المصفوفات والسلاسل Arrays and Strings

المحاضرة الثالثة
د. كندة سليمان أبوقاسم

جامعة المنارة
MANARA UNIVERSITY

هدف المحاضرة دراسة المفاهيم التالية

1. السلاسل string
2. قراءة سلسلة من لوحة المفاتيح Reading a String from the Keyboard
3. العمليات على السلاسل
   - strcpy() : copy characters from one string to another
   - strcat() : concatenation of strings
   - strlen() : length of a string حساب طول سلسلة
   - strcmp() : comparison of strings

السلاسل Strings

الاستخدام الأكثر شيوعًا للمصفوفات أحادية البعد هو تخزين السلاسل

تعريف السلسلة في C ++، يتم على أنها مصفوفة أحرف منتهية بصفر الرمز ("\ 0").

'H'	'e'	'l'	'l'	'o'	'\0'

تحتوي السلسلة الخالية ، "" ، فقط على حرف النهاية الصفري وتمثله السلسلة الفارغة

قراءة سلسلة من لوحة المفاتيح

كيف تقرأ سلسلة تم إدخالها من لوحة المفاتيح؟

```
#include <iostream>
Using namespace std;
int main()
{
char str[80];
cout << “Enter a string: “;
cin >> str; // read string from keyboard
cout << “Here is your string: “;
cout << str;
return(0);}
```

المشكلة: إدخال السلسلة " "This is a test، البرنامج أعلاه فقط إرجاع "This” ، وليس الجملة بأكملها

السبب: توقف نظام الإدخال / الإخراج في C ++عن قراءة سلسلة عند تمت مصادفة أول حرف مسافة بيضاء.

الحل:

استخدم دالة مكتبة C ++أخرى ، تحصل على gets()

Some C++ Library Functions for Strings

C++ supports a range of string-manipulation functions.

The most common are:

- strcpy() : copy characters from one string to another
- strcat() : concatenation of strings
- strlen() : length of a string
- strcmp() : comparison of strings

```
#include <iostream.h>
#include <cstdio.h>
int main()
{
char str[80]; // long enough for user input?
cout << “Enter a string: “;
gets(str); // read a string from the keyboard
cout << “Here is your string: “;
cout << str << endl;
return(0);
}
```

String Copy: طباعة سلسلة

strcpy(to_string سلسلة الهدف, from_string سلسلة المصدر)

مثال : اكتب برنامج لطباعة العبارة helllo
الى مصفوفة المحارف a [10]

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
char a[10];
strcpy(a, "hello");
cout << a;
return(0);
}
```

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]
h	e	l	l	o	\0	?	?	?	?

حساب طول سلسلة مدخلة من لوحة امفاتيح باستخدام **get(str)** ويجب التوجيه الى مكتبة **cstdio**

```
#include <iostream>
#include <cstdio>
#include <cstring>
int main()
{
char str[80];
cout << “Enter a string: “;
gets(str);
cout << “Length is: “ << strlen(str);
return(0);
}
```

حساب طول سلسلة String Length

strlen(string)

strlen تُرجع طول السلسلة التي يشير إليها str، أي ،عدد الأحرف باستثناء النهاية الفارغة

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
char str[80];
cout << "Enter a string: ";
cin>>str;
cout << "Length is:" << strlen(str);
return(0);
}
```

Concatenation of Strings الحاق سلسلة بسلسلة أخرى

الشكل العام

strcat(string_1, string_2)

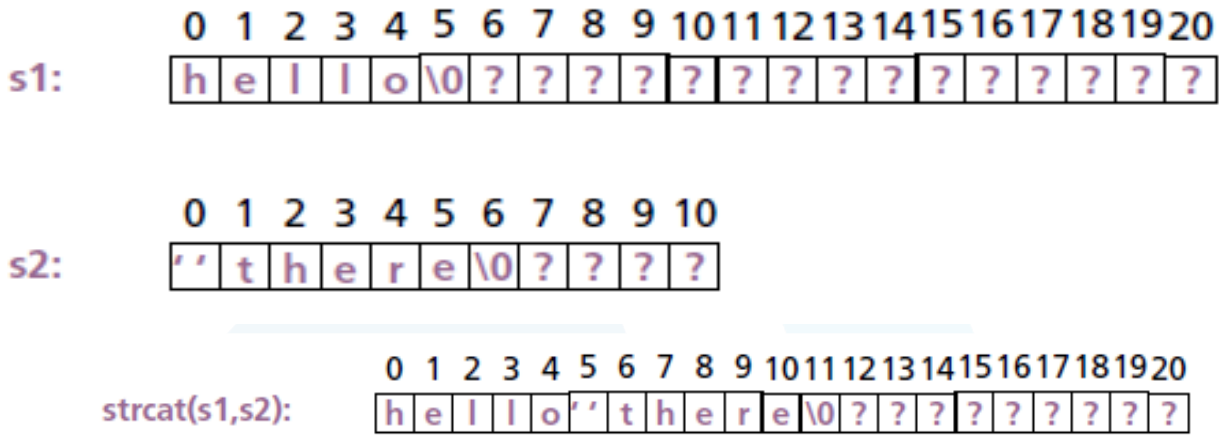
The strcat()

