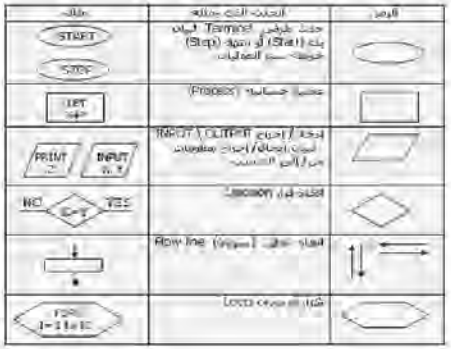
PROGRAMMING

LECTURE # 2

Flowchart

Flowchart

تستخدم مجموعة من الأشكال الرمزية الاصطلاحية المبينة في الجدول التالي :



فوائد استخدام flowchart قبل كتابة البرنامج لمسألة ما :

١. تمكن المبرمج من الإلمام الكامل بالمسألة المراد حلها و السيطرة على كل أجزائها بحيث تساعده على اكتشاف الأخطاء المنطقية (Logic Error)و التي تعتبر من أهم الأخطاء التي تجهد المبرمج.
٢. تساعد بيسر و سهولة على تعديل البرامج الموضوعة بمجرد التفكير.
٣. يعتبر الاحتفاظ برسوم خرائط سير العمليات لحلول مسائل معينة أمراً مهماً إذ يكون مرجعاً عند إجراء تعديلات عليها أو استخدامها لحل مسائل أخرى مشابهة دون الحاجة إلى الرجوع إلى المبرمج الأول باعتبار أن الحلول الأولى قد صيغت في خطوات واضحة بسيطة و مفهومة.
٤. توفير وسيلة مدلسية ومساعدة في كتابة البرامج ذات التفرعات الكثيرة

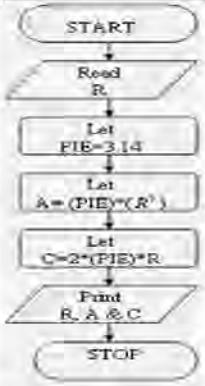
يمكن تصنيف flowcharts بما يلي:

- خرائط التتابع البسيط (Simple sequential Flowchart)
- خرائط التفرع (Branched Flowchart)
- خرائط الدوران البسيط (Loop Flowchart)
- خرائط الدورانات المتداخلة (Nested)

Simple sequential Flowchart

- يخلو هذا النوع من التفرعات Branches
- و الدورانات loops .

Example: Draw a flowchart to find the area and circumference of a circle with a known radius R :



Area= ΠR^2
Circumference=$2\Pi R$
$\Pi \approx 3.14$

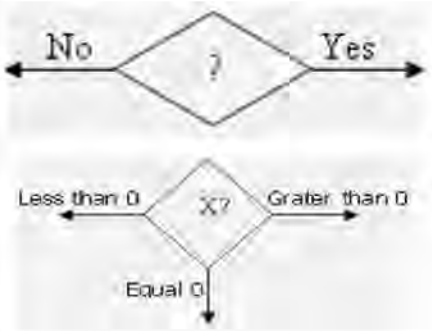
Input: R

Output: R,A,C

الخطوات :

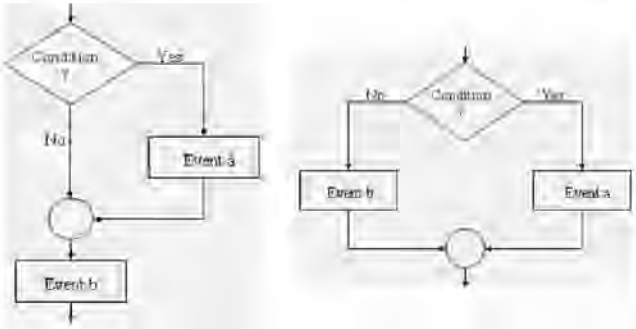
١. ابدأ.
٢. اقرأ قيمة R .
٣. ضع قيمة PIE =٣,١٤
٤. احسب المساحة A من المعادلة A =(PIE)*R*R
٥. احسب المحيط C من المعادلة C =2*(PIE)*R
٦. اطبع قيم كل من C, A, R.
٧. توقف.

Branched Flowchart



ويحدث التفرع في البرامج بسبب الحاجة لاتخاذ قرار أو مفاضلة بين اختيارين أو أكثر، وهناك أسلوبان في تنفيذ القرار .

بشكل عام فإن Branched Flowchart يمكن أن تأخذ إحدى الصورتين الآتيتين :



Example: compare between tow number and print the max :

Input : number1 ,number2

Output: max

الخطوات :

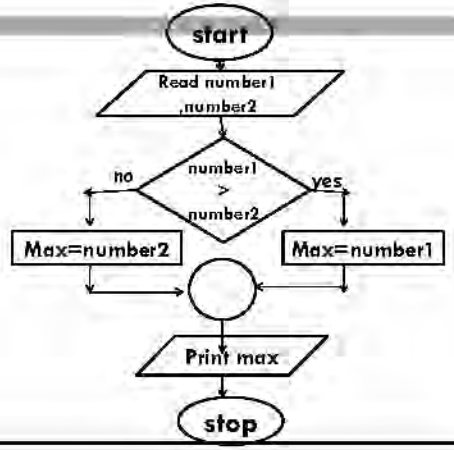
١. ابدا

٢. ادخل العدد الاول number1 والعدد الثاني number2

٣. اذا كان number1 >number2 اجعل max=number1 واذا لا اجعل max=number2

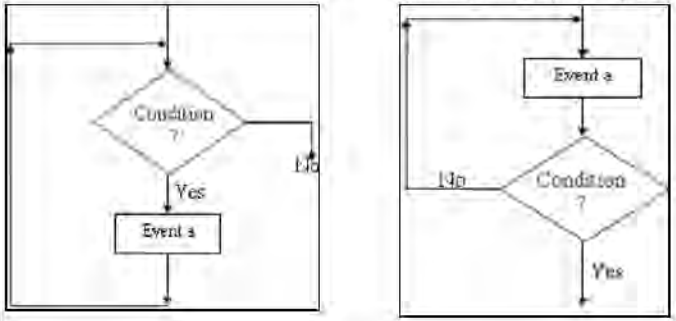
٤. اطبع max

٥. توقف

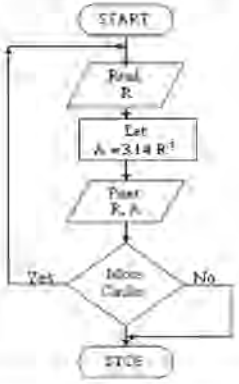


Nested

نحتاج إليها لإعادة عملية أو مجموعة من العمليات في البرنامج عددًا محدودًا أو غير محدود من المرات.
ويكون الشكل العام لمثل هذه الخرائط كما يلي:



Example: Draw a flowchart to find the area a group of circles with know semi-diameter:



Input: R

Output: R, A

- خطوات الحل :
  - ابدأ.
  - اقرأ نصف قطر الدائرة R
  - أوجد مساحة الدائرة A
  - اطبع قيم كل من A, R
  - هل هناك مزيد من الدوائر؟
    فإن كان نعم فعد إلى الخطوه(٢)
    وإن كان لا فعد إلى الخطوه (٦).
  - توقف.

العداد Counter :

- كثير من الأحيان نحتاج في برامج الحاسب الإلكتروني إلى العد Counting، فقد نريد مثلاً أن نعد عدد كل من الطلاب والطالبات ضمن الشعبة، وقد تكون هذه العملية سهلة للإنسان لأنها أصبحت ضمن قدراته العقلية التي يكتسبها من الطفولة، إلا أن الحاسب يحتاج إلى تصميم خوارزمية للعد Counting Algorithmتتضمن خطوات معينة إذا اتبعتها استطاع أن يعد.

ويمكن تحديد الخطوات التي يتبعها الحاسب حتى يتمكن من العد في الخطوات الأساسية:

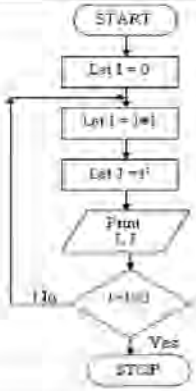
١.اجعل العداد مساويًا للصفر.

٢.اجعل القيمة الجديدة للعداد تساوي القيمة القديمة لها زائد واحد، أي أن:

قيمة العداد (الجديدة)= قيمة العداد (القديمة)+١

٣.كرر الخطوات ابتداء من الخطوة ٢.

Example: Draw a flowchart followed by the computer to print whole numbers from 1 to 100 with squares of it:



Input: nothing

Output:I,J

خطوات الحل :

١. ابدأ.
٢. اجعل I=0
٣. اجعل I=I+1
٤. اجعل J=I²
٥. اطبع I ,J
٦. إذا كانت I=100 اذهب إلى الخطوة ٧ وإلا اذهب إلى الخطوة ٣.
٧. توقف

Summers Algorithm

المجاميع الإجمالية:

- في كثير من الأحيان نحتاج في برامج الحاسب الإلكتروني إلى جمع مجموعة كبيرة من الأعداد التي تمثل معطيات ظاهرة معينة، فمثلاً قد نرغب في إيجاد الوسط الحسابي لأعمار طلاب الجامعة، ولتحقيق هذا أولاً يجب أن نحسب مجموع أعمار الطلاب، وطبعًا ليس عمليًا إعطاء رمز أبجدي لكل عمر طالب فقد نحتاج لأكثر من عشرة الآلاف رمز، في مثل هذه الحالات نصمم خوارزمية معينة للتجميع تسمى خوارزمية التجميع summers Algorithm تتضمن خطوات محددة إذا اتبعها الحاسب استطاع أن يجمع أي كمية من البيانات باستخدام متغيرين اثنين إحداهما هو المتغير الذي نجمعه والآخر هو الجمع الإجمالي (المجمع).

