



المرحلة الأولى لإنتاج البرنامج

فئات البرامج

أولاً يوجد فئة من البرامج تسمى system programs وهي البرامج التي تهدف إلى حل مشاكل النظام (أي الحاسب خلافاً للمخصصة لحل مشاكل المستخدم والتي تسمى application program) من هذه الفئة الـ operating system وهو مجموعه من البرامج التي تمكن باقي البرامج من التعامل مع الـ hardware وهناك فئة أخرى وهي programming language tools وهي تنقسم إلى low level programming tools وهي أدوات مخصصة للغات البرمجة التي تتعامل مع الحاسب علي low level وتسمى low level programming language حيث لا تقدم هذه اللغات abstraction كبير للـ hardware وتمتاز هذه اللغات بقدره أكبر علي التعامل مع hardware ولذا تستخدم في عمل الـ device driver و أشهر هذه اللغات في مجال الحاسب الشخصي هي الـ assembly و أداة البرمجة الخاصة بها تسمى assembler و لغة الـ assembly تمتاز بأن كل أمر من أوامر اللغة يكفي أمر واحد من أوامر الـ machine language (اللغة الوحيدة التي يفهمها الحاسب مباشرة ويتكون الأمر فيها من سلسلة من الأرقام و الوحايد) أن الـ assembler يقوم بعملية assemble للبرنامج (البرنامج المكتوب بلغته غير machine language يسمى source code) فيحول كل أمر بلغته الـ assembly إلى الأمر المكافئ بالـ machine language ثم تأتي بعد ذلك مرحلة الـ linking وفيها يتم ربط ملف البرنامج بالملفات التي تحتاجها سواء ملفات أخرى صنعناها نحن أو ملفات نحتاجها من الـ library (وهي مجموعه من المكونات البرمجية الجاهزة التي يمكن أن نستخدمها) وبعد هذه المرحلة يخرج الملف القابل للتنفيذ أو الـ exe و بالطبع هو ملف بالـ machine language بعد عملية الـ linking

الـ high level programming tools

القسم الثاني هو الـ high level programming tools و هو أدوات لغات البرمجة الـ high level language التي تقوم بعمل abstraction كبير للـ hardware و الأمر في هذه اللغة ذات صياغة مشابهة للغة الإنجليزية مثل ++c , Fortran , Pascal و هذه اللغات يكون الأمر الواحد فيها بأكثر من أمر من أوامر الـ machine language والبرنامج أو الـ tool التي تحول أوامر الـ high level language إلى الـ machine language يسمى الـ compiler (بعض اللغات القديمة تستخدم ما يسمى بالـ interpreter بدلاً من الـ compiler وهو يحول أوامر البرنامج إلى أفعال بمجرد قراءة الأمر و لكنه أبطئ من الـ compiler و يحتاج إلى الملف source code عند الاستخدام) الـ compiler يحول الـ source code كله إلى الـ machine language مرة واحدة (وليس أمر أمر كما في الـ interpreter) ثم تأتي عملية الـ linking و يحدث فيها مثل ما حدث عند مرحلة الـ linking في لغة الـ assembly من ربط الملف مع الملفات الأخرى أو ملفات الـ library وبعد هذه المرحلة يخرج الملف القابل للتنفيذ أو الـ exe و الذي يوزع للمستخدم و نحفظ نحن بالـ source code





تركيب برنامج c++

ما المقصود بالFunction

أي برنامج في C++ يتضمّن على واحدة أو أكثر من الـ functions والـ function ببساطة كتلة مكتفية ذاتياً من الـ code ذات اسم فريد و التي تُستحضر invoked باستعمال اسم الـ Function. لتقريب مفهوم الـ function خذ مثلاً النشرة التي تصف جميع سيارة لعبة.. السيارة لها أربع عجلات و النشرة تكتب خطوات جميع العجلة مرة واحدة و عند الحاجة إلى تركيب عجلة تحيلك إلى الجزء المدون فيه خطوات جميع العجلة. أي برنامج C++ في بيئة DOS تحتوي على الـ function المسماة main(). جميع برامج C++ تبدأ التنفيذ من الـ main(). ويتميز اسم الـ functions بوجود الأقواس () بعده

فوائد الـ Functions

يُمكن أن تُكتب وتُختبر كل جزء من البرنامج منفصلاً. الـ function التي تؤدي مهمة خاصة يمكن إعادة استعمالها في البرامج الأخرى. مثلاً الـ standard library (مجموعة الـ functions والـ classes) التي تأتي مع C++ تزودك بالكثير من الـ standard functions التي يمكن أن تستعمل في برامجك