تقوم الوظيفة بإلحاق s2 بنهاية s1. السلسلة s2 لم تتغير

```
#include <iostream.h>
#include <cstdio.h>
#include <cstring.h>
int main()
{
char s1[21], s2[11];
strcpy(s1, “hello”);
strcpy(s2, “ there”);
strcat(s1, s2);
cout << s1 << endl;
cout << s2 << endl;
return(0);
}
```

Output: hello there
there

يجب أن تكون مصفوفة السلسلة الأولى كبيرة بما يكفي لاستيعاب كلا السلسلتين:



strlen(s1concats2) >= strlen(s1) + strlen(s2)

```
// Comparing strings
#include <iostream.h>
#include <cstring.h>
#include <cstdio.h>
int main()
{
char str[80];
cout << “Enter password: “;
gets(str);
if(strcmp(str, “password”)) {
// strings differ
cout << “Invalid password.\n”;}
else cout << “Logged on.\n”;
return(0);
}
```

مقارنة سلسلة بسلسلة أخرى

Comparison of Strings

strcmp(string_1, string_2)

The strcmp(str_1, str_2)

مقارنة سلسلتين نتيجة المقارنة

- str_1 == str_2 : 0
- str_1 > str_2 : positive number
- str_1 < str_2 : negative num

تتم مقارنة السلاسل من حيث المعجمية (أي وفقًا لـترتيب القاموس)

a < aa < aaa < … < b < ba < bb < … < bz < baa < … < abca < abd < ..

مصفوف من سلاسل

مصفوفة من السلاسل هي شكل خاص من مصفوفة ثنائية الأبعاد.

- حجم الفهرس الأيسر يحدد عدد السلاسل.
- حجم الفهرس الصحيح يحدد الطول الأقصى لكل سلسلة .

على سبيل المثال ، يوضح ما يلي مصفوفة من 30 سلسلة ، كل منها بحد أقصى 80 حرفًا (مع حرف إضافي واحد لفاصل الصفر)

```
char string_array[30][81];
```

للوصول إلى سلسلة فردية ، يحدد المرء ببساطة اليسار فقط الفِهرِس:

```
firstString = string_array[0];
sixthString = string_array[5];
```

يستدعي المثال التالي الدالة gets () مع السلسلة الثالثة في المصفوفة:

```
gets(string_array[2]);
```

```
#include <iostream>
#include <cstring>
 #define MAX_SIZE 100 //Maximum size of the string
using namespace std;
 int main()
{
   char str1[1000], str2[1000];
    cout<<"Enter the first string"<<endl;
   cin>>str1;
    cout<<"Enter the second string"<<endl;
   cin>>str2;
    if (strcmp(str1,str2) == 0)
       cout<<"Entered strings are equal"<<endl;
   else
       cout<<"Entered strings are not equal"<<endl;
 return 0;
 }
```

مثال :

اكتب برنامج لمقارنة سلسلتين كل منها من 1000 محرف

اذا كانت السلسلتين متساويتن يطبع الرسالة Entered strings are equal

واذا كانتا مختلفتين يطبع الرسالة Entered strings are not equal

Functions الدوال

1. Introduction
2. Math Library Functions
3. Functions
4. Function Definitions
    1. square Function
    2. maximum Function
5. Function return value
6. Function void
7. Example

