بسم الله الرحمن الرحيم

Principles Of Programming Languages

مبادئ لغات البرمجة

عدد الساعات: ٢ نظري+٢عملي

الرمز: 211 حسب

المتطلبات: 111 حسب (مقدمة في الحاسب)

أستاذة/المادة: م. نجلاء حسن

Lecture 2:

- الربط binding
- التحقق من النوع Type checking
- مدي المتغير Variable Scope
- مقدمة عن C++
- الشكل العام لبرنامج C++
- عبارات الادخال /الاخراج
- المتغيرات والثوابت

الربط binding

- رابطة بين خاصية و فئة (المتغير و نوعه) أو بين عملية و رمز.
- الزمن الذي يتم فيه الربط يسمي بزمن الربط.
- عملية الربط تختلف من لغة للغة من حيث الزمن الذي يتم فيه الربط

حيث عملية الربط ممكن أن تتم في:

- ✓ زمن تصميم اللغة language design time
- ✓ زمن تعريف اللغة definition time
- ✓ زمن الترجمة compile time
- ✓ زمن التنفيذ run time

وبالنظر إلى الوحدة البرمجية التالية من لغة C حيث يتم إسناد قيمة إلى المتغير count

```
Int   count;

Count = count +5;
```

فالذي يحدث كالاتي:

المجموعة المحتملة من النوع للمتغير count : تربط في زمن التصميم.

- نوع count : تربط في وقت الترجمة.
- مجموعة القيم المحتملة لـcount : نربط في زمن تصميم المترجم.
- قيمة count : نربط في زمن التنفيذ مع الكود.
- مجموعة المعاني للعلامة(+) : تربط في زمن تعريف اللغة.
- معنى العلامة(+): تربط في زمن الترجمة.

1.4. ربط النوع Type Binding

قبل استخدام متغير معين في برنامج، يجب أن يربط إلى نوع بيانات (data type) . المهم في الأمر هو أمران الأول هو كيف نحدد نوع البيانات والثاني هو متى يتم الربط.

يمكن أن يحدد النوع في السكون عن طريق تصريح مباشر أو غير مباشر.

أ- الربط الساكن static binding :

يمكن التصريح عن المتغيرات بإحدى طريقتين :

1- التصريح المباشر Explicit Dedaration

هو عبارة عن صيغة في البرنامج يذكر فيها أسماء المتغيرات ونوع بياناتها .

2- التصريح غير المباشر Implicit Dedication

هو عبارة عن رابطة بين المتغيرات والنوع عبر العلاقة الافتراضية في الصيغة، في هذه الحالة أول ظهور لاسم المتغير يعتبر هو التصريح غير المباشر لهذا المتغير .

وفي الحالتين ينشأ نوع من الربط الساكن بين المتغير والنوع. أغلب اللغات التي تم تصنيعها قبل منتصف الستينيات من القرن الماضي تحتاج إلى تصريح مباشر للمتغيرات ، أما اللغات التي تم تطويرها بعد ذلك تحتاج إلى تصريح غير مباشر مثال ذلك Fortran و Basic .

ب-الربط غير الساكن أو الديناميكي Dynamic Binding

في هذا النوع من الربط نجد أن نوع البيانات لا يحدد بواسطة صيغة تصريح بل بربط المتغير بنوع البيانات عندما تسند له قيمة في صيغة الإسناد وعندما يتم تنفيذ صيغة الإسناد تتم عملية ربط المتغيرات في صيغة الإسناد **لقيمة النوع** ، أو **متغير**، أو **تغير** في الجهة اليمنى من صيغة الإسناد.

الفائدة المهمة للربط غير الساكن أن هنالك مرونة في البرمجة مثال ذلك البرنامج الذي يكتب بلغة تتعامل مع الربط غير الساكن يمكن استخدامه مع أي نوع بيانات و السبب في ذلك لأن المتغير يتم ربطه مع أي نوع بيانات أثناء التنفيذ.

والعكس صحيح في اللغات التي تتطلب التصريح الساكن لابد من تحديد نوع البيانات مسبقاً مثل لغة C و Pascal ولكن هنالك مساوئ أيضاً للتصريح غير الساكن وهي:

عيوب الربط الغير ساكن:

- لا يمكن تتبع الاخطاء لان نوعي البيانات يمكن ان يظهرا في جانبي صيغة الاسناد.
- التكلفة بسبب استخدام اوعية تخزين (مخازن) متغيرة الحجم لعملية التحقق من النوع في زمن التشغيل لان كل نوع يتطلب حجم مختلف.

