذي نكتبه فيه ، أي أننا إذا وضعناه و حددنا الملف الذي يتم **تضمينه** في مكان ما في البرنامج فكأننا قمنا بنسخ محتويات الملف و لصقها مكان الأمر ، لاستخدام أمر **تضمن** نكتب في سطر الأمر رمز **المربع #** بعده كلمة **include** بعدها اسم الملف الذي نريد **تضمينه** و يكون بين قوسين زاويين < > أو بين علامتي اقتباس مزدوجتين " " :

```
#include <file>
```

أو

```
#include "file"
```

يسمى الملف الذي يتم **تضمينه** ملفا رأسيا ، تلاحظ أننا في كل البرامج التي كتبناها في الدروس السابقة كتبنا في بدايتها الأمر :

```
#include <iostream>
```

**iostream** هو ملف رأسي قياسي يُستخدم في كل برامج سي++ التي تقوم بالإدخال و الإخراج و هو موجود في مجلد المخصص للملفات الرأسية و يتم إعداد المصرف لمعرفة هذا المجلد ، داخل الملف **iostream** نجد كل التعريفات المخصصة بنوع سلسلة الرموز **string** و أوامر الإدخال و الإخراج **cin** و **cout** و غير ذلك مما يتعلق بالإدخال و الإخراج لهذا تلاحظ أننا استخدمنا الأوامر **cin** و **cout** مع أننا لم نقوم ببرمجتها بأنفسنا ، أثر **تضميننا** لهذا الملف هو نفسه أثر نسخ محتويات الملف و لصقها مكان أمر **التضمين** ، هناك فرق بين الطريقتين في **تضمين** ملف ، الطريقة الأولى التي نستخدم فيها قوسين زاويين < > يبحث فيها المصرف عن الملف الذي حددنا اسمه في المجلدات المخصصة للملفات الرأسية و يكون هذا المجلد محفوظا في إعدادات المصرف ، هذا هو الحال مع ما فعله في كل البرامج من **تضمين** الملف **iostream** بأن نكتب في بداية البرنامج :

```
#include <iostream>
```

في الحالة الثانية حين نقوم **بتضمين** ملف بين علامتي اقتباس مزدوجتين " " يقوم المصرف بالبحث عن الملف في المجلد الذي يحوي الملف الذي كتبنا فيه أمر **التضمين** ، إذا لم يجد المصرف الملف يبحث بعدها في المجلد المخصص للملفات الرأسية و تم إعداد المصرف عليه ، استخدمنا هذه الطريقة في هذا الدرس حين كتبنا في الملف **main.cpp** :

```
#include "class.hpp"
```

هنا يقوم المصرف بتضمين الملف **class.hpp** ، ينظر المصرف أولا في نفس المجلد الذي يحوي الملف **main.cpp** ، لهذا السبب وضعنا الملف **class.hpp** في نفس المجلد مع **main.cpp** ، تلاحظ أننا استخدمنا

**الفئة** التي عرّفناها في الملف class.hpp بشكل عادي و كأننا قمنا بتعريف **الفئة** في الملف main.cpp ، هذا لأننا قمنا بتضمين الملف الذي يحوي **الفئة** أي و كأننا نسخنا محتويات الملف و وضعناها مكان أمر **التضمين** ، إذا لم يجد المصرف الملف المحدد فإنه يظهر خطأ و لا يقوم بتصريف البرنامج.

### أمر تعريف و إلغاء تعريف :

أمر **تعريف** يقوم بتعريف ثابت و يعطيه قيمة إن تم كتابة قيمة له و إلا يتركه دون قيمة ، لتعريف ثابت نقوم بكتابة رمز **المربع** # بعده كلمة **define** بعدها معرف الثابت ، نستطيع بعد معرف الثابت أن نكتب قيمة للثابت و نستطيع أن نترك الثابت دون قيمة :

```
#define IDENTIFIER value
```

تعلمنا هذا الأمر في درس الثوابت ، بعد **تعريف** الثابت فإن المصرف إذا وجد معرف الثابت فإنه يقوم باستبدال المعرف الذي حددناه في **التعريف** بالقيمة التي حددناها و إذا لم نحدد قيمة يقوم بحذف المعرف و وجود المعرف و عدمه لا يؤثر في البرنامج ، بالإضافة إلى أمر **التعريف** فهناك أمر **لإلغاء التعريف** و صيغته هي رمز **المربع** # بعده كلمة **undef** بعدها معرف الثابت الذي عرفناه مسبقا :

```
#undef IDENTIFIER
```