هدف الحاضرة الرابعة

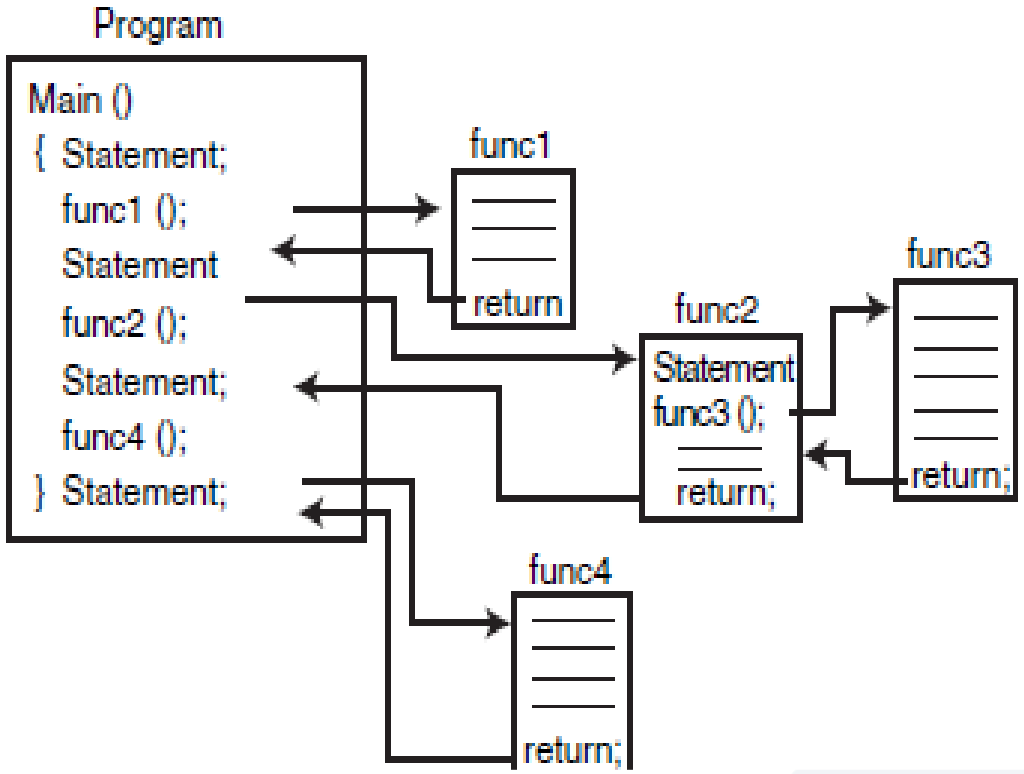
1. مدخل الى الدول
2. دوال المكتبات الجاهزة
    1. دالة مربع عدد
    2. دالة التكعيب
3. تعريف الدول من قبل المستخدم
4. تابع يعيد قيمة
5. تابع لايعيد قيمة void
6. أمثلة محلولة

تستخدم الدوال لتشكيل البرامج. تتم كتابة البرامج عادةً عن طريق تجميع الدوال الجديدة التي تكتبها مع الدوال المعبأة مسبقًا المتوفرة في دليل C++القياسي.

سندرس كلا النوعين من الدوال .

توفر مكتبة C++ القياسية مجموعة غنية من الوظائف لأداء العمليات الحسابية الشائعة ، وسحب السلاسل ، ومعالجة الأحرف ، والإدخال / الإخراج ، والعديد من العمليات المفيدة الأخرى. لأن هذه الدوال توفر العديد من القدرات التي تحتاجها.

الدالة (التوابع) Function



الدوال (التوابع)هي برنامج فرعي يمكنه العمل على البيانات وإرجاع قيمة.
يحتوي كل برنامج C ++على دالة واحدة على الأقل ، .() mainعند بدء تشغيل البرنامج ، يتم استدعاء الدالة () الرئيسية تلقائيًا. () main قد تستدعي دوال أخرى ، قد يستدعي بعضها لا يزال البعض الآخر.

نظرًا لأن هذه الدوال ليست جزءًا من كائن ، فإنها تسمى "عمومي " أي يمكن الوصول إليها من أي مكان في برنامجك..

كل دالة لها اسمها الخاص ، وعندما يتم العثور على هذا الاسم ، يتم تنفيذ فروع البرنامج لجسم تلك الدالة . يشار إلى هذا باسم استدعاء التابع .عندما ينتهي التابع (من خلال مواجهة بيان الإرجاع أو الدعامة النهائية للتابع) ، يُستأنف التنفيذ في السطر التالي من استدعاء التابع

Function التوابع

1 -تساعد الدوال المخزنة في ذاكرة الحاسب على اختصار البرنامج إذ يكتفى باستعادتها باسمها فقط لتقوم بالعمل المطلوب.
2 -تساعد البرامج المخزنة في ذاكرة الحاسب ، أو التي يكتبها المبرمج على تلافي عمليات التكرار في خطوات البرنامج التي تتطلب عملا طويلا وشاقا.
3 -تساعدالدوال الجاهزة على تسهيل عملية البرمجة نفسها.
4 -توفر مساحة من الذاكرة المطلوبة
5- الميزة الرئيسية للتوابع هي تقليل تعقيد البرنامج وإزالة تكرار الكود
المزايا الرئيسية لاستخدام الدوال هي.
اختصار عمليات زمن البرمجة وتنفيذ البرنامج بأسرع وقت ممكن.