ويمكن تحديد الخطوات التي يجب أن يتبعها الحاسب لتحقيق ذلك في أربع خطوات هي:

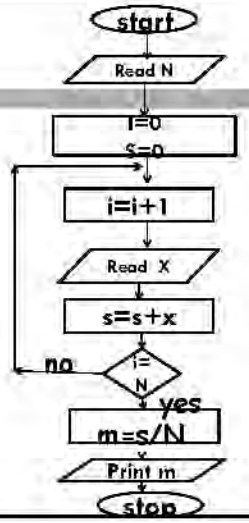
١. اجعل المجمع مساويًا الصفر.
٢. ادخل قيمة واحدة للمتغير.
٣. اجعل القيمة الجديدة للمجمع تساوي القيمة القديمة له زائد القيمة المدخلة للمتغير، أي أن:
قيمة المجمع الجديدة=قيمة المجمع القديمة + آخر قيمة مدخلة للمتغير.
٤. كرر ابتداءً من الخطوة الثانية.

Example : Draw a flowchart to find the mean age of the student :

- **Solution :**
  - الحل: نفترض أن إجمالي عدد الطلاب N
  - ونستخدم عدادًا لرقم كل طالب ونرمز له بالرمز I
  - ونرمز لعمر الطالب بX
  - ونستخدم مجمعًا لأعمار الطلبة ونرمز له بالرمزS
  - ونستخدم الرمز m ليدل على معدل أعمار الطلبة.
- Input : N,X
- Output : m

خطوات الحل:

١. ابدأ
٢. ادخل إجمالي عدد الطلاب (N)
٣. اجعل I=0
٤. اجعل S=0
٥. اجعل I=I+1
٦. ادخل X (عمر الطالب)
٧. اجعل S=S+X
٨. إذا كانت I=N اذهب إلى الخطوة ٩ وإلا اذهب إلى الخطوة ٥
٩. اجعل m=S/N
١٠. اطبع m
١١. توقف



END

C++ Course

Prepared by
Eng. Mohammad Shaker

C++ Section

- Intro
- Variables
- Control Structure
- Functions
- Arrays \ Pointers \ Strings
- Structs

C++ Section

- New Approaches
  - Classes
  - Inheritance
  - Polymorphism
- Template (Optional)
- STL (Optional)
- File processing in .NET Only (Optional)

Programming Languages Wars

Now let's get into the Action !

C++ Preferences & features

- Why C \ C++ ?
  - OOP* Integrated
- Another look :
  - Console \ Interface
    - from "Console" to "Visual" .
    - All Applications need Visual Interacting \ IDE**

* OOP : Object Oriented Programming
** IDE : (Integrated Development Environment)

C++ Preferences & features

- Moore's Law
  - Computer processing power can be doubled every 18-24 months
  - Software with Hardware improves together
    - More complicated Hardware
    - More advanced Software

C \ C++

- The "C" Programming language is a Modular one !
- Why C++ then ?
  - Objects !
    - OOP
  - C \ C++ are portables ones
- Comparison :
  - C : is Action Oriented
    - Procedural
  - C++ : is object oriented
    - Compiler checking
    - Extensible language
      - Class
    - Reusing

C++ Applications

- Microsoft
  - XP , NT , Office ....
- Google
- HP
- Amazon.com

Playing around ...

Program structure: header files and implementation files

- Interface/header File (`.h` file)
  - Contains class *declarations* with free functions and operators *declarations*
  - Useful to programmers, as it is an overview of a component that omits implementation details
- Implementation File (`.cpp` file)
  - Contains free/member function *definitions*, i.e. the implementation code
  - The `.cpp` file is a compilation unit

Program structure: header files inclusion

- A component's entities' declarations (classes and free functions declaration) should *always* be in its header file.
- Other components that use this component will "`#include`" it. There is different syntax to use if you are including a user-defined module, or an existing library module:

```
#include "mymodule.h"
```

  - Quotes indicate a user-defined module
  - The compiler will find it in your project directories

```
#include <mymodule.h>
```

  - < > indicate predefined library header file
  - The compiler will find it in the library directories
  - Using different search paths

Program structure: implementation files

- A component's implementation code should be in a `.cpp` file
- Give the header file and the implementation file the same name
  - `mymodule.h` and `mymodule.cpp`
  - Not enforced by the compiler, but failure to do so is confusing to other programmers
- A component is composed of classes and free functions
- Implementation file must `#include` the module's header file, as it contains the module's classes and data structures declarations
  - If it does not, the compiler will complain that its on entities are undeclared
- `cpp` files contain the executable code
  - Function definitions, including `main()`, free functions, member functions.

Structure of C++ program

```
#include <iostream>

void main ()
{

}
```

Is the same as...

```
#include <iostream>

int main ()
{

    return 0 ;
}
```

Program structure: namespaces

- Namespace: Collection of name definitions inside a program, potentially across different files that share the same namespace
- For example, namespace "`std`" is common in many libraries

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

- Includes entire standard library of name definitions

```
#include <iostream>
using std::cin;
using std::cout;
```

- Can specify just the objects we want
- Can be more efficient, as it avoids including things we don't use

Input and output: streams

- I/O stream objects `cin, cout, cerr`
- Defined in the C++ library called `<iostream>`
- Must have these lines (pre-processor directives) :
  - `#include <iostream>`
    `using namespace std;`
  - Tells C++ to use appropriate library so we can use the I/O objects `cin, cout, cerr` or
  - `#include <iostream>`
    `using std::cout;`
  - To include only the `cout` object

Input and output: streams

- What can be outputted?
  - Any data can be outputted to display screen
    - Variables
    - Constants
    - Literals
    - Expressions (which can include all of above)

```
cout << numberOfGames << " games played.";
```

- 2 values are outputted:
  - value of variable `numberOfGames`,
  - literal string `" games played."`;
- `cout` is a stream, `<<` is a stream operator to output to the stream.
- Similar streams exist for file input/output (see later)

Structure of C++ program

```
#include <iostream>
using namespace::std;   // Note the new namespace

void main()
{
        cout << "we're having fun !";
}
```

Is the same as ...

```
#include <iostream>
using namespace::std; // Note the new namespace

int main ()
{
        cout << "we're having fun !";
        return 0 ; // indicate successful completion
}
```

Structure of C++ program

```
#include <iostream>
using namespace::std; // Note the namespace
void main()
{cout << "we're having fun !";}
```

Is the same as ...

```
#include <iostream>
using namespace::std;  // Note the namespace
int main ()
{ cout << "we're having fun !";  return 0 ;
}
```

Structure of C++ program

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

void main()
{
        cout << "We're having fun !";
        cout << "I need to eat ! :D ";
}
```

```
We're having fun !I need to eat ! :D
```

Structure of C++ program

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

void main()
{
	cout << "we're having fun !" << endl ;
	cout << "I need to eat ! :D " << endl ;
}
```

```
we're having fun !
I need to eat ! :D
```

Structure of C++ program

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

void main()
{
Cout<<"we're having fun !"<< endl<<"I need to eat ! :D " <<
endl ;
}
```

```
we're having fun !
I need to eat ! :D
```

Structure of C++ program

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

void main()
{
Cout<<"we're having fun !"<<endl<<"I need to eat ! :D "
<< endl ;
}
```

```
We're having fun !
I need to eat ! :D
```

Structure of C++ program

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

void main()
{
Cout<<"we're having fun !" << endl << "I need to
eat ! :D "
<< endl ;
}
```

```
Compiles error !
```

Cause every line in C++ language should be ended with a semi colon ";"

Structure of C++ program

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

void main()
{
        cout << "We're having fun !" << "\n" <<  "I need to eat ! :D "
        << "\n" ;
}
```

```
We're having fun !
I need to eat ! :D
```

\n is the same as endl

Structure of C++ program

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

void main()
{
  cout << "We're having fun ! \n I need to eat ! :D \n";
}
```

```
We're having fun !
I need to eat ! :D
```

\n is the same as endl

Comments

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

void main()
{
        cout << "GoGo"; // this is a line !
}
```

```
Compile and run
```

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

void main()
{
        cout << "GoGo"; /* this is a line ! */
}
```

```
Compile and run
```

Comments

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

void main()
{
        cout << "GoGo"; // this is
        a line !
}
```

```
Compiler error
```

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

void main()
{
        cout << "GoGo"; /* this is not a single
        line ! */
}
```

