كلية العلوم

القسم : الرياضيات

السنة : الاولى

المادة : لغات البرمجة1

المحاضرة : الثانية/عملي/
تنزيل الانسة

{{ مكتبة A to Z }}

Facebook Group : مكتبة A to Z

الجمهورية العربية السورية

جامعة طرطوس

كلية العلوم قسم الرياضيات

السنة الاولى

المادة: لغات البرمجة 1 – عملي

المحاضرة الثانية
أساسيات كتابة البرنامج

2026

أولاً: البنية الأساسية لبرنامج C++ :

تتكون C++ من مجموعة عناصر أساسية يجب توافرها ليتمكن البرنامج من التنفيذ بشكل صحيح.

1) تضمين المكتبات:

يتم تضمين المكتبة بالشكل التالي:

```
#include<iostream>
```

تستخدم عبارة #include لتضمين ملف رأس (Header File) ضمن البرنامج ، بحيث تتيح للمبرمج استخدام تعريفات الدوال و الأوامر الجاهزة الموجودة فيها ، دون الحاجة إلى إعادة تعريفها.

تعد مكتبة <iostream> من أهم المكتبات في لغة C++ ، وهي اختصار ل Input Output Stream ، حيث توفر أدوات الإدخال و الإخراج القياسية مثل : cin لإدخال البيانات ، cout لطباعة البيانات .

التطور التاريخي لكتابة المكتبات:

في الإصدارات القديمة من لغة C++ (وخاصة المتأثرة بلغة C) كانت المكتبة تكتب بالشكل التالي:

```
#include<iostream.h>
```

- كانت تحتوي على الامتداد h. وهو اختصار ل Header ، وكان يستخدم للدلالة على ملفات الرأس
- لم تكن تستخدم مفهوم مساحات الأسماء (Namespaces)
- كانت جميع الدوال تستخدم مباشرة دون تنظيم ضمن نطاق معين

مع تطور لغة C++ تم تحديث طريقة كتابة المكتبة لتصبح :

```
#include<iostream>
```

- تم إزالة الامتداد h.
- تم إدخال مفهوم مساحات الأسماء (Namespaces) ، حيث لم تعد الدوال مثل cin و cout متاحة بشكل مباشر بل أصبحت ضمن مساحة اسم محددة.

2) مساحة الأسماء (Namespace) :

تستخدم مساحات الأسماء لتنظيم العناصر البرمجية (مثل الدوال والمتغيرات) ضمن نطاقات محددة ، وذلك لتجنب التعارض بين الأسماء ، وتعتبر std المساحة القياسية في C++ ، والتي تحتوي على العديد من العناصر الجاهزة مثل cin و cout .

يتم كتابتها بطريقتين:

الطريقة الكاملة:

```
std::cout<< "hello";
```

حيث عند استخدام cin أو cout ، يجب الإشارة إلى هذه المساحة.

الطريقة المختصرة:

```
using namespace std;
```

ونستخدم هذه الطريقة لتبسيط الكتابة ، حيث تسمح باستخدام عناصر المكتبة مباشرة دون الحاجة إلى كتابة std:: في كل مرة .

3) الدالة الرئيسية (Main Function) :

تعد الدالة main() نقطة البداية لتنفيذ أي برنامج مكتوب بلغة C++ ، حيث يبدأ المترجم بتنفيذ التعليمات البرمجية من داخل هذه الدالة ، وإن لم تكن موجودة فستظهر رسالة خطأ توضح ذلك؛ كما أنه لاتوضع فاصلة منقوطة في نهاية اسم الدالة .

يتم كتابتها بالشكل التالي:

```
int main()
{


}
```

- () : تشير هذه الأقواس إلى أن العنصر دالة وليس متغير .
- {} : تحتوي هذه الأقواس على التعليمات البرمجية التي يقوم البرنامج بتنفيذها ، ويتم التنفيذ بشكل متسلسل من الأعلى إلى الأسفل داخلها.