بعد **إلغاء التعريف** يتم **إلغاء** الثابت الذي تم تعريفه و لا يمكن استخدامه في الأسطر التي تلي أمر **إلغاء التعريف** إلا بعد **تعريفه** مرة أخرى بأمر **تعريف** آخر ، لا تتأثر الثوابت التي تم تعريفها بالنطاق بل يمكن استخدامها بغض النظر عن النطاق الذي تم تعريفها فيه ، في درسنا استخدمنا أمر **التعريف** في الملف class.hpp حين كتبنا في السطر السادس :

```
#define PERSON_CLASS
```

تلاحظ أننا عرفنا الثابت PERSON\_CLASS بدون أي قيمة ، قد تتساءل عن فائدة **تعريف** ثابت بدون قيمة ، سنجد الإجابة عن هذا حين نتعلم الأوامر الشرطية إذا تم تعريف و إذا لم يتم تعريف ، بالإضافة إلى تعريف الثوابت يمكن أن نعرف أوامر على شكل دوال تأخذ مدخلات بأن نضع المدخلات بين أقواس كما نفعل مع الدوال ، مثلاً نستطيع أن نعرف عملية حساب العدد الأعلى بين عددين كما يلي :

```
#define getmax(a,b) ( (a) > (b) ? (a) : (b) )
```

بهذا نستطيع أن نستخدم الأمر getmax الذي عرفناه و نضع بعده بين قوسين أي قيمتين نود إدخالهما و يتم استبدال كل معرف من المدخلات a و b بالقيمة التي نضعها مكانه تماماً كما يحدث في الدوال فإذا كتبنا :

```
int x=4;
cout << getmax(x,9);
```

### أمر إذا تم تعريف و إذا لم يتم تعريف :

باستخدام أمر إذا تم تعريف و إذا لم يتم تعريف يمكننا من إضافة أو تجاهل جزء من البرنامج في شروط معينة ، في بداية الملف "class.hpp" كتبنا :

```
#ifndef PERSON_CLASS
```

معنى هذا السطر هو إذا لم يتم تعريف PERSON\_CLASS ، يقرأ المصرف الأسطر بعد هذا السطر إذا لم يكن الثابت PERSON\_CLASS معرفاً ، في غير ذلك يتم تجاهل الأسطر التالية و كأننا لم نكتبها في

البرنامج ، مباشرة بعد هذا السطر كتبنا :

```
#define PERSON_CLASS
```

قمنا بتعريف الثابت PERSON\_CLASS ، لاحظ أن الملف الذي كتبنا فيه هذه الأسطر هو ملف رأسى و استخدمناه بأن قمنا بتضمينه في ملف آخر باستخدام الأمر include ، تخيل أننا قمنا بتضمين الملف "class.hpp" أكثر من مرة كأن نكتب :

```
#include "class.hpp"
#include "class.hpp"
```

إذا قمنا بتضمين الملف مرتين فكأنما قمنا بكتابة أوامر الملف مرتين ، هكذا نكون قد عرفنا الفنة Person مرتين و يظهر لنا المصرف خطأ حيث لا يجوز تعريف أي فنة إلا مرة واحدة ، لكن في حالتنا يعمل أول سطران كطريقة للحماية من التضمين المتعدد ، إذا ضمنا الملف "class.hpp" أول مرة نفحص إذا كان الثابت PERSON\_CLASS ليس معرفاً في السطر الأول :

```
#ifndef PERSON_CLASS
```

الثابت لا يكون معرفاً في المرة الأولى لذا يتم قراءة الأسطر الباقية في الملف ، أول سطر بعد الفحص هو تعريف الثابت PERSON\_CLASS :

```
#define PERSON_CLASS
```

في حالة أخطأنا و قمنا بتضمين الملف مرة أخرى فإن المصرف يفحص مرة أخرى إن لم يكن الثابت PERSON\_CLASS معرفاً ، في هذه المرة يكون الثابت معرفاً من التضمين الأول و بهذا لا يتم قراءة الأسطر التالية بل يتم تجاهلها و بهذا لا نكون قد عرفنا فنة Person مرتين ، إذا لاحظت فقد قمنا بتضمين الملف iostream مرتين مرة في ملف البرنامج الرئيسي و مرة في الملف الرأسى ، لم تظهر أخطاء بسبب طريقة حماية مشابهة لما فعلنا في هذا الدرس ، يجب إنهاء أمر الفحص إذا لم يتم تعريف بأمر إنهاء إذا ، لكتابة هذا الأمر نكتب رمز المربع # بعده كلمة endif :

```
#endif
```