والآن مع أول مثال:

```
// A Simple Example of a Program
#include <iostream>
using namespace std;
int main ( )
{
    int apples, oranges;
    int fruit;
    apples = 5; oranges = 6;
    fruit = apples + oranges;
    cout << endl;
    line cout << "Oranges are not the only fruit... " << endl << "- and we have " << fruit <<
    " fruits in all.";
    cout << endl;
    line return 0 ;
}

// Declare two integer variables
// ...then another one
// Set initial values
// Get the total fruit
// Start output on a new
// Start output on a new
// Exit the program
```

ما المقصود بالComment

السطر الأول في هذا الـ code هو comment والـ comment جزء من الـ code غير قابل للتنفيذ وهو ببساطة ليُساعد القارئ للبرنامج على فهم الـ code. كل الـ comments تهمل من قبل الـ compiler. عند أي سطر من الـ code فان وجود شرطتان متتاليتان // يدلان على أن باقي السطر (الجزء الذي يلي الشرطتان) هو comment. هناك عدة سطور في هذا البرنامج تحتوي على comment. و يُمكن أن يستعمل شكل بديل للـ comment يكون فيه الـ comment محصور بين العلامة /* و العلامة */ فمثلاً السطر الأول يمكن أن يكتب





/* A Simple Example of a Program */

ويعتبر هذا النوع من comment بأنه يمكن أن يمتد لأكثر من سطر
خلافًا للنوع الآخر (باستخدام الشرطتان) فمثلًا يمكن كتابة
الcomment السابق في أكثر من سطر

/* A Simple Example of a Program */

صوره شائعة لاستخدام هذا النوع من comment وفيها تتراس نجوم
كجزء من التعليق بين علامتي التعليق (بالون الأحمر)

/******

* EX1-01.CPP *

* A Simple Program Example *

*****/*

الcomment يجب أن تكون كافية لأي مبرمج آخر، أو أنت في وقت تالي
أن تفهم غرض أي قطعة من code وأن تفهم كيف يعمل.

ما المقصود بالPreprocessor directives

السطر الذي يلي comment هو preprocessor directive يسمى
include directive و عموماً تبدأ directives بالعلامة # و يجب أن
تكون في أول جزء في البرنامج قبل كتابته أي نوع آخر من الأوامر
وفي الحقيقة directives ليست ++ statement وإنما هي لتوجيه
الcompiler لتنفيذ أشياء على source code قبل أن يقوم بعملية
compilation

ما الذي يقوم به Include directive

include يجعل compiler يدخل محتويات الملف المذكور بين علامتين
< > في source code قبل compilation هذا النوع من الملفات
يدعى header file ويحتوي declarations ضرورية لكي تستخدم
الobjects و الfuctions الموجودة في standard library وكل
header file يحتوي مجموعه من declarations للobjects و الfuctions
الخاصة بعمليات معينه فهناك header file خاص بالعمليات على ال
strings (السلاسل الحرفيه) مثلاً و هناك اخر خاص بالماتيماتيكا
وهكذا

ما هو File iostream

هذا file يحتوي declarations يمكن لنا بها أن نستعمل ال input و
ال output واذالم نعمل include لهذا file فان البرنامج سيصدر خطأ
في اثناء linking لانه سيجد الobjects الخاصه بالinput و
الoutput (والتي استخدمت في البرنامج) ليست لها declarations

ما الذي تقوم به Using namespace

using هذا directive ايضا لوانه شذا عن القاعده في انه لم يسبق
ب # و دوره انه يحدد ال namespace الذي تنتمي اليه ال objects و
ال fuctions المستخدمه في البرنامج ال objects و ال fuctions الموجوده
في standard library محتواه داخل ال namespace المسمى std وبما اننا
سوف نستخدم في برنامجنا ال objects و ال fuctions الموجوده في
ال standard library فاننا يجب ان نحدد namespace الذي تنتمي





اليه ال objects و ال fuctions في برنامجنا وهو std وذلك ما تفعله الجملة الثالثه

والان قلب البرنامج ال () Function main

ال Function المسماه () main هي ال Function التي تستدعي اولاً عند بدأ البرنامج من قبل نظام التشغيل و هي عبارة عن ال Function التي يتم منها عمل ال calling لل member Functions (ال Functions المنتميه ل object) وال stand alone function (ال Functions الغير منتميه ل object) و يتوقف ال flow للبرنامج علي ال calling الذي يحدث داخل ال () main وهنا في هذا البرنامج فانه لا يوجد Function نكتبها نحن سوى () main

مكونات ال Function

تتكون ال Function من function header وهو السطر الـ اربع وهو يتكون من ثلاثة اجزاء الاول و هو هنا ال keyword المسماه int وهي تدل علي ال data type ال returned بواسطة الداله اما الجزء الثاني فهو اسم الداله و هو هنا main اما الجزء الثالث فهو الاقواس () وهي ليست دائماً فارغه (كما في هذه الحاله) و يوضع فيها ال data type لل arguments المرسله الي ال function ثم يلي ال function header ال function body ويبدأ ب { وينتهي ب } وما بينهما هما ال c ++ statements الخاص بهذه ال function