استخدم وظائف مكتبة C++القياسية بدلاً من كتابة وظائف جديدة. هذا يمكن أن يقلل من وقت تطوير البرنامج. تمت كتابة هذه الوظائف من قبل خبراء ، تم اختبارها جيدًا وفعالة.

يساعد استخدام الدوال الموجودة في مكتبة C++القياسية في جعل البرامج أكثر قابلية للنقل

تسمح لك الدوال بتكوين وحدات البرنامج. جميع المتغيرات المحددة في تعريفات الوظائف هي متغيرات محلية local variables- لا يمكن الوصول إليها إلا في الدالة التي تم تعريفها فيها.

تحتوي معظم الدوال على قائمة من المعاملات parameters التي توفر الوسائل لتوصيل المعلومات بين الدوال عبر الوسائط في استدعاءات الدوال. معاملات الدالة هي أيضًا متغيرات محلية لتلك الدالة .local variables

في البرامج التي تحتوي على العديد من الدوال، غالبًا ما يتم تنفيذ main كمجموعة من الاستدعاءات للدوال التي تؤدي الجزء الأكبر من عمل البرنامج.

ميزات استخدام الدوال في البرنامج functionalizing.

1. يجعل منهج فرق تسد تطوير البرامج أكثر قابلية للإدارة.
2. إعادة استخدام البرامج –
3. استخدام الدوال الحالية كأجزاء بناء لإنشاء برامج جديدة.
4. تعد قابلية إعادة استخدام البرامج عاملاً رئيسيًا في حركة البرمجة الموجهة للكائنات والتي ستتعلم المزيد عنها عند دراسة اللغات مثل C ++،

يجب أن تقتصر كل وظيفة على أداء مهمة واحدة محددة جيدًا ، ويجب أن يعبر اسم الوظيفة عن هذه المهمة. هذا يسهل التجريد ويعزز إعادة استخدام البرامج.

Math library functions

Function	Description	Example
`sqrt(x)`	square root of *x*	`sqrt(900.0)` is 30.0 `sqrt(9.0)` is 3.0
`cbrt(x)`	cube root of *x* (C99 and C11 only)	`cbrt(27.0)` is 3.0 `cbrt(-8.0)` is -2.0
`exp(x)`	exponential function e^x	`exp(1.0)` is 2.718282 `exp(2.0)` is 7.389056
`log(x)`	natural logarithm of *x* (base *e*)	`log(2.718282)` is 1.0 `log(7.389056)` is 2.0
`log10(x)`	logarithm of *x* (base 10)	`log10(1.0)` is 0.0 `log10(10.0)` is 1.0 `log10(100.0)` is 2.0
`fabs(x)`	absolute value of *x* as a floating-point number	`fabs(13.5)` is 13.5 `fabs(0.0)` is 0.0 `fabs(-13.5)` is 13.5
`ceil(x)`	rounds *x* to the smallest integer not less than *x*	`ceil(9.2)` is 10.0 `ceil(-9.8)` is -9.0
`floor(x)`	rounds *x* to the largest integer not greater than *x*	`floor(9.2)` is 9.0 `floor(-9.8)` is -10.0
`pow(x, y)`	*x* raised to power *y* (x^y)	`pow(2, 7)` is 128.0 `pow(9, .5)` is 3.0

تتيح لك دوال مكتبة الرياضيات إجراء بعض العمليات الحسابية الشائعة.
نستخدم بعضها هنا لتقديم مفهوم الوظائف.