```
Compile and run
```

Comments

- // comment
  - For a one line
- /*
  comments
  */
  - For one line or more
    - multi lines

Escape code

SEQUENCE	MEANING
\n	New line
\r	Carriage return (Positions the cursor at the start of the current line. You are not likely to use this very much.)
\t	(Horizontal) Tab (Advances the cursor to the next tab stop.)
\a	Alert (Sounds the alert noise, typically a bell.)
\\	Backslash (Allows you to place a backslash in a quoted expression.)

Escape code

\'	Single quote (Mostly used to place a single quote inside single quotes.)
\"	Double quote (Mostly used to place a double quote inside a quoted string.)

The following are not as commonly used, but we include them for completeness:

\v	Vertical tab
\b	Backspace
\f	Form feed
\?	Question mark

Error output stream

- Output with `cerr`
  - `cerr` works same as `cout`
  - Provides mechanism for distinguishing between regular output and error output

Keyboard input stream

- `cin` for input (from the keyboard), `cout` for output (to the screen)
- Differences:
  - "`>>`" (extraction operator) points opposite
    - Think of it as "pointing toward where the data goes"
  - Object name "`cin`" used instead of "`cout`"
  - No literals allowed for `cin`
    - Must input to a variable
- `cin >> num;`
  - Waits on-screen for keyboard entry
  - Value entered at keyboard is "assigned" to the variable `num`

Variables

C++ data types

Variables

- Every variable has :
  - Name
  - Type
  - Size
  - Value
- Data Types :
  - Integer , Double , float , char .....

Variables

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

void main()
{
        int i ;
        float j ;
}
```

Variables

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

void main()
{
        int i ;
        int j ;
}
```

Both Working

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

void main()
{
        int i , j ;
}
```

Variables

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

void main()
{
        int i ;
        cin >> i ;
        cout << " i = " << i ;
}
```

Variables

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

void main()
{
        int i  , j ;
        i = 0 ;
        j = 4 ;
}
```

Both Working The SAME!

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

void main()
{
        int i = 0  , j = 4 ;
}
```

Initiating When Declaring

Variables

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

void main()
{
        int i ;
        cin >> i ;
        cout << " i = " << I ;
}
```

Compiler error : 'undeclared "I" identifier'

C++ is case sensitive

Variables

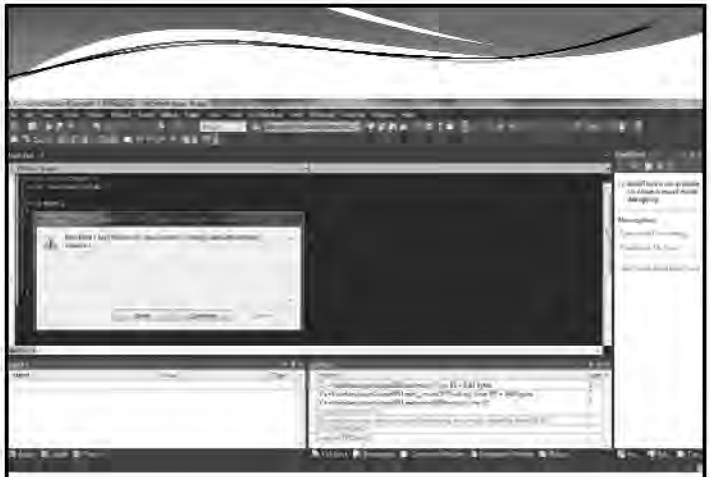
```
#include <iostream>
using namespace::std;

void main()
{
        int i , j ;
        cout << j ;
}
```

Print random value !!!

Variables

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

void main()
{
    int i ;
    cout << i ;
}
```

Runtime Error



Variables

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

void main()
{
        int i , j ;
        cin >> i >> j;
        int sum = i + j;
        cout << sum ;
}
```

Variables

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

void main()
{
        int i , j ;
        cin >> i >> j;
        int sum = i + j;
        cout << " sum = " << sum ;
}
```

Variables

```
// Operating with variables
#include <iostream>
using namespace std;
int main ()
{
  // declaring variables:
  int a, b;
  int result;

  // process:
  a = 5;  b = 2;  a = a + 1;
  result = a - b;

  // print out the result:
  cout << result;

  // terminate the program:
  return 0;
}
```

Variables

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

void main()
{
        int i = 0;
        i = i + 1;
        cout << i;
}
```

```
1
```

Variables

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

void main()
{
	int i = 0;
	i += 1;
	cout << i;
}
```

```
1
```

Variables

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

void main()
{
	int i = 0;
	i++;
	cout << i;
}
```

```
1
```

Variables

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

void main()
{
	int i = 0;
	i--;
	cout << i;
}
```

```
-1
```

Variables

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

void main()
{
	int i , j ;
	i++;
	j--;
	cout << i << j ;
}
```

```
Runtime Error
```

Variables

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

void main()
{
        int i , j ;
        cin >> i ;
        cout << endl ;
        cout << "i = " << i << endl << "\t" ;
        cin >> j ;
        cout << "j = \n" << j ;
        cout << "_____" << endl ;
        cout << "j+i = " << j+i  << "\n" ;
        cout << "2*i = " << 2*i  << "\n" ;
}
```

Assignment operators

- Assignment between objects of the same type is always supported

Assignment operator	Sample expression	Explanation	Assigns
Assume: int c = 3, d = 5, e = 4, f = 6, g = 12;			
+=	c += 7	c = c + 7	10 to c
-=	d -= 4	d = d - 4	1 to d
*=	e *= 5	e = e * 5	20 to e
/=	f /= 3	f = f / 3	2 to f
%=	g %= 9	g = g % 9	3 to g

Increment and Decrement Operators

- Unary operators
  - Adding 1 to or (subtracting 1 from) variable's value
- Increment operator gives the same result of

```
(c=c+1) or (c+=1)
```

- Decrement operator gives the same result of

```
(c=c-1) or (c-=1)
```

Increment and Decrement Operators (cont.)

Operator	Called	Sample expression	Explanation
++	Preincrement	++a	Increment a by 1, then use the new value of a in the expression in which a resides.
++	Postincrement	a++	Use the current value of a in the expression in which a resides, then increment a by 1.
--	Predecrement	--b	Decrement b by 1, then use the new value of b in the expression in which b resides.
--	Postdecrement	b--	Use the current value of b in the expression in which b resides, decrement b by 1.

Examples

98

Example # 1

```
int   x = 10;

cout << "x = " << ++x << endl;
cout << "x = " << x << endl;
```

output # 1

```
x = 11
x = 11
```

Example # 2

```
int   x = 10;

cout << "x = " << x++ << endl;
cout << "x = " << x << endl;
```

output # 2

```
x = 11
x = 11
```

Examples

Example # 1

```
int   x = 10 , y;

y = ++x;

cout << "x = " << x << endl;
cout << "y = " << y << endl;
```

output # 1

```
x = 11
y = 11
```

Example # 2

```
int   x = 10 , y;

y = x++;

cout << "x = " << x << endl;
cout << "y = " << y << endl;
```

output # 2

```
x = 11
y = 11
```

C++ data types

Integral

- `char , short , int , long , bool`
- Boolean :
  - Watch out that the "Boolean" type is an integral one !
  - `bool`
    - `false = 0 , true = any other number`

Integral

- `char`
  - Used to represent character such as :
    - Letters
    - Digits
    - Special symbols
      - '+' , '&' , '$' , '*'
  - Each character is enclosed with **single** quote mark '' and **not double** ones " "
  - Space is represented by ' ' with space between them .
- ASCII & EBCDIC

C++ data types

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

void main()
{
        char c1 = 'd' , c2 ;
        cout << c2 << c1 << endl   ;

}
```

```
 d
Press any key to continue
```

Note *char*:

- **The default value for a char is** NULL **represented as** SPACE' '
  ... but it's not a space, it's a NULL **!**
- **The char has ' ' and not " "**

C++ data types

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

void main()
{
        bool b1 , b2 ;
        if (3 <= 2)
        {
                b1 = false ;
                b2 = true  ;
        }
        else
        {
                b1 = 52 ;
        }
        cout << b1 << " - " << b2 << endl ;
}
```

```
1 - 0
```

Note that:

- **A numeric value is printed when printing a "boolean"**
- **The default value for** bool **type is false (0)**

C++ data types

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

void main()
{
        bool b1 , b2 ;
        if (3 <= 2)
        {
                b1 = false ;
                b2 = true  ;
        }
        else
        {
                b1 = 52 ;
        }
        cout << b1 << " - " << b2 <<
}
```

Evaluate Compiler
if (3 <= 2)

Parsing Steps
- if (3 <= 2)
- 3 <= 2
- if (3 <= 2)

```
1 - 0
```

Note that:

- **A numeric value is printed when printing a "boolean"**
- **The default value for** bool **type is false (0)**

C++ data types

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

void main()
{
        bool b1 , b2 ;
        if (3 <= 2)
        {
                b1 = false ;
                b2 = true ;
        }
        else
        {
                b1 = 53 ;
        }
        cout << b1 << " - " << b2 << "
}
```

Evaluate Compiler
if (3 <= 2)

Parsing Steps

- 3 <= 2
- if (3 <= 2)

1 - 0

Note that:

- A numeric value is printed when printing a "boolean"
- The default value for bool type is false (0)

C++ data types


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

void main()
{
        bool b1 , b2 ;
        if (3 <= 2)
        {
                b1 = false ;
                b2 = true ;
        }
        else
        {
                b1 = 53 ;
        }
        cout << b1 << " - " << b2 << "
}
```


Evaluate Compiler
if (3 <= 2)

Parsing Steps

- 3 <= 2
- if (3 <= 2)

1 - 0

Note that:

- A numeric value is printed when printing a "boolean"
- The default value for bool type is false (0)