اللغات التي تستخدم التصريح الغير ساكن تستخدم مفسرات.

ج- ربط التخزين ودورة حياة المتغير

في العادة يتم تخصيص خلية أو يتم حجز مساحة من الذاكرة المتوفرة للمتغير فيما يسمى عملية حجز الذاكرة (allocation) عند عملية الربط ، ثم يتم تحرير هذه الخلية بعد فك ربط المتغير (Deallocation) وإعادتها إلى الذاكرة المتوفرة.

دورة حياة المتغير هي عبارة عن **زمن** ربط المتغير بخلية محددة في الذاكرة والتي تبدأ مع بداية ربط المتغير مع خلية الذاكرة وتنتهي بنهاية عملية ربط المتغير مع تلك الخلية .

التحقق من النوع Type Checking

التحقق من نوع المعامل operator مناسب للـ operands ام لا.

خطأ النوع Type Error

هناك رسائل تنتج عند تنفيذ البرنامج من نوع type error أي ان المعامل المطبق علي الـ operands من نوع غير مناسب لها.

تتم عملية التحقق من النوع كالتالي:

- ✓ اذا كان الربط ساكن فان عملية التحقق تكون ساكنة (في زمن الترجمة).
- ✓ اذا كان الربط غير ساكن فان عملية التحقق تكون غير ساكنة (في زمن التنفيذ/التشغيل).

- التحقق الساكن افضل.
- هناك لغات مثل APL لا تسمح بالتحقق الساكن.

مدي/مجال المتغير Variable Scope

المجال الذي يري فيه المتغير (فترة حياة المتغير).

انواع المدي

١. عام/مشترك Global

متغير يعرف و يستخدم في كافة اجزاء البرنامج.

٢. خاص/محلي local

متغير يعرف و يستخدم في جزء معين من البرنامج.

مثال:

في لغة C++

```
int x;                          // متغير عام
x=50;
void main()                     //  البرنامج (الدالة الرئيسية)
{
    if  (x>20)                  //  كتلة برمجية
    {                           // بداية الكتلة
        cout<<x;
        int x=40;               // متغير محلي
        cout<<x;
    }                           //  نهاية الكتلة
    cout<<x;
}
```

تابع انواع المدي

هناك تصنيف آخر لأنواع المدي:

١. المدي الساكن Static Scope
يتحدد فيه مجال المتغير قبل التنفيذ (الترجمة).

٢. المدي الديناميكي Dynamic Scope
يتحدد فيه مجال المتغير اثناء التنفيذ و ذلك مع استدعاءات الاجراءات و الدوال.

مقدمة عن C++

- تعتبر لغة C++ من أشهر اللغات التي تتمتع بطابع القوة والمرونة لإنتاج أسرع برامج وأفضلها أداءاً. وعلى الرغم من وجود العديد من لغات البرمجة الأخرى إلا أنها تفتقر شمولية لغة C++ وقوتها . فاللغة C++ تتميز بقابليتها على معالجة التطبيقات الكبيرة والمعقدة، والقوة في صيانة البرامج المكتوبة بها مما يوفر وقتاً في تصميم البرامج وتطويرها .
- تعتبر اللغة C++ امتداداً للغة C. وقد أنشأها Bjarne Stroustrup عام ١٩٧٩ م، وكانت تسمى حينها C مع فئات(C with classes)، وتغير اسمها إلى C++في العام ١٩٨٣م.
- تعتمد لغة C++ أسلوب البرمجة كائنية المنحى Object Oriented Programming، والذي يعرف اختصاراً بـ (OOP)، والذي تم تطويره بسبب قيود كانت أساليب البرمجة القديمة المتمثلة في اللغات الإجرائية تفرضها على المبرمجين

الشكل العام للبرنامج:

Header Files

```
1. #include <library_name.h>
```

```
2. Public  Declaration
```

Program Body

```
3. Main ()
4. {
5.          Private Declaration

6.          Statements..
            Statements..
            Statements..
7. }
```

١. استيراد المكتبات

٢. منطقة التصاريح العامة

٣. الدالة الرئيسية

٤. بداية الدالة الرئيسية

٥. منطقة التصاريح الخاصة

٦. جمل برمجية

٧. نهاية الدالة الرئيسية

مثال: برنامج يعرض نص على الشاشة

```
//This program will display a message on the screen.
#include <iostream.h>
int main ( )
{
        cout <<''welcome to C++ !\n'';
return 0;
}
```

Output:

welcome to C++

التعليقات:Comments

عبارة عن توضيحات يكتبها المبرمج لا تدخل في تركيب البرنامج (لا ينفذها المترجم).