Function	Description	Example
`fmod(x, y)`	remainder of *x*/*y* as a floating-point number	`fmod(13.657, 2.333)` is 1.992
`sin(x)`	trigonometric sine of *x* (*x* in radians)	`sin(0.0)` is 0.0
`cos(x)`	trigonometric cosine of *x* (*x* in radians)	`cos(0.0)` is 1.0
`tan(x)`	trigonometric tangent of *x* (*x* in radians)	`tan(0.0)` is 0.0

c++ توابع مكتبات لغة Functions

تملك لغة C++
عدة توابع وصفوف قياسية تقوم بإنجاز مهمة

تابع الجذر التربيعي (sqrt)

هو أحد التوابع القياسية الرياضية المعرفة في الملف الرأسي <math >

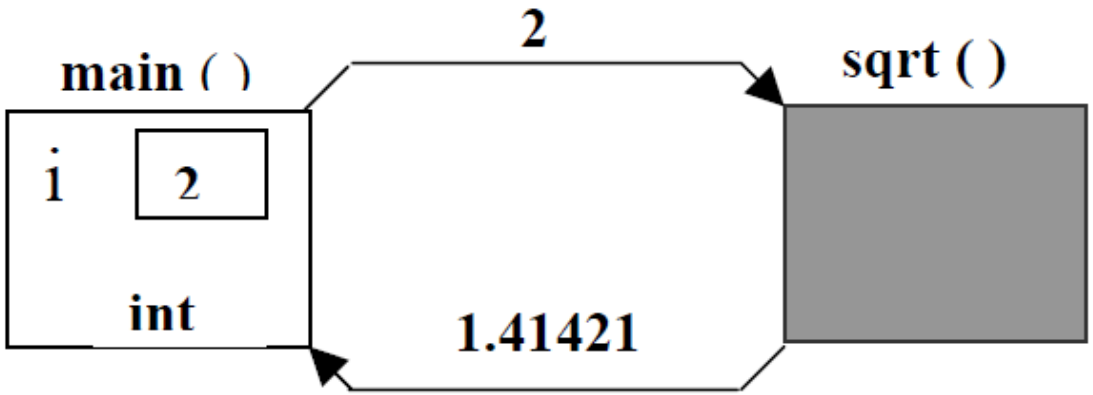
يعيد الجذر التربيعي للعدد (sqrt x).

مثال ليكن لدينا برنامج حساب الجذر التربيعي للأعداد من واحد حتى 5 التالي:

```
#include<iostream >
#include<math>
int main()
{
  for(int i=0;i<6;i++)
    cout<<i<<"\t"<<sqrt(i)<<endl;
}
```

Output

```
0    0
1    1
2    1.41421
3    1.7305
4    2
5    2.2306
```



يعرف التابع كما يلي

التوابع المعرفة من قبل المستخدم (المبتكرة)

Function defined by user

Function definitions

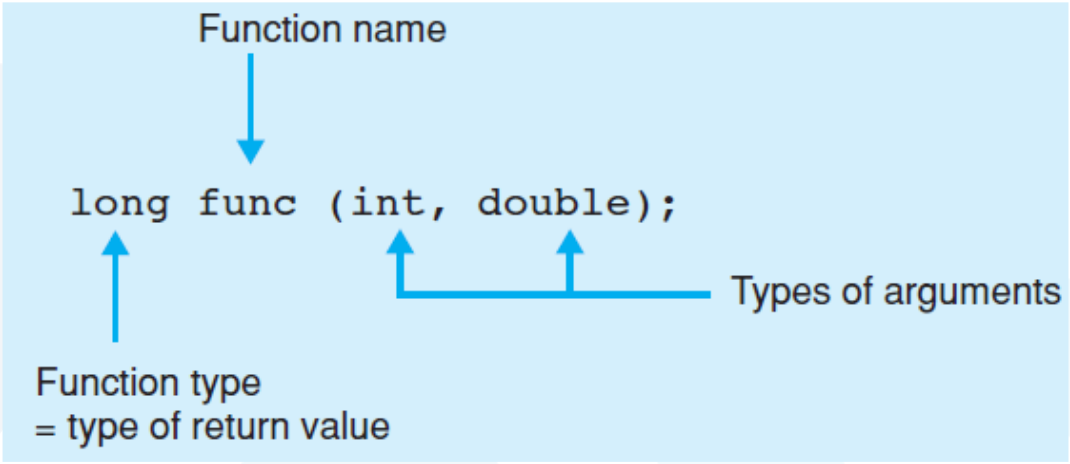
```
Return value type   function name ( parametet_list )
{
declarations and statements ;
}
```

حيث أن : type value return

نوع بيانات القيمة التي سيعيدها التابع

نوع القيمة التي يعيدها التابع باستخدام تعليمة return

Example of a function prototype



function name : هو اسم التابع ينصح بأن يعبر اسم التابع عن مهمته وأن لا يكون طويل اكثر من الضروري وينطبق علية ما ينطبق على أسماء المتغيرات

Parameter_list : قائمة بمعاملات التابع وهي سلسلة المتغيرات التي تأخذ قيمها من الحدود التي تستدعي التابع، ويفصل بين عناصرها فاصلة عادية، ويجب أن تكون محددة النوع

Function Prototype نموذج الدالة

إذا لم تتمكن من اختيار اسم موجز يعبر عن وظيفة ادالة ، فمن المحتمل أن الدالة تحاول أداء العديد من المهام المتنوعة.