C++ data types


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

void main()
{
        bool b1 , b2 ;
        if (3 <= 2)
        {
                b1 = false ;
                b2 = true ;
        }
        else
        {
                b1 = 53 ;
        }
        cout << b1 << " - " << b2 << "
}
```


Evaluate Compiler
if (3 <= 2)

Parsing Steps

- 3 <= 2
- false
- if (3 <= 2)

1 - 0

Note that:

- A numeric value is printed when printing a "boolean"
- The default value for bool type is false (0)

C++ data types


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

void main()
{
        bool b1 , b2 ;
        if (3 <= 2)
        {
                b1 = false ;
                b2 = true ;
        }
        else
        {
                b1 = 53 ;
        }
        cout << b1 << " - " << b2 << "
}
```


Evaluate Compiler
if (3 <= 2)

Parsing Steps

- if (3 <= 2)

1 - 0

Note that:

- A numeric value is printed when printing a "boolean"
- The default value for bool type is false (0)

C++ data types

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

void main()
{
        bool b1 , b2 ;
        if (3 <= 2)
        {
                b1 = false ;
                b2 = true ;
        }
        else
        {
                b1 = 53 ;
        }
        cout << b1 << " - " << b2 << 
}
```

Evaluate Compiler
if (3 <= 2)

Parsing Steps

- if [false]

1 - 0

Note that:

- A numeric value is printed when printing a "boolean"
- The default value for bool type is false (0)

C++ data types

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

void main()
{
        bool b1 , b2 ;
        if (3 <= 2)
        {
                b1 = false ;
                b2 = true ;
        }
        else
        {
                b1 = 53 ;
        }
        cout << b1 << " - " << b2 << 
}
```

Evaluate Compiler
if (3 <= 2)

Parsing Steps

- Jump up to else

1 - 0

Note that:

- A numeric value is printed when printing a "boolean"
- The default value for bool type is false (0)

C++ data types

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

void main()
{
        bool b1 , b2 ;
        if (3 <= 2)
        {
                b1 = false ;
                b2 = true ;
        }
        else
        {
                b1 = 53 ;
        }
        cout << b1 << " - " << b2 << 
}
```

Evaluate Compiler
if (3 <= 2)

Parsing Steps

- Jump up to else !!

1 - 0

Note that:

- A numeric value is printed when printing a "boolean"
- The default value for bool type is false (0)

C++ data types

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

void main()
{
        bool b1 , b2 ;
        if (3 <= 2)
        {
                b1 = false ;
                b2 = true ;
        }
        else
        {
                b1 = 53 ;
        }
        cout << b1 << " - " << b2 << 
}
```

Evaluate Compiler
if (3 <= 2)

Parsing Steps

- cout << b1 << " - " << b2 << endl ;

1 - 0

Note that:

- A numeric value is printed when printing a "boolean"
- The default value for bool type is false (0)

C++ data types

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

void main()
{
        bool b1 , b2 ;
        if (3 <= 2)
        {
                b1 = false ;
                b2 = true ;
        }
        else
        {
                b1 = 52 ;
        }
        cout << b1 << " - " << b2 <<
}
```

Evaluate Compiler
if (3 <= 2)

Parsing Steps

1- cout << b1 << " - " << b2 << endl ;

```
1 - 0
```

Note that:
- A numeric value is printed when printing a "boolean"
- The default value for bool type is false (0)

C++ data types

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

void main()
{
        bool b1 , b2 ;
        if (3 <= 2)
        {
                b1 = false ;
                b2 = true ;
        }
        else
        {
                b1 = 52 ;
        }
        cout << b1 << " - " << b2 <<
}
```

Evaluate Compiler
if (3 <= 2)

Parsing Steps

1- cout << b1 << " - " << b2 << endl ;

Printing the integral type in the form of number

```
1 - 0
```

Note that:
- A numeric value is printed when printing a "boolean"
- The default value for bool type is false (0)

C++ data types

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

void main()
{
        bool b1 , b2 ;
        if (3 <= 2)
        {
                b1 = false ;
                b2 = true ;
        }
        else
        {
                b1 = 52 ;
        }
        cout << b1 << " - " << b2 <<
}
```

Evaluate Compiler
if (3 <= 2)

Parsing Steps

1- cout << << " - " << b2 << endl ;

Printing the integral type in the form of number

```
1 - 0
```

Note that:
- A numeric value is printed when printing a "boolean"
- The default value for bool type is false (0)

C++ data types

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

void main()
{
        bool b1 , b2 ;
        if (3 <= 2)
        {
                b1 = false ;
                b2 = true ;
        }
        else
        {
                b1 = 52 ;
        }
        cout << b1 << " - " << b2 <<
}
```

Evaluate Compiler
if (3 <= 2)

Parsing Steps

1- cout << << " - " << << endl ;

Printing the integral type in the form of number

```
1 - 0
```

Note that:
- A numeric value is printed when printing a "boolean"
- The default value for bool type is false (0)

C++ data types

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

void main()
{
        bool b1 , b2 ;
        if (3 <= 2)
        {
                b1 = false ;
                b2 = true ;
        }
        cout << b1 << " - " << b2 << "
}
```

Evaluate Compiler
if (3 <= 2)

Parsing Steps

Printing the [illegible] type in the form of [illegible]

```
1 - 0
```

Note that:

- A numeric value is printed when printing a "boolean"
- The default value for bool type is false (0)

Floating point data types

- float , double :
  - float ... 4 bytes / double ... 8 bytes
    - float has a single precision
    - double has a double precision
  - double = long double (in new compilers)
  - The size of float , double , long double are machine dependent .

double data type

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

void main()
{
        double d = 0.4;
        cout << d << endl ;
        system("pause");
}
```

```
0.4
```

double data type

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

void main()
{
        double d = 0.0;
        cout << d << endl ;
        system("pause");
}
```

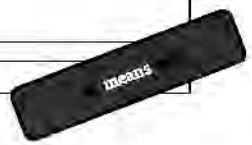
```
0
```

No need to print .0

double data type

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

void main()
{
        double d = .0;
        cout << d << endl ;
        system("pause");
}
```

```
0
```



float data type

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

void main()
{
        float f = 1.2;
        cout << f << endl ;
        system("pause");
}
```

```
1.2
```

float data type

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

void main()
{
        float f = 1.2f;
        cout << f << endl ;
        system("pause");
}
```

```
1.2
```

String data type

- String
  - **Sequence** of **Zero** or more character
  - The position of the first character is **0** and not **1** !
  - Length of the string is the number of characters in it .
  - Represented by enclosed with double quote marks " "
  - (***we will see it later***)

String **data type**

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

void main()
{
	string s = "Hello !";
	cout << s ;
	system("pause");
}
```

```
Compiler error ! See how to deal with it later ;)
```

String **data type**

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

void main()
{
	string s = "Hello !";
	cout << s ;
	system("pause");
}
```

cout
deals with numbers
(int, bool, char ...) only!

```
Compiler error ! See how to deal wi      later ;)
```

String **data type**

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

void main()
{
	string s = "Hello !";
	cout << s ;
	system("pause");
}
```

cout
deals with numbers
(int, bool, char ...) only!

```
Compiler error ! Se
```

Remember that
char, bool are integral types!

Data types: simple types size, range and precision

short (also called short int)	2 bytes	−32,767 to 32,767	Not applicable
int	2 bytes	−2,147,483,647 to 2,147,483,647	Not applicable
long (also called long int)	4 bytes	−2,147,483,647 to 2,147,483,647	Not applicable
float	4 bytes	approximately $\pm 10^{-38}$ to $\pm 10^{38}$	7 digits
double	8 bytes	approximately $\pm 10^{-308}$ to $\pm 10^{308}$	15 digits

Data types: simple types size, range and precision

long double	8 bytes	approximately -10^{312} to 10^{312}	15 digits
char	1 byte	All ASCII characters (Can also be used as an integer type, although we do not recommend doing so.)	Not applicable
bool	1 byte	true, false	Not applicable

The values listed here are only sample values to give you a general idea of how the types differ. The values for any of these entries may be different on your system. *Precision* refers to the number of meaningful digits, including digits in front of the decimal point. The ranges for the types float, double, and long double are the ranges for positive numbers. Negative numbers have a similar range, but with a negative sign in front of each number.

Cracking The Code Section

Code Cracking

```
#include <iostream>

int main ()
{
      return 2 ;
}
```

```
// Will compile & Run !
// Any number will do the job
```

Code Cracking

```
#include <iostream>

int main ()
{
      return ;
}
```

```
// compiler error
// must have a return value
```

Code Cracking

```
#include <iostream>

void main()
{
        return 2.54 ;
}
```

```
Compiler error
```

Code Cracking

```
#include <iostream>

int main ()
{

}
```

```
// Will compile & Run, although no return type
```

Watch out !

- **Just the `main` function can be acceptable without the `"return"` statement when returning a value in its header `"int"`**
- **No other function with return value is acceptable without the `"return"` statement when returning a value**

Code Cracking

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

void main()
{
        cout << "GoGo"; /* this is a line !!
}
```

```
Compiler error : missing
```

Code Cracking

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

void main()
{
        cout >> "GoGo"; /* this is a line !! */
}
```

```
Compiler error
```

Code Cracking

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

void main()
{
	int i , int j ;
}
```

```
Compiler error
```

Code Cracking

```
#include <iostream>

int main()
{
	int i_j ;
	return -0001223333 ;
}
```

```
Kello tmam :D
```