ثانياً: أساسيات الإدخال و الإخراج:

1) الإدخال باستخدام تعليمة cin:

بعد تعريف المتغيرات في لغة C++ وإسناد قيم مباشرة لها باستخدام عامل الإسناد = ، يمكننا أيضاً تهيئة هذه المتغيرات بشكل ديناميكي من قبل المستخدم عبر لوحة المفاتيح. في هذه الحالة لايتم تحديد القيمة مسبقاً داخل الكود، وإنما يتم استقبالها وقت تشغيل البرنامج باستخدام تعليمة الإدخال cin.

يتم كتابتها بالشكل التالي:

cin>>variable; => cin >> اسم المتغير;

- cin : تعليمة تستخدم لإدخال البيانات من المستخدم
- >> : معامل الإدخال ويعني خذ البيانات من المستخدم
- Variable: هو المتغير المعرف مسبقاً الذي تخزن فيه القيمة ، ويجب دائماً تعريف المتغير قبل استخدام تعليمة cin .

يتم تخزين القيم المدخلة داخل الذاكرة، أي تعيين قيمة المتغير في الذاكرة باستخدام لوحة المفاتيح؛

لا يجوزاستخدام المتغير قبل تعريفه.

```
cin >> a;
```

خاطئ، لم يتم التصريح عن المتغير.

```
int a;
cin>> a;
```

صحيح، تم التصريح عن المتغير ثم استخدامه في دالة الإدخال.

لإدخال عدد من المتغيرات من نفس النوع نكتب:

```
int x,y;
cin >> x>>y;
```

يتم الفصل بين المتغيرات المدخلة باستخدام معامل الإدخال (>>).

في هذا المثال يتم إدخال قيمتين من المستخدم ، حيث تخزن القيمة الأولى في المتغير x والقيمة الثانية في المتغير y وفق ترتيب الإدخال.

2) الإخراج باستخدام تعليمة cout:

تستخدم تعليمة cout لعرض البيانات على الشاشة ، سواء كانت هذه البيانات قيم متغيرات أو نصوص توضيحية أو مزيجاً بينهما.

يتم كتابتها بالشكل التالي:

cout<<value;

- cout : تعليمة تستخدم لطباعة البيانات على الشاشة
- >>: معامل الإخراج ويعني أرسل البيانات إلى الشاشة
- value : يمكن أن يكون قيمة لمتغير أو نصاً أو تعبيراً

طباعة عبارة نصية:

يمكن استخدام تعليمة cout لطباعة نصوص مباشرة من خلال وضعها بين علامتي اقتباس مزدوجتين.

```
cout << "الجملة";
```

تتم طباعة الجملة التي بين قوسين كما هي.

طباعة القيم العددية:

```
cout << 10;
```

طباعة قيمة متغير:

```
int x=5;
cout << x;
```

طباعة تعبير رياضي:

```
cout << 25*5;
```

طباعة متغيرين عدديين أو متغير وجملة نصية:

```
int x =3; float y= 4.3;
cout<< "x = "<<x<<" and y = "<<y;
```

يتم الفصل بين المتغيرات المراد طباعتها باستخدام معامل الإخراج (<<).

في هذا المثال يتم عرض الناتج بشكل متسلسل على الشاشة ، ويكون على الشكل التالي :

```
x = 3 and y = 4.3
```

تنسيق الخرج:

عند استخدام تعليمة cout فإن الإخراج لايقتصر على طباعة القيم فقط ، بل يمكن التحكم بشكل عرض البيانات وتنظيمها باستخدام بعض المحارف الخاصة مثل \n \t endl .

- المحرف \n : يستخدم لإضافة انتقال إلى سطر جديد.

مثال:

```
cout<<"Hello\nWorld";
```

ويكون الناتج:

```
Hello
World
```

- المؤثر endl: وهو اختصار ل end line أي إنهاء السطر ، عند إدراج هذا المؤثر في عبارة cout فإنه يجبر المؤشر على الانتقال إلى بداية السطر التالي.