لتحليل.decomposition

يكون من الأفضل كسر مثل هذه الدالة إلى دوال أصغر

Format of a Function Definition

The format of a function definition is

```
return-value-type function-name(parameter-list)
{
 statements
}
```

اسم الدالة هو أي معرف صالح.

1- نوع return-value-type القيمة المرتجعة هو نوع بيانات النتيجة التي يتم إرجاعها إلى المتصل.

2- Void يشيرإلى أن الدالة لا تُرجع قيمة.

النموذج الأولي للدالة (يسمى أيضًا تعريف الدالة).

- the number of arguments is correct,،
- the arguments are of the correct type,
- the argument types are in the correct order,

and the return type is consistent with the context in which the function is called.

- عدد الوسائط صحيح
- الوسائط من النوع الصحيح ،
- أنواع الوسيطة بالترتيب الصحيح ، ونوع الإرجاع يتوافق مع السياق الذي يتم فيه استدعاء الدالة.

قائمة المعاملات parameter list

تحدد المعاملات التي تتلقاها الدالة عند استدعائها ويفصل قائمة المعاملات بفواصل.
إذا لم تستقبل الدالة أي قيم ، تكون قائمة المعاملات باطلة.
يجب إدراج بشكل صريح نوع البيانات لكل معامل.

قد تؤدي الدالة التي تتطلب عددًا كبيرًا من المعاملات إلى عدد كبير جدًا من المهام.
تقسيم الدالة إلى دوال أصغر تؤدي مهام منفصلة.
يجب أن يتم احتواء رأس الدالة في سطر واحد إن أمكن.

جسم الدالة body function

تشكل العبارات الموجودة داخل الأقواس جسم الوظيفة ، وهو أيضًا كتلة.
يمكن التصريح عن المتغيرات في أي كتلة ، ويمكن أن تتداخل الكتل (لكن الدوال لا يمكن أن تكون متداخلة)
يعد تحديد دالة داخل دالة أخرى خطأً في بناء الجملة.
اختيار أسماء الدالة ذات المعنى وأسماء المعاملات ذات المعنى يصنع البرامج أكثر قابلية للقراءة ويساعد على تجنب الاستخدام المفرط للتعليقات

يجب أن يتفق النموذج الأولي للدالة ، ورأس الدالة ، واستدعاءات الدالة في عدد ، ونوع ، وترتيب الوسائط والمعاملات ، وفي نوع القيمة المرتجعة

إعادة التحكم من الوظيفة Returning Control from a Function

. توجد ثلاث طرق لإعادة التحكم من دالة مستدعاه إلى النقطة التي تم عندها استدعاء الدالة .
إذا لم تُرجع الدالة نتيجة ، فسيتم إرجاع عنصر التحكم ببساطة عند الوصول إلى القوس الأيمن لنهاية الوظيفة ،
أو عن طريق تنفيذ العبارة;return
إذا كانت الدالة لا تعيد نتيجة ،

return *expression*;

ارجاع قيمة التعبير إلى المتصل

تعمل الوظائف الصغيرة على تعزيز إمكانية إعادة استخدام البرامج

يجب كتابة البرامج كمجموعات من الوظائف الصغيرة. هذا يجعل البرامج أسهل لكتابة وتصحيح وصيانة وتعديل.