Code Cracking

```
#include <iostream>

Int main()
{
	int i_j ;
	return -12323333 ;
}
```

```
Compiler error "Int" I is capital letter
```

Code Cracking

```
#include <iostream>

void main()
{
	int i_j ;
	cout << i_j;
}
```

```
Compiler error : Unknown "cout" Undeclared identifier
Missing using namespace std
```

Code Cracking

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

void main()
{
        bool b1  , b2 ;
        if (b1 >= b2)
        {
                b1 = 0 ;
                b2 = 1.3 ;
        }
        cout << b1 << b2 << endl ;
}
```

```
0
```

Code Cracking

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

void main()
{
        bool b1  , b2 ;
        if (b1 >= b2)
        {
                b1 = not(0) ;
                b2 = 1.3 ;
        }
        cout << b1 << b2 << endl ;
}
```

```
Compiler error:  'not' undeclared (need '#include <iso646.h>' or '-fext-numeric-literals' to enable)
```

Code Cracking

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

void main()
{
        bool b1  , b2 ;
        if (b1 >= b2)
        {
                b1 = !(0) ;
                b2 = 0 ;
        }
        cout << b1 << b2 << endl ;
}
```

```
10
```

Code Cracking

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

void main()
{
        bool b1=1  , b2 ;
        if (b1 != b2)
        {
                b1 = !(1-2312) ;
                b2 = !!b1 ;
        }
        cout << b1 << b2 << endl ;
}
```

```
01
```

Code Cracking

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

void main()
{
        bool b1  , b2 ;
        if (b1 >= !(23423))
        {
                b1 = !(1-1231233) ;
                b2 = !!b1 ;
        }
        cout << b1 << b2 << endl ;
}
```

01

Code Cracking

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

void main()
{
        double b1  , b2 ;
        if (b1 !=!(34))
        {
                b1 = !(1-1231233) ;
                b2 = !b2 + b1 ;
        }
        cout << b1 << b2 << endl ;
}
```

Compiler error ? Compile & run ? Output ?

00

Code Cracking

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

void main()
{
        double b1  , b2 ;
        bool f ;
        if (b1 !=!(f))
        {
                b1 = !(1-1231233) ;
                b2 = !b2 + b1 ;
        }
        cout << b1 << b2 << endl ;
}
```

Compiler error ? Compile & run ? Output ?

01

Code Cracking

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

void main()
{
        double i1 = 9.3 , i2 = 3.6 ;
        bool b1 , b2 ;
        if (i1 > i2)
        {       b1 = -1 ;       b2 = true ;     }
        else
        {       b1 = 53 ;       }
        cout << b1 << "\t -\t" << b2 <<"\n\n\n\t\t - "<<b2<< endl;
;
}
```

Code Cracking

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

void main()
{
	char c1 = 'd' , c2 ;
	bool b1 ;
	cout << int(c2) << " - c1\t - " << int(b1) << endl    ;

}
```

```
0 - c1      - 0
Press any key to continue . . .
```

Code Cracking

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

void main()
{
	char c1 = "d" , c2 ;
	bool b1 ;
	cout << int(b1) << endl    ;

}
```

```
Compiler error " "
```

Code Cracking

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

void main()
{
    cout << "I'm number 4 or 77, I don't know :D" << endl;
    ;
    ;
    ;
    ;
    system("pause");
}
```

```
I'm number 4 or 77, I don't know :D
```

www.msdn.com

Operators

- Assign "="
  - `b = 5 ;` `// a = 5`
  - `a = 3 + (b = 6 ) ;` `// a = 8`
  - `a = b = 5 ;` `// legal a = b = 5`

Arithmetic Operators

- Arithmetic Operation

C++ operation	C++ arithmetic operation	Algebraic expression	C++ expression
Addition	+	f + 7	p + c
Subtraction	-	p – c	p - c
Multiplication	*	bm or b · m	b * m
Division	/	x / y or $\frac{x}{y}$ or x ÷ y	x / y
Modulus	%	r mod s	r % s

- Arithmetic expressions appear in straight-line form
- Parentheses () are used to maintain priority of manipulation

Operators

- / : just between ***integers***
  - The result of division depends on the type of the two operands
    - If one of the operands is float
      - The result is float typed
    - Otherwise
      - The result is integer typed
  - `13 / 4 = 3`
  - `13.0 / 4 = 3.25`
  - `13 / 4.0 = 3.25`
  - `13.0 / 4.0 = 3.25`

Operators

- %: Applied to **integer** types **only**
  - Operands :
    - Both positive
      - Result is positive
    - One or both negative
      - Result is machine-dependent

Operators

```
11.3 % 4 ;
```

```
Compiler error … left side is a double
```

```
22 % 7 ;
```

```
1
```

```
21 % 7 ;
```

```
0
```

```
2%6 ;
```

```
2
```

Operators

```
23%-6 ;
```

```
5
```

```
-23%-6 ;
```

```
-5
```

```
2%-6 ;
```

```
2
```

Arithmetic Operators Precedence

Operator(s)	Operation(s)	Order of evaluation (precedence)
( )	Parentheses	Evaluated first. If the parentheses are nested, the expression in the innermost pair is evaluated first. If there are several pairs of parentheses "on the same level" (i.e., not nested), they are evaluated left to right.
* / %	Multiplication Division Modulus	Evaluated second. If there are several, they are evaluated left to right.
+ -	Addition Subtraction	Evaluated last. If there are several, they are evaluated left to right.

Example

- The statement is written in algebra as

$$z = pr\,\%\,q + w\,/\,(x - y)$$

- How can we write and evaluate the previous statement in C++ ?

```
z = p * r % q + w / (x - y);
```

6 2 3 5 4 1

Example

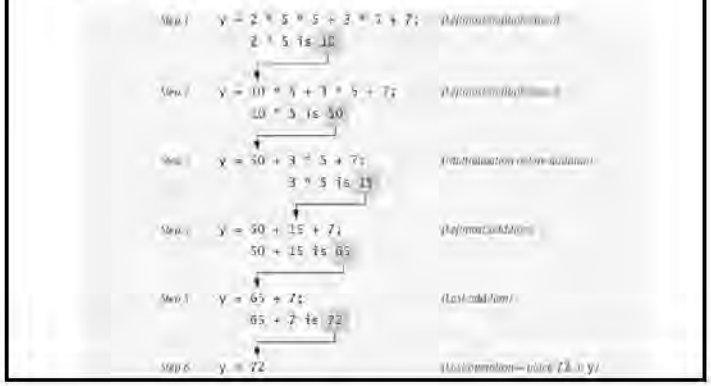
Algebra: $m = \frac{a+b+c+d+e}{5}$

C++: `m = ( a + b + c + d + e ) / 5;`

Algebra: $y = mx + b$

C++: `y = m * x + b;`

Example:



Exercise

- State the order of evaluation of the operators in each of the following C++ statements and show the value of x after each statement is performed.
- `x = 7 + 3 * 6 / 2 - 1;`
- `x = 2 % 2 + 2 * 2 - 2 / 2;`
- `x = ( 3 * 9 * ( 3 + ( 9 * 3 / ( 3 ) ) ) );`

Answer:

- ١٥
- ٣
- ٣٢٤

Type Conversion

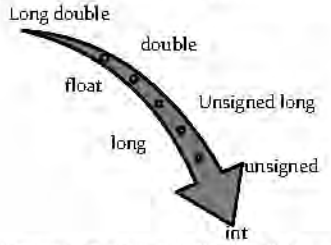
- Used when different data types are used together
  - Expression
    - Arithmetic
    - Relational
  - Assignment operations
  - Parameter passage (through functions)
  - Return type (From Value-Returning functions)

Type Conversion

- Two types of Conversion :
  - Implicit
    - *Promotion* (widening , upcast)
      - Promote a smaller type to a larger one
        - Thus , no loss of data
      - How to promote ?
        - 1st :Promotion
        - 2nd : Arithmetic
    - *Demotion*
      - Demote a larger type to a smaller one
        - Thus , *maybe* loss of data !
  - Explicit

Type Conversion – Implicit promotion

- How to promote ?
  - bool , int , char , short , enum
    - Promoted to int
    - Integral values that can't be represented as int are promoted to unsigned
  - if the expression still of a mixed type , the follow will be applied :



- Notice :
  - When using "=" operator , the expression on the right will be converted to the type of the left

Type Conversion – Implicit promotion

```
float x = 7 ;                    // "int" is promoted to float
```

```
3.4 + 'b'                        // b "char" is promoted to double
```

```
3.43265L + 'b'                   // b "int" is promoted to long double
```

```
int i1 ;
char i2 ;
float i3 ;
i1 + i2 + i3
// i1 , i2 are promoted to float
```

Type Conversion – Implicit promotion

```
int i1 ;
int i2 ;
double i3 ;
i1 + i2 + i3
// i1 , i2 are promoted to double
```

```
int i1 ;
double i3 ;
i1 + 'N' + i3
// i1 , N are promoted to double
```

Type Conversion – Implicit demotion

```
int x = 7.4 ;          // 7 "double" is demoted to int = 7
```

Note the loss of data when demotion sometimes !

Type Conversion – Explicit Casting

```
float x = float(7) ;      // becomes 7.0
```

- Is the same as

```
float x = (float)7 ;      // becomes 7.0
```

TypeName (Expression)
(TypeName) Expression

```
int (7.6)     // becomes 7
(int) 7.6     // becomes 7
```

Type Conversion – Explicit Casting

```
7/4
// 1
float (7/4)
// 1.0
float(7) / float(4)
// 1.75
double(7) / double(4)
// 1.75
```

Type Conversion – Explicit Casting

- Types of explicating Casting
  - static_cast
  - dynamic_cast
  - const_cast
  - reinterpret_cast

Type Conversion – Explicit Casting

- Converting char to int
- Let's 1st see this ...

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

void main()
{
        char c ;
        int i ;
        i = c ;         // implicit casting
        cout << i << endl ;
}
```

```
0
```

Type Conversion – Explicit Casting

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

void main()
{
        char cece = 'c' ;
        int i ;
        i = cece ;              // implicit casting
        cout << i << endl ;
}
```

```
99
```

Type Conversion – Explicit Casting

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

void main()
{
        char cece = 'c' ;
        int i = cece ;                  // explicit casting
        cout << i << endl ;
}
```

```
99
```

Type Conversion – Explicit Casting

- Now , ***Explicit*** type casting is like that ....