مثال:

```
cout<<"Hello"<<endl<<"World";
```

ويكون الناتج:

Hello

World

- المحرف \t : يستخدم لإضافة مسافة أفقية بين العناصر ، مثل الضغط على زر Tab في لوحة المفاتيح.

مثال:

cout<<"Nama:\tAli";

ويكون الناتج:

Nama:	Ali

مثال 1: اكتب برنامج لإدخال ثلاث علامات لطالب وحساب المعدل وطباعته على الشاشة.

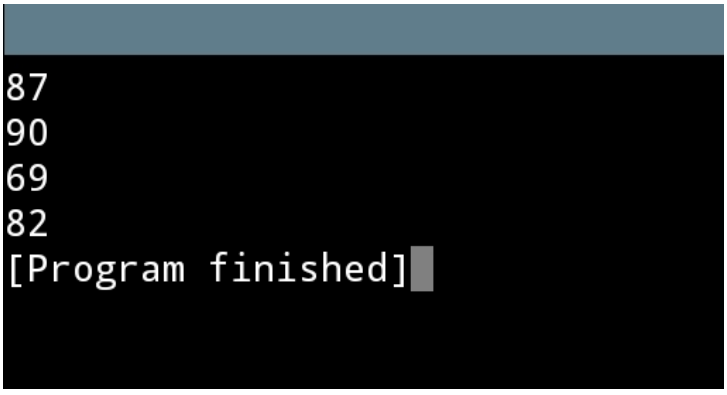
```
#include<iostream>
using namespace std;
int main(){
    int a,b,c;
    cin>>a>>b>>c;
    cout<<(a+b+c)/3;
    }
```

تم التصريح عن المتغيرات الثلاثة التي تمثل العلامات.

إدخال قيم المتغيرات.

طباعة المعدل وهو مجموع العلامات على عددها.

الخرج:



مثال 2 : اكتب برنامج يقوم بجمع عددين وعرضهما.

```
#include<iostream>
using namespace std;
int main(){
        int x=10, y=20;
        cout <<10<<"+"<<20<<"="<<10+20;
        cout << "\nx+y= "<<x+y<<endl;
        cout <<"10+20="<<10+20;
         }
```

تم تعريف المتغيرين الصحيحين.

طباعة قيم عددية مع قيم نصية.

طباعة قيم نصية مع تعبير رياضي.

endl تعني سطر جديد تعمل كمحرف الهروب '\n'

الخرج:

```
10+20=30
x+y= 30
10+20=30
[Program finished]
```

أمثلة تطبيقية:

مثال 1 : ما هي الأخطاء في هذا البرنامج ثم أوجد الخرج:

البرنامج الصحيح:

```
#include<iostream>

using namespace std;

int main(){

        int x,y,s;

        cout<< “Enter the number x:”;

        cin>>x;

        cout<<”Enter the number y:”;

         cin >>y;

         s =x+y;

         cout<< “x+y= “<<s;

         }
```

البرنامج الخاطئ:

```
#include>iosrream>

using namespace std;

int main({

        integer x,y,s

        cout>> “Enter the number x:”;

        cin<<x;

        cout<<Enter the number y:”;

         cin >>y

         s =x+y;

         cout<< x+y=>>s;

         };
```

الخرج:

```
Enter the number x:23
Enter the number y:5
x+y= 28
[Program finished]
```

مثال 2 : في هذا المثال تم التعريف عن ثلاث متغيرات وإسناد قيم لها ، المتغير result1 تكون قيمته هي ناتج العملية الحسابية، سيتم تنفيذ عملية الزيادة اللاحقة أولاً تصبح قيمة a تساوي 6 ثم الضرب (3*10) ثم يتم تعديل قيمة b في الذاكرة فتصبح 2 ، تتم طباعة قيمة المتغير result1 على الشاشة ثم البدء بسطر جديد.

```cpp
#include <iostream>
using namespace std;

int main() {
    int a = 5;
    int b = 3;
    int c = 10;
    int result1 = ++a + b-- * c;
    cout << result1 << endl;