مثال تابع لإيجاد مكعب عدد

```
int cube ( int x )
{
return x*x*x ;
}
```

هذا التابع المعرف من قبل المستخدم له جزآن :

تعريف التابع

int (cube int(x))

والذي يحدد فيه نوع القيمة التي يعيدها التابع ، اسمه التابع هنا اسمه cub معاملاته اعداد صحيحة

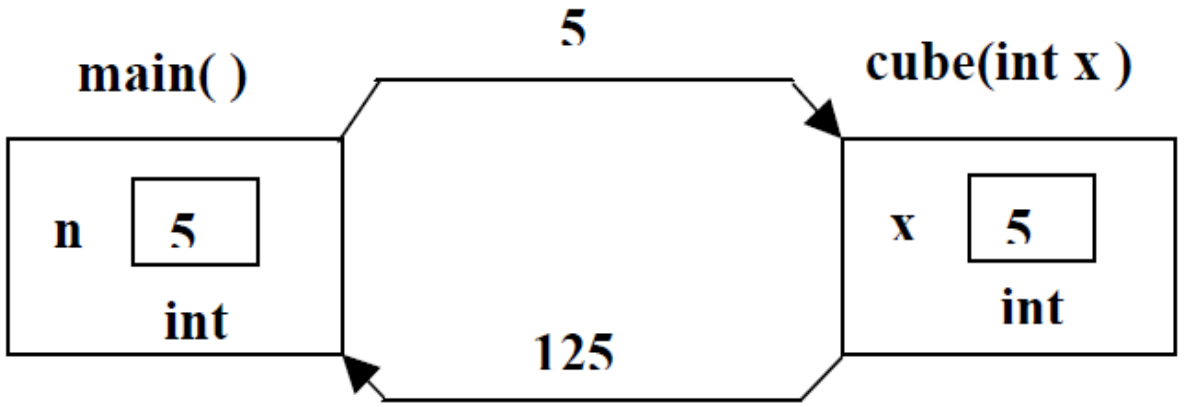
جسم التابع { return x*x*x ; } :

وهو الكتلة التي تحتوي على أوامر التابع ويتبع الإعلان عنه، فجسم التابع يحتوي على الأوامر التي تؤدي وظيفة التابع متضمنا التعليمة

التعليمة Return

لها هدفان : 1 -تنهي التابع .

2 -ترجع قيمة إلى البرنامج الذي تم فيه استدعاء التابع.



فالتابع main( ) أرسل القيمة 5 إلى التابع cube الذي أرجع القيمة 125 إلى التابع main( ).

```
#include<iostream>
int cube(int x)
   {
     return x*x*x;
   }
int main()
{
  int n=1;
  cout<<"enter number to find its cube :";
  while(1)
      {
        cin>>n;
        if(n==0)break;
      cout<<"\t cube("<<n<<")=" <<cube(n) <<endl;
cout<<"enter another number,0 to end :";
      }
Return(0);
}
```

مثال

اكتب برنامج لحساب مكعب عدد صحيح يدخل من لوحة المفاتيح باستخدام التوابع

الحل : تم في البرنامج استخدام حلقة تكرارية محققة دوما
لادخال عدد صحيح طالما n=1
شرط الخروج من الحلقة في حال break n= 0

خرج البرنامج

```
enter number to find its cube :9
cube(9)=729
 enter another number ,0 to end :3
cube(3)=27
```

Example 1:

Write C++ program, to uses a function called square to calculate the number squares from 1 to 10?

```
#include<iostream>
    int square(int y)
        {
          return y*y;
         }
Int main()
{
     for(int x=1;x<=10;x++)
          cout<<square(x)<<" ";
    cout<<endl;
}
```

يتم استدعاء الدالة squre داخل الدالة الرئيسية main وذلك بكتابة squre(x)

مكان استدعاء الدالة square x تقوم الدالة بنسخ قيمة في الوسيط square(y) ثم تقوم بحساب y*y ويتم إرجاع وعرض النتيجة وتتكرر هذه العملية عشر مرات باستخدام حلقة التكرار for

```
#include<iostream.h>
int cube(int);
void main()
{
cube(2);
}
int cube(int x)
{
return x*x*x;
}
```

تعريف الدالة هناك طريقتين الاولى تدعى

الطريقة الأولى وهي انه الدالة تكون مكتوبة definition قبل الاستدعاء اي يكون الاستدعاء من الاسفل الى الاعلى كما هو مكتوب في كل الامثلة السابقة في الدوال.

اما الطريقة الثانية

وهي انه الدالة header كما تدعى احيانا declaration تكون مكتوبة بعد الاستدعاء أي يكون الاستدعاء من الاعلى الى الاسفل كما في المثال التالي.