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

void main()
{
        char cece = 'c' ;
        int i ;
        i = int(cece) ;
        cout << i << endl ;
}
```

99

Type Conversion – Explicit Casting

- Or like that ....

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

void main()
{
        char cece = 'c' ;
        int i ;
        i = (int)cece ;
        cout << i << endl ;
}
```

99

Type Conversion – Explicit Casting

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

void main()
{
        char c = 'c' ;
        int i ;
        i = int(c - 'd') ;
        cout << i << endl ;
}
```

-1

Type Conversion – Explicit Casting

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

void main()
{
        char c = 'c' ;
        int i ;
        i = int(c - 'c') ;
        cout << i << endl ;
}
```

0

Type Conversion – Explicit Casting

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

void main()
{
        char c = '2' ;
        int i ;
        i = int(c - '2') ;
        cout << i << endl ;
}
```

0

Type Conversion – Explicit Casting

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

void main()
{
        char c = '2' ;
        int i ;
        i = int(c - '0') ;
        cout << i << endl ;
}
```

2

Explicit Casting – static_cast

- At compile time !
  - Thus , Type & object should be fully known at compile time

Explicit Casting – static_cast

- We were used to do that

```
#include <iostream>
using namespace::std;
void main()
{
        int i = 25;     long l ;        float f ;
        l = i ;
        f = i ;
}
```

- Now , we can do that ... (Not necessary .. Just highlighting the cast)

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

void main()
{
        int i = 25;     long l ;        float f ;
        l = static_cast<long>(il) ;
        f = static_cast<float>(il) ;
}
```

Explicit Casting – static_cast

- Now , let's see this ...

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

void main()
{
        int i1 = 25  ;
        long i2  ;
        float i3 ;
        i1 = i2 ;
        i1 = i3 ;
        i2 = i3 ;
}
```

```
Compile : Run
```

```
Two warnings encountered :
```

Explicit Casting – static_cast

- To eliminate warning ... we use "**static_cast**"

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

void main()
{
        int i1 = 25  ;
        long i2  ;
        float i3 ;
        i1 = static_cast<long>(i2) ;
        i1 = static_cast<float>(i3) ;
        i2 = static_cast<long>(i3) ;
}
```

```
Compile : Run
```

```
No more Warnings :
```

Indicating that we are aware of the warnings

Explicit Casting – static_cast

- Returning to our first example

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

void main()
{
        int i1 = 25  ;
        long i2  ;
        float i3 ;
        i1 = i2 ;
        i1 = i3 ;
        i2 = i3 ;
}
```

```
Compile : Run But warnings encountered
```

```
Warnings should be "as expected" coming following conversions
        i1 = i2 ;
        i1 = i3 ;
        i2 = i3 ;                       "Right"
```

Explicit Casting – static_cast

```
But in fact the warnings are for them :
        i1 = i3 ;
        i2 = i3 ;
```

Explicit Casting – static_cast

```
But in fact the warnings are for just :
      i1 = i3 ;
      i2 = i3 ;
```

Why?

C++ data types

enum

- **Intro**
  - Deal with limited range of constants (set of constants)
    - Set of colors
  - Used in place of the actual **integer values**
  - C++ allows creation of a new **simple** type by listing (enumerating) all the ordered values in the domain of a **new** type

Syntax :

```
enum TypeName { value1 , value2 , value3 , … };
```

enum

```
#include <iostream>

void main()
{
 enum Cars { Nissan , BMW , Mercedes , Ferrari , Renault};
  // ALL possible values between the braces
}
```

```
#include <iostream>
enum Cars { Nissan , BMW , Mercedes , Ferrari , Renault};
   // ALL possible values between the braces
// Global Variable
void main()
{
}
```

enum

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

void main()
{
        enum Cars { Nissan , Nissan , Mercedes , Ferrari ,Renault};
}
```

Compiler error

Values must have UNIQUE names

enum

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

void main()
{
        enum Cars { Nissan , BMW , Mercedes , Ferrari ,Renault};
}
```

Nissan < BMW < Mercedes < ….

The default Values are ordered from 0 upward

Nissan = 0
BMW = 1
Mercedes = 2
…

enum

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

void main()
{
   enum Cars { Nissan = 1 , BMW , Mercedes , Ferrari ,Renault};
}
```

Nissan < BMW < Mercedes < ….

The Values are ordered from 1 upward

Nissan = 1
BMW = 2
Mercedes = 3
…

enum

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

void main()
{
enum Cars { Nissan = 1 , BMW , Mercedes = 5 , Ferrari ,Renault};
}
```

Compiler error ?

No .. It's not !

The Values are incremented by 1 upward

Nissan = 1 , BMW = 2
Mercedes = 5 , Ferrari = 6 , Renault = 7

enum

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

void main()
{
 enum Cars {Nissan = 1, BMW , Mercedes = 7, Ferrari , Renault  };
 cout << Renault ;
}
```

```
9
```

enum

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

void main()
{
enum Cars { Nissan = 1 , BMW , Mercedes = BMW + 4 , Ferrari
,Renault = Ferrari + 2};
}
```

```
Compiler error ?
```

```
No ! It's not
```

The Values are incremented by 1 upward

```
Nissan = 1  , BMW = 2
Mercedes = 6 , Ferrari = 7 , Renault = 9
```

enum

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

void main()
{
enum Cars { Nissan = 1 , BMW , Mercedes = BMW + 4 , Ferrari
,Renault = Renault + 2};
}
```

```
Compiler error , Renault undeclared identifier
```

Look of how compiler think ;) It's an automaton

enum

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

void main()
{
enum Cars { Nissan = 1 , BMW , Mercedes = 1 , Ferrari ,Renault };
}
```

```
Compiler error ?
```

```
No ! It's not
```

The values are ordered from 1 upward

```
Nissan = 1  , BMW = 2
Mercedes = 1 , Ferrari = 2 , Renault = 3
```

enum

- Declaring variables from the "enumerated" type

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

void main()
{
 enum Cars {Nissan = 1 , BMW ,Mercedes ,Ferrari,Renault };
 Cars MyCar , YourCar ;
}
```

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

void main()
{
enum Cars { Nissan = 1  , BMW  , Mercedes  , Ferrari , Renault
} MyCar , YourCar;
}
```

enum

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

void main()
{
  enum Cars { Nissan = 1, BMW , Mercedes ,Ferrari, Renault };
  Cars MyCar , YourCar ;
  cout << MyCar ;
}
```

```
0
```

```
Default as 0
```

enum

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

void main()
{
// declare MyCar of type Cars and initialize it to Nissan
 enum Cars{ Nissan,BMW,Mercedes,Ferrari,Renault} MyCar = Nissan ;
}
```

```
Nothing wrong !!
```

enum

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

void main()
{
  // initialized MyCar to Nissan
 enum Cars { Nissan , BMW ,Mercedes  , Ferrari , Renault  } ;
 Cars MyCar = Nissan ;
}
```

```
Nothing wrong !!
```

enum

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

void main()
{
        enum Cars {Nissan=1,BMW,Mercedes,Ferrari,Renault} MyCar=Nissan;
        MyCar = Renault ;                       // legal
        int i1 = BMW , i2 = Mercedes ;          // legal
        i2 = Ferrari ;                          // legal
        cout << "i2 = " << i2 << endl ;
}
```

```
i2 = 4
```

enum

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

void main()
{
        enum Cars {Nissan=1,BMW,Mercedes,Ferrari,Renault}MyCar=Nissan ;
        MyCar = Renault ;
        int i1 = BMW ;
        i1 = MyCar ;
}
```

```
i1 = MyCar ; //legal
```

enum

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

void main()
{
        enum Cars {Nissan=1,BMW,Mercedes,Ferrari,Renault}MyCar=Nissan;
        MyCar = Renault ;
        int i1 = BMW ;
        MyCar = i1 ;
}
```

```
Compiler error
MyCar = i1 ;   //illegal ...
```

Remember how compiler can work and manipulate code ;)

Can't put an integer into an enum type ...guess why ?

enum

- Can assign enum type to another enum one

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

void main()
{
        enum Cars { Nissan, BMW ,Mercedes,Ferrari ,Renault};
        Cars Mine , Yours ;
        Mine = BMW ;
        Mine = Yours ;
        cout << Mine << endl ;
}
```

```
1
```

enum

- Can assign enum type to another enum one

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

void main()
{
	enum Cars {Nissan,BMW ,Mercedes ,Ferrari ,Renault };
	Cars Mine , Yours ;
	Mine = BMW ;
	Yours = Mine ;
	cout << Yours << endl ;
}
```

```
)
```

enum

- Can't assign int type to another enum ... fatte7 3yonak now :D

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

void main()
{
	enum Cars {Nissan,BMW ,Mercedes ,Ferrari ,Renault };
	Cars Mine , Yours ;
	Mine = BMW ;
	Yours = Mine + 1 ;
	cout << Yours << endl ;
}
```

Casting ... COMPILEEERRRR!