وتكون على شكلين :

- تعليق السطر الواحد :

```
1. // This is a comment
2. // And this is another comment
```

- تعليق الأسطر المتعددة:

```
1. /*
2. This is a comment
3. In tow lines
4. */
```

مرشدات المهيئ (Preprocessor Directive):-

#include<iostream.h>

- يسمى هذا بمرشد المهيئ Preprocessor directive، وهو عبارة عن تعليمة للمصرف أن يدرج كل النص الموجود في الملف iostream.hفي البرنامج، وهو ملف يجب تضمينه مع أي برنامج يحتوى على عبارات تطبع بيانات على الشاشة أو تستقبل بيانات من لوحة المفاتيح.
- يسمى iostream ملف ترويسة (header file)، وهنالك الكثير من ملفات الترويسة الأخرى، مثل:

stdio.h : مكتبة عامة «اقدم مكتبة»

conio.h : للتعامل مع دوال أوامر الشاشة

math.h : للتعامل مع الدوال الرياضية (sin(), cos())

String : للتعامل دوال معالجة النصوص (سلاسل الأحرف)

الدالة main :-

main( )

- يبدأ تشغيل أي برنامج C++ من دالة تدعي ( )main، وهي دالة مستقلة ينقل نظام التشغيل التحكم إليها. وهي جزء أساسي في برنامج C++.
- الأقواس بعد main تشير إلى أن mainهي عبارة عن دالة. قد يحتوى برنامج C++ على أكثر من دالة إحداهما بالضرورة هي main. يحتوى البرنامج السابق على دالة واحدة.
- يبدأ تنفيذ البرنامج من الدالة main حتى لو لم تكن هي الأولى في سياق البرنامج. يتم حصر جسم الدالة main بأقواس حاصرة { } .

الخرج إلى الشاشة:-

```
cout<<" welcome to C++ !\n";
```

- هذه العبارة (statement) تجبر الحاسوب أن يظهر على الشاشة النص المحصور بين علامتي الاقتباس " ". ويسمى هذا النص ثابت سلسلي.
- يجب أن تنتهي كل عبارة في برنامج C++ بفاصلة منقوطة (semi colon) ;
- الاسم cout والذي يلفظ كـ C out يمثل كائن في C++ مقترن مع الشاشة والعامل >> والذي يسمى بعامل الوضع Put to operator يجبر على إرسال الأشياء التي على يمينه إلى أي شئ يظهر على يساره.
- مثال:

```
#include <iostream.h>
main ( )
{
 cout  << 7 << " is an integer.\n";
cout << 'a' << "is a character.\n";
}
```

Output:

7 is an integer.
a is a character

تتابعات الهروب (Escape Sequences):

\ تسمى الشرطة الخلفية (Back slash)أو حرف هروب (Escape character) وتسمى هي والحرف الذي يليها تتابع هروب. تتابع الهروب n\ يعنى الانتقال إلى سطر جديد حيث يجبر المؤشر على الانتقال إلى بداية السطر التالي ، الآن إليك بعض تتابعات الهروب الشائعة:-

تتابع الهروب	الوصف
\n	سطر جديد.
\t	مسافة أفقية.
\b	حرف التراجع back space.
\\	لطباعة شرطة خلفية.
\r	حرف الإرجاع، يجبر المؤشر على الانتقال إلى بداية هذا السطر.
\"	لطباعة علامة اقتباس

المناور **endl**:-

العبارة:

```
cout<<''sum= ''<<sum<<endl
```

تطبع النص =sum متبوعاً بقيمة sum ، نلاحظ أننا استخدمنا endl وهو وسيلة أخرى في C++ للانتقال إلى سطر جديد، ويسمى مناور manipulator و endl اختصاراً لـ end line، وهو يعمل تماماً كما يعمل تتابع الهروب \n .