كما ان استدعاء الدالة ليس شرطا ان main فقط اذ يمكن ان تستدعى من دالة أخرى وتلك الدالة تستدعى من main

```
#include<iostream.h>
int cube(int );
void main()
{
        int n=1;cout<<"enter number to find its cube  :";
        while(1)
                {
                cin>>n;
                if(n==0)
                        break;
                cout<<"\t cube("<<n<<")="<<cube(n)<<endl;
                cout<<"enter another number,0 to end :";
                }
}
int cube(int x)
{
        return x*x*x;
 }
```

الشكل الثاني : يتم التصريح عن الدوال قبل البرمج الرئيسي بعد ذلك يتم كتابة التعريف الكامل للتابع أسفل البرنامج الرئيسي

مثال

اكتب برنامج بلغة C++ يستخدم الدوال لحساب القيمة المتوسطة لعددين صحيحين مدخلين من لوحة المفاتيح من البرنامج الرئيسي

```
#include<iostream>
using namespace std;
float aver (int x1, int x2);
int main( )
{
    float x;
     int num1,num2;
     cout << "Enter 2 positive number \n";
     cin >> num1 >> num2;
     x = aver (num1, num2);
     cout <<"avg= " << x <<endl;

}
float aver (int x1, int x2)
     {
      float z;
       z = ( x1 + x2) / 2.0;
       return ( z);
      }
```

```
#include<iostream>
using namespace std;
float aver (int x1, int x2)
     {
      float z;
       z = ( x1 + x2) / 2.0;
       return ( z);
      }
int main( )
{
    float x;
     int num1,num2;
     cout << "Enter 2 positive number \n";
     cin >> num1 >> num2;
     x = aver (num1, num2);
     cout <<"avg= " << x <<endl;
}
```

أنواع الدوال Types of Functions

يمكن تصنيف الدوال المعرفة من قبل المستخدم بالطرق الثالث التالية بناء على الوسائط الرسمية التي تم تمريرها واستخدام بيان الارجاع ، هناك ثلاث دوال محددة من قبل المستخدم:

1 .يتم استدعاء الدالة دون تمرير أي وسيطة رسمية من الجزء المتصل بالبرنامج وأيضا لا تقوم الدالة بإرجاع أي قيمة إلى الدالة التي تم استدعاؤها.

2 .يتم استدعاء الدالة باستخدام الوسائط الرسمية من جزء الاستدعاء في البرنامج ، لكن الدالةلا تُرجع أي قيمة إلى جزء الاستدعاء.

3 .يتم استدعاء الدالة باستخدام الوسائط الرسمية من جزء الاستدعاء في البرنامج الذي يعيد قيمة إلى بيئة الاستدعاء.

ملاحظة تابع void لايعيد قيمة يجب أن لايسند الى متغير في حال الاستدعاء أو أمر طباعة

مثال :تابع لايعيد قيمة void
برنامج يقوم بطباعة مربع العدد بدون ارجاع قيمة

```
# include <iostream>
void square(int n)
    {
      float value;
      value=n*n;
      cout<<”i=”<<n<<”square=”<<value<<endl;
    }
int main()
{
      int max;
      cout<<”Enter a value for n ?\n”;
      cin>>max;
      for (int i=0;i<=max-1;++i)
      square (i)
}
```

مثال : تابع يعيد قيمة
برنامج يقوم بطباعة مربع العدد باستخدام عبارة ارجاع قيمة

```
# include <iostream>
float square(float n)
      {
      float value;
      value=n*n;
      return (value);
      }
int main()
 {
      float I,max,value;
      max=1.5;
      i=-1.5;
      while (i<=max) {
      value=square(i);
      cout<<”i=”<<i<<”square=”<<value<<endl;
      i=i+0.5;
    }
 }
```

// **Finding the maximum of three integers.**

```
#include <iostream>
 int maximum(int x, int y, int z); // function prototype
 int main(void)
 {
 int number1; // first integer entered by the user
 int number2; // second integer entered by the user
 int number3; // third integer entered by the user
 cout<<("Enter three integers: ");
 cin>>number1>>number2, >>number3;
// number1, number2 and number3 are arguments
 // to the maximum function call
cout<<"Maximum is: " <<maximum(number1,number2, number3)<<endl;
 }
 int maximum(int x, int y, int z)
{
 int max = x; // assume x is largest
  if (y > max) { // if y is larger than max,
 max = y; // assign y to max
 }
  if (z > max) { // if z is larger than max,
 max = z; // assign z to max
 }
  return max; // max is largest value
}
```

جَامعة المَنارة
MANARA UNIVERSITY

مثال

اكتب برنامج لايجاد وطباعة أكبر عدد بين ثلاثة أعداد صحيحة باستخدام التوابع

output

Enter three integers: 22 85 17
Maximum is: 85