```
Compiler error
```

enum

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

void main()
{
	enum Cars {Nissan , BMW ,Mercedes ,Ferrari ,Renault };
	Cars Mine , Yours ;
	Mine = BMW ;
	Yours = Mine ++ ;
	cout << Yours << endl ;
}
```

```
Compiler error
```

enum

- Can assign int type to another enum one but with using "type cast" .. fatte7 3yonak now :D

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

void main()
{
	enum Cars {Nissan , BMW ,Mercedes ,Ferrari ,Renault };
	Cars Mine , Yours ;
	Mine = BMW ;
	Yours = Cars(Mine +1) ;
	cout << Yours << endl ;
}
```

Force the compiler to cast from int type to the programmer defined CARS type

```
)
```

enum

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

void main()
{
	enum Cars {Nissan , BMW ,Merced
	Cars Mine , Yours ;
	Mine = BMW ;
	Yours = Cars(Mine ++) ;
	cout << Yours << endl ;
}
```

Remember how compiler manipulate code ...
1st: Mine++
Check
2nd : Cars(Mine++)
Check
3rd : Assignment "="
Check

Compiler error anyway cause of "++" .. Mine++

enum

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

void main()
{
	enum Cars {Nissan , BMW ,Merced
	Cars Mine , Yours ;
	Mine = BMW ;
	Yours = Cars(Mine ++) ;
	cout << Yours << endl ;
}
```

Remember how compiler manipulate code ...
1st: Mine++
Check
2nd : Cars(Mine++)
Check
3rd : Assignment "="
Check

Compiler error anyway cause of "++" .. Mine++

enum

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

void main()
{
	enum Cars {Nissan , BMW ,Merced
	Cars Mine , Yours ;
	Mine = BMW ;
	Yours = Cars(Mine ++) ;
	cout << Yours << endl ;
}
```

Remember how compiler manipulate code ...
1st: Mine++
Check
2nd : Cars(Mine++)
Check
3rd : Assignment "="
Check

Compiler error anyway cause of "++" .. Mine++

enum

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

void main()
{
	enum Cars {Nissan , BMW ,Mercedes ,Ferrari ,Renault };
	Cars Mine , Yours ;
	Mine = BMW ;
	Yours = int(Mine) ;
	cout << Yours << endl ;
}
```

Compiler error

enum

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

void main()
{
        enum Cars {Nissan , BMW ,Merced
        Cars Mine , Yours ;
        Mine = BMW ;
        Yours = int(Mine) ;
        cout << Yours << endl ;
}
```

Remember how compiler manipulate code …
1st : int(Mine) Check
2nd : Assignment "=" Check

Compiler error

enum

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

void main()
{
        enum Cars {Nissan , BMW ,Merced
        Cars Mine , Yours ;
        Mine = BMW ;
        Yours = int(Mine) ;
        cout << Yours << endl ;
}
```

Remember how compiler manipulate code …
1st : int(Mine)
2nd : Assignment "=" Check

Compiler error

enum

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

void main()
{
        enum Cars {Nissan , BMW ,Merced
        Cars Mine , Yours ;
        Mine = BMW ;
        Yours = int(Mine) ;
        cout << Yours << endl ;
}
```

Remember how compiler manipulate code …
2nd : Assignment "=" Check

Compiler error

enum

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

void main()
{
        enum Cars {Nissan , BMW ,Merced
        Cars Mine , Yours ;
        Mine = BMW ;
        Yours = Mine + BMW ;
        cout << Yours << endl ;
}
```

Remember how compiler manipulate code …
1st: Mine + BMW Check
2nd : Assignment "=" Check

Compiler error

enum

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

void main()
{
	enum Cars {Nissan , BMW ,Merced
	Cars Mine , Yours ;
	Mine = BMW ;
	Yours = Mine + BMW ;
	cout << Yours << endl ;
}
```

Remember how compiler manipulate code ...

1st: Mine + BMW

2nd : Assignment "=" Check

Compiler error

enum

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

void main()
{
	enum Cars {Nissan , BMW ,Merced
	Cars Mine , Yours ;
	Mine = BMW ;
	Yours = Mine + BMW ;
	cout << Yours << endl ;
}
```

Remember how compiler manipulate code ...

1st: Mine + BMW

2nd : Assignment "=" Check

Compiler error

enum

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

void main()
{
	enum Cars {Nissan , BMW ,Merced
	Cars Mine , Yours ;
	Mine = BMW ;
	Yours = Mine + BMW ;
	cout << Yours << endl ;
}
```

Remember how compiler manipulate code ...

1st: Mine + BMW

2nd : Assignment "=" Check

Compiler error

enum

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

void main()
{
	enum Cars {Nissan , BMW ,Merced
	Cars Mine , Yours ;
	Mine = BMW ;
	Yours = Mine + BMW ;
	cout << Yours << endl ;
}
```

Remember how compiler manipulate code ...

1st: Mine + BMW

2nd : Assignment "=" Check

Compiler error

enum

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

void main()
{
        enum Cars {Nissan , BMW ,Merced
        Cars Mine , Yours ;
        Mine = BMW ;
        Yours = Mine + BMW ;
        cout << Yours << endl ;
}
```

Remember how compiler manipulate code ..

1st: Mine = BMW

2nd : Assignment "="

Can't Convert "Mine+BMW" which is Integer to "Yours" which is an enum !!

Constants

- A Constant :
  - Any expression that has a "fixed" value
- 3 kind of constants :
  - Integer Numbers
  - Floating-Point Numbers
  - Characters & Strings

Constants
1. Integer Numbers

- `1225` `// Decimal`
- `-982` `// Decimal`
- `05356` `// Octal !`
  - Octal numbers are preceded by **0**
- `0x3c` `// Hexadecimal`
  - Hexadecimal numbers are preceded by **0x**

Constants
2. Floating points

- Either :
  - Decimal
  - Exponent
- Examples :
  - `5.0` `// 5.0 (double)`
  - `5.0f` `// 5.0 (float)`
  - `45.556779` `// 45.556779`
  - `8.36e18` `// 8.36 x 10^18`
  - `8.36e-18` `// 8.36 x 10^-18`

Constants 3. Characters & Strings

- 'Z' //Char - Single Character
- 'M' //Char - Single Character

- "Where's the cat ?" //String - Several Character
- "I just don't know !" //String - Several Character
- "c" //String - One Character

Constant Definition

```
#include <iostream>
using namespace::std;
#define PI 3.14;                    //@ - Sign
#define MyTab '\t'                  //@ - Sign
#define PonPon ":D"                 //@ - Sign

void main()
{
}
```

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

void main()
{
    #define PI 3.14;
    #define MyTab '\t'
    #define PonPon ":D"
}
```

Constant Definition

```
#include <iostream>
using namespace::std;
#define MyTab '\t'

void main()
{
        #define PI 3.14;
        float Radius = 0  ;
        cout << "Enter the Radius" << endl ;
        cin >> Radius ;
        float Circle = PI ;
        Circle = Circle * 2 * Radius ;
        cout << "The perimeter of the Circle = "<< Circle
        << MyTab ;
}
```

Constant Definition

```
Enter the Radius
2.2
The perimeter of the Circle = 13.816    press any key to continue
```

Constant Definition

```
#include <iostream>
using namespace::std;
#define MyTab '\t'
#define PI 5;


void main()
{
	#define PI 3.14;
	float Radius = 0  ;
	cout << "Enter the Radius" << endl ;
	cin >> Radius ;
	float Circle = PI ;
	Circle = Circle * 2 * Radius ;
	cout << "The perimeter of the Circle = "<< Circle
	<< MyTab ;
}
```

Constant Definition

The same as last time !

```
Enter the Radius
3
The perimeter of the Circle = 30.000    Press any key to continue . . .
```

Constant Definition

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


void main()
{
	#define PI 3.14;
	float Radius = 0  ;
	cout << "Enter the Radius" << endl ;
	cin >> Radius ;
	float Circle = 2 * PI * Radius ;
	cout << "The perimeter of the Circle = " << Circle  ;
}
```

```
Illegal indirection  2 * PI * Radius
```

```
Will see why  later
```

Constant Definition

```
#include <iostream>
using namespace::std;
const int x = 20 ;

void main()
{
	const int y = 90 ;
}
```

```
Everything's fine :)
```

Constant Definition

```
#include <iostream>
using namespace::std;
const int x = 20 ;

void main()
{
      const y = 90 ;
}
```

```
Compile & Run
int assumed for const type when neglecting the type
```

Constant Definition

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

const char Me = 'M';
const int Height = 5 ;
const char MyCharTab = '\t';              // Char tab
const char *MyStringTab = "\t";           // String tab

void main()
{
      cout << MyStringTab ;
}
```

```
Press any key to continue . . .
```

Constant Definition

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

const char MyCharTab = '\t';              // Char tab
const char *MyStringTab = "\t";           // String tab
#define MyStringTab "\t" ;                // String tab

void main()
{
      cout << MyStringTab ;
}
```

```
Press any key to continue . . .
```

We're finished for today !