المتغيرات variables:

- العناصر التي تأخذ قيم متغيرة في البرنامج.
- عند كتابة أي برنامج بلغة C++، نحتاج لتخزين المعلومات الواردة للبرنامج في ذاكرة الحاسوب تحت عناوين يطلق عليها أسماء المتغيرات، وبما أن أنواع المعلومات المراد تخزينها تكون عادة مختلفة مثل القيم الحقيقية أو الصحيحة أو الرمزية فإننا نحتاج أن نعلم المترجم في بداية البرنامج عن أنواع المتغيرات التي نريد استخدامها فمثلاً :-

الكلمات sum integer1 , integer2, هي أسماء لمتغيرات عبارة عن أعداد صحيحة(النوع int) وهو أحد أنواع البيانات المتوفرة في C++ .

- يجب تعريفها قبل استعمالها، وتنقسم الى نوعين:

- عام أو مشترك (global): تستخدمه كافة أجزاء البرنامج (الدوال) وتعرف ونعرِّف هذه المتغيرات بعد جمل #include< > مباشرة.

- خاص أو محلي (Local): يستخدمه جزء معين اما برنامج أو طبقة أو دالة معينه ، ونعرِّف هذه المتغيرات داخل الجزء الذي يستخدمه فقط.

- يمكن تعريف المتغيرات التي تنتمي إلى نفس النوع في سطر واحد.

الثوابت Constants:

- العناصر التي تأخذ قيم ثابتة طوال تشغيل البرنامج.
- يتم تعريفها مثل تعريفها مثل تعريف المتغيرات.
- طرق تعريفها:

- باستخدام const قبل نوع بيانات الثابت، مثال:

```
const int Max =10;
```

- باستخدام #define قبل اسم الثابت، مثال:

```
#define nn 5 ;
```

شروط تسمية المتغيرات والثوابت:

١. لا يبدأ برقم أو عملية حسابية أو رمز ما عدا _ (underscore).
٢. ألا يحتوي على عملية حسابية أو رمز أو فراغ.
٣. ألا يزيد عن 255 حرفاً.

مهم جداً:

اسم المتغير أو الثابت اذا أعيد استخدامه مع تغير حالة حرف أو أكثر من (صغيرة الى كبيرة وبالعكس) فانه يكون لدينا اسماً جديد اً لا علاقة له بالاسم الذي سبق وعرفنا ه .(يولد خطأ في التنفيذ)

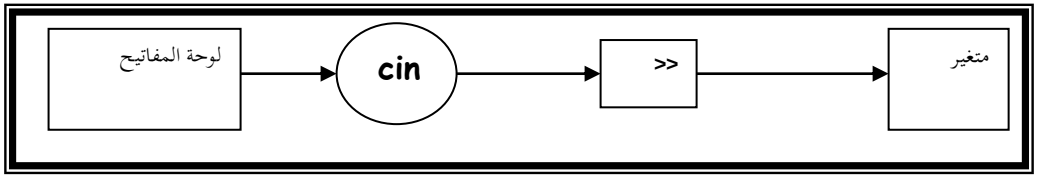
الدخل من لوحة المفاتيح:-

cin>>integer1

- هذه العبارة تخزن الرقم الذي يكتبه المستخدم من لوحة المفاتيح في متغير يدعي integer1. يمثل الكائن cin-والذي يلفظ كـ C in- لوحة المفاتيح، ويأخذ عامل الحصول (>>) get from الأشياء الموضوعة على يساره ويضعها في المتغير الموجود على يمينه، عند تنفيذ هذه العبارة ينتظر البرنامج أن يكتب المستخدم رقماً من النوع integer ويضغط على مفتاح Enter ، يتم تعيين القيمة التي أدخلها المستخدم إلى المتغير integer1 .
- يمكن استعمال عامل الحصول عدة مرات في نفس العبارة:

cin >> integer1>>integer2

يضغط المستخدم هنا Enter، أو مفتاح المسافة Space، أو مفتاح Tab بعد كل قيمة، قبل أن يكتب القيمة التالية، ولكنه من الأفضل عادة إدخال قيمة واحدة في كل مرة لتجنب الخطأ.



شكل ي وضح الدخل بواسطة C++

الاسناد :

- هو منح (اسناد) قيمة لمتغير أو أكثر، ويمكن أن تكون القيمة عبارة عن عدد معين أو تكون قيمة مخزنة في متغير آخر أو الاثنين معاً.
- تتلخص عملية الاسناد في استخدام علامة التساوي (=)
- مثال:

* A=5.5 ، تعني تخزين القيمة 5.5 في المتغير A
* c=d ، تعني تخزين قيمة المتغير d في المتغير c
* h=i=j=1 ، تعني .....