كتبنا هذا السطر في نهاية الملف الرأسى لإنهاء عملية الفحص ، أمر إذا تم تعريف يعمل بنفس الطريقة لكنه يقوم بقراءة الأسطر التي تليه إذا كان الثابت المعطى معرفاً بخلاف أمر إذا لم يتم تعريف الذي استخدمناه حيث يقرأ الأسطر التي تليه إذا كان الثابت المعطى غير معرف ، لاستخدام أمر إذا تم تعريف نكتب رمز المربع # بعده كلمة ifdef بعده الثابت الذي نريد فحصه :

```
#ifdef IDENTIFIER
```

بإمكاننا استخدام أوامر معالجة أولية شرطية أخرى تماثل في سلوكها جمل إذا و أما إذا و فيما عدا ذلك حيث هناك أوامر إذا و أما إذا و فيما عدا ذلك للمعالجة الأولية ، نكتب لأمر إذا كلمة if و لأمر أما إذا كلمة elif و لأمر فيما عدا ذلك كلمة else ، يمكن أن نعطي هذه أمري إذا و أما إذا شرطاً تكون قيمته ثابتة و ليست متغيرة و يمكننا فقط استخدام الثوابت الحرفية و الثوابت التي تم تعريفها باستخدام أمر التعريف define ، تنتهي سلسلة أوامر إذا و أما إذا و فيما عدا ذلك بأمر إنهاء إذا endif واحد ، مثلاً نستطيع أن نكتب :

```
#define NUMBER 20
#if NUMBER > 10
cout << "greater than 10" << endl;
#elseif NUMBER < 5
```

```
cout << "less than 5" << endl;
#endif
```

بإمكاننا أيضا استخدام أمر معرفّ **defined** و غير معرفّ **defined**! للحصول على نفس تأثير أوامر إذا تم تعريف و إذا لم يتم تعريف :

```
#if !defined PERSON_CLASS
#define PERSON_CLASS
#else
#if defined NUMBER
cout << "NUMBER is defined and PERSON_CLASS is not" << endl;
#endif //!defined PERSON_CLASS
#endif //defined NUMBER
```

تعلمنا اليوم :

١- معنى المعالجة الأولية : المعالجة الأولية هي أوامر يتم تنفيذها قبل تصريح البرنامج لتغيير شيء في شكله ، كل أوامر المعالجة الأولية تكون في سطر لوحدها و يتم بدء هذا السطر برمز **المربع #**.

٢- أمر **تضمن** : أمر **تضمن** هو أمر للمعالجة الأولية يقوم بإضافة ملف بلغة سي++ في المكان الذي نكتبه فيه :

```
#include <file>
#include "file"
```

٣- أمر **تعريف** و **إلغاء تعريف** : أمر **تعريف** يقوم بتعريف ثابت و يعطيه قيمة إن تم كتابة قيمة له ، أمر **إلغاء التعريف** يقوم بإلغاء ثابت تم تعريفه :

```
#define IDENTIFIER value
#undef IDENTIFIER
```

٤- أمر **إذا تم تعريف** و **إذا لم يتم تعريف** : أمر **إذا تم تعريف** يأخذ معرفًا و يخبر المصنف أن يقرأ الأسطر التي تلي الأمر فقط إذا تم تعريف الثابت مسبقا و إلا يتم تجاهل الأسطر التي تلي الأمر حتى أمر **إنهاء إذا** ، أمر **إذا لم يتم تعريف** يأخذ معرفًا و يخبر المصنف أن يقرأ الأسطر التي تلي الأمر فقط إذا لم يتم تعريف الثابت مسبقا و إلا يتم تجاهل الأسطر التي تلي الأمر حتى أمر **إنهاء إذا** :

```
#ifdef IDENTIFIER
.
.
.
#endif

#ifndef IDENTIFIER
.
.
.
#endif
```

حاول القيام بالآتي :

١- قم بكتابة ملفات رأسية مختلفة و قم بتضمينها ، إذا كان في برنامجك **فئات** فاجعل كل **فئة** في ملف رأسي لوحدها.

٢- قم بتضمين الملف الرأسي "class.hpp" أكثر من مرة ، لاحظ عدم وجود الأخطاء بسبب وجود مقطع الحماية ، احذف مقطع الحماية إذا لم يتم تعريف و **إنهاء إذا** من الملف الرأسي و لاحظ كيف تظهر الأخطاء بعد التصريف.