```

```
36

[Program finished]
```

مثال 3 : في هذا المثال تم التعريف عن ثلاث متغيرات ، سيتم ادخال قيم a,b من لوحة المفاتيح والمتغير result ستكون قيمته تساوي ناتج العملية الحسابية، سيتم طباعة العبارتين التوضيحتين قبل عملية الادخال ثم عرض قيمة المتغير result وإضافة سطر جديد.

```cpp
#include <iostream>
using namespace std;
int main() {
    int a, b, result;
    cout << "Enter a number  ";
    cin >> a;
    cout << "Enter a number  ";
    cin >> b;
    result = ++a * (b + 2) / 2;
    cout << result << endl;
}
```

```
Enter a number  1
Enter a number  2
4

[Program finished]
```

مثال 4 : في هذا المثال تم التعريف عن ثلاث متغيرات وإسناد قيم لها، المتغير result تكون قيمته هي ناتج العملية الحسابية ، سيتم تنفيذ عملية الزيادة اللاحقة أولاً تصبح قيمة a تساوي 7 ثم الضرب (4*7) ثم يتم تعديل قيمة b في الذاكرة فتصبح 3 ثم يتم حساب العملية العلائقية ((4*7)>12) العبارة صحيحة فتكون النتيجة 1، باعتبار أن التعبير الأول ل OR يساوي ال 1 فمن المؤكد أن النتيجة تساوي الواحد فلا تهم قيمة التعبير الثاني لأنه أياً يكن نتيجة العملية معروفة وبالتالي لا يقوم المترجم بتنفيذ التعبير الثاني، تتم طباعة قيمة المتغير result على الشاشة ثم البدء بسطر جديد يحتوي قيم المتغيرات الثلاثة الجديدة a,b,c ونلاحظ أن قيمة المتغير c تبقى كما هي نتيجة تحقق التعبير الأول ل OR وعدم الحاجة للتحقق من التعبير الثاني .

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4     int a = 6;
5     int b = 4;
6     int c = 12;
7
8     bool result = ((++a * b-- > c) || (--c + a <
   b));
9
10    cout << result << endl;
11    cout <<" a = "<< a << ", b = " << b << ", c =
   " << c << endl;
12 }
```

```
1
 a = 7, b = 3, c = 12

[Program finished]
```

مثال 5 : اكتب برنامج يقوم بحساب المعادلة التالية:

$$e = \frac{2xy}{(x+y)} + \frac{2x}{2(x-z)}$$

تم التعريف عن ثلاث متغيرات وإدخال قيمها من لوحة المفاتيح، سيظهر على الشاشة العبارة النصية ضمن دالة الطباعة ثم نتيجة المعادلة.

```cpp
#include<iostream>
using namespace std;
int main(){
   int x,y,z;
   cin>>x>>y>>z;
   cout<<" e = 2xy/(x+y) + 2x/2(x-z) = "<< ((2*x*
y)/(x+y)) + ((2*x)/(2*(x-z)));
    }
```

```
4
2
3
 e = 2xy/(x+y) + 2x/2(x-z) = 6
[Program finished]
```

مثال 6 : اكتب برنامج لحساب مساحة ومحيط دائرة وطباعتهما، علماً أن نصف القطر مدخل من لوحة المفاتيح.

تم التعريف عن متغيرين r هو نصف قطر الدائرة سيتم إدخاله من لوحة المفاتيح والمتغير p وهو ثابت وتساوي قيمته 3.14، سيظهر على الشاشة العبارة النصية لحساب المحيط ثم قيمة المحيط ثم سطر جديد يحتوي على عبارة نصية لحساب مساحة الدائرة وقيمتها.

```cpp
#include<iostream>
using namespace std;
int main(){
   int r;
   float p;
   cout<<"enter r: ";
   cin>>r;
   p=3.14;
   cout << " S = "<< 2*p*r<<endl;
   cout <<" A = "<<p*r*r;
    }
```

```
enter r: 9
 S = 56.52
 A = 254.34
[Program finished]
```