Go EAT!

Operator manipulating

- Rules of precedence :
  - 1st : ()
    - When multi nested ()
      - Inners come first
  - 2nd : * , / , %
    - Left to right
  - 3rd : + , -
    - Left to right

Operator manipulating

Operators	Precedence
!, +, - (unary operators)	first
*, /, %	second
+, -	third
<, <=, >=, >	fourth
==, !=	fifth
&&	sixth
||	seventh
= (assignment operator)	last

Operator manipulating

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

void main()
{
cout << (9*2) * 3 / 2 / 2 + (3+2) * (1-10 ) << endl ;
}
```

```
-32
```

Operator manipulating

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

void main()
{
cout << 9*2 * 3 / 2 / 2 + (3+2) * (1-10 ) << endl ;
}
```

```
-33
```

```
No pause needed for "
```

Operator manipulating

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

void main()
{
        cout << 3/2 << endl ;
}
```

```
1
```

```
Nor 1.5
```

Operator manipulating

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

void main()
{
    cout << (float)3/2 << endl ;
    system("pause");
}
```

```
Now 1.5
```

Operator manipulating

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

void main()
{
cout << (9*2) * 3 / 2 /* 2 + (3+2) * (1-10 ) << endl ;
}
```

```
illegal indirection/* ...compiler error
```

Operator manipulating

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

void main()
{
        int c = 1 ;
        c = 1 + 2 ;
        cout << c << endl ;
        c += 2 ;
        cout << c << endl ;
}
```

```

```

Operator manipulating

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

void main()
{
        int c = 1 ;
        c = 1 + 2 ;
        cout << c << endl ;
        c = 1 ;
        c += 2 ;
        cout << c << endl ;
}
```

```

```

Operator manipulating

```
+=   -=   /=   *=   %=
```

Operator manipulating

- **Special Case :**
  - **When incrementing "+" or decrementing "-" by 1**

```
b = b + 1 ;
b += 1 ;
b++ ;
```

```
are all the same
```

Operator manipulating

- Now , let's begin from scratch again ...

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

void main()
{
        int c = 1 ;
        c++ ;
        c-- ;
        cout << c << endl ;
}
```

```
1
```

Operator manipulating

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

void main()
{
        int c = 1 ;
        ++c ;
        cout << c << endl ;
}
```

```
2
```

Operator manipulating

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

void main()
{
        int c = 1 ;
        --c ;
        cout << c << endl ;
}
```

```
0
```

Operator manipulating

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

void main()
{
        int c ;
        c = 1 ;
        cout << c++ << endl ;
}
```

```
1
```

Operator manipulating

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

void main()
{
        int c ;
        c = 1 ;
        cout << ++c << endl ;
}
```

:

Operator manipulating

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

void main()
{
        int c = 1 , a  ;
        a = c++ ;
        c-- ;
        cout << c << endl ;
}
```

:

Operator manipulating

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

void main()
{
        int c = 1 , a  ;
        a = ++c ;       // a = 2 , c = 2
        c-- ;
        cout << a << endl ;
}
```

:

Operator manipulating

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

void main()
{
        int i = 0 ;
        ++i = 2;
        cout << i << endl ;
        system("pause");
}
```

:

Operator manipulating

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

void main()
{
        int i = 0 ;
        i++ = 2;
        cout << i << endl ;
        system("pause");
}
```

```
Compiler error
```

Relational Operator

Operator	Description
==	equal to
!=	not equal to
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to

Relational Operator

```
void main()
{
        int c = 1 , a = 0  ;
        if (c == 2)
        {
                c = 9 ;
        }
        cout << c << endl ;
        system("pause");
}
```

```
1
```

Relational Operator

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

void main()
{
        int c = 1 , a = 0  ;
        if (c++ == 2)
        {
                c = 9 ;
        }
        cout << c << endl ;
        system("pause");
}
```

```
2
```

Relational Operator

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

void main()
{
        int c = 1 , a = 0  ;
        if (++c == 2)
        {
                c = 9 ;
        }
        cout << c << endl ;
        system("pause");
}
```

Relational Operator

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

void main()
{
        int c = 1 , a = 0  ;
        if (c = 2)
        {
                c = 9 ;
        }
        cout << c << endl ;
        system("pause");
}
```

Relational Operator

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

void main()
{
        int c = 1 , a = 0  ;
        if (c = 2)
        {
                c = 9 ;
        }
        cout << c << endl ;
        system("pause");
}
```

Relational Operator

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

void main()
{
    int c = 1 , a = 0  ;
    a = c++ ;
    cout << a << endl << c << endl;
    system("pause");
}
```

Relational Operator

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

void main()
{
    int c = 1 , a = 0  ;
    a = ++c ;
    cout << a << endl << c << endl;
    system("pause");
}
```

```
1
1
```

Relational Operator

C++

Introduce a *L-VALUE* **not** a variable

++C

Introduce a *VARIABLE* **not** a value or L-Value

Relational Operator

C++

Introduce a *l-VALUE* **not** a variable

- **l-VALUE**
  Expressions that refer to memory locations are called "l-value" expressions. An l-value represents a storage region's "locator" value, or a "left" value, implying that it can appear on the left of the equal sign (=). L-values are often identifiers.

Relational Operator

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

void main()
{
    int c = 1 , a = 0  ;
    c++ = a ;
    cout << a << endl << c << endl;
    system("pause");
}
```

l-Value can't be assigned!

Compiler error!

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

void main()
{
    int c = 1 , a = 0  ;
    ++c = a ;
    cout << a << endl << c << endl;
    system("pause");
}
```

Variable can be assigned!

```
0
0
```

Relational Operator

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

void main()
{
        int c = 1 , a = 0  ;
        if (c++ > 2 )
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

:

Relational Operator

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

void main()
{
        int c = 1 , a = 0  ;
        if (c++ >= 2 )
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

:

Relational Operator

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

void main()
{
        int c = 1 , a = 0  ;
        if (++c > 2 )
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

:

Relational Operator

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

void main()
{
        int c = 1 , a = 0  ;
        if (++c >= 2 )
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

:

Relational Operator

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

void main()
{
        int c = 1 , a = 0  ;
        if (++c = 2 )
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

9

A variable can't be assigned

Relational Operator

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

void main()
{
        int c = 1 , a = 0  ;
        if (c++ = 2)
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

Compiler error

Relational Operator

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

void main()
{
        int c = 1 , a = 0  ;
        if (c++ = 2 )
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

Compiler error

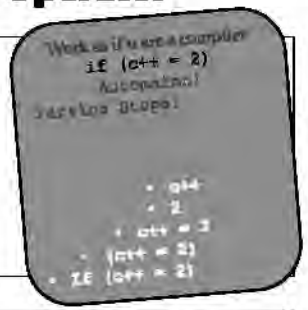
Relational Operator

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

void main()
{
        int c = 1 , a = 0  ;
        if (c++ = 2 )
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

Compiler error

Relational Operator

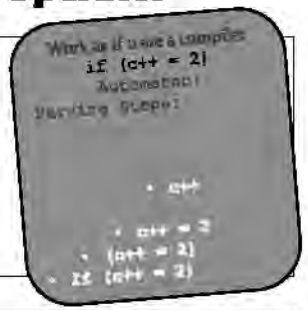
```
#include <iostream>
using namespace::std;

void main()
{
        int c = 1 , a = 0  ;
        if (c++ = 2 )
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```



Compiler error

Relational Operator

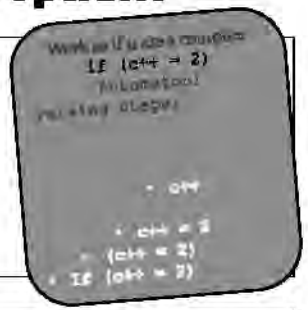
```
#include <iostream>
using namespace::std;

void main()
{
        int c = 1 , a = 0  ;
        if (c++ = 2 )
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```



Compiler error

Relational Operator

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

void main()
{
        int c = 1 , a = 0  ;
        if (c++ = 2 )
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

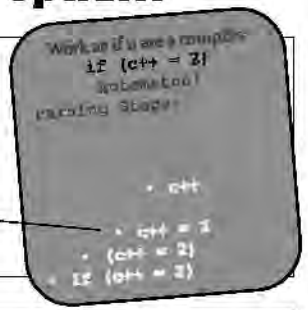


Compiler error

Relational Operator

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

void main()
{
        int c = 1 , a = 0  ;
        if (c++ = 2 )
        {
                ;
        }
                ndl ;
}
```

L-Value must not be assigned



Compiler error

Relational Operator

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

void main()
{
        int c = 1 , a = 0  ;
        if (c++ = 2 )
        {
                ;
        }
}
```

l-Value must not be assigned

Work as if u are a compiler
if (c++ = 2)
Automatic!
Parsing steps!

- c++
- c++ = 2
- (c++ = 2)
- If (c++ = 2)

Compiler error

Relational Operator

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

void main()
{
        int c = 1 , a = 0  ;
        if (c++ = 2 )
        {
                ;
        }
}
```

But why ?!

l-Value must not be assigned

Work as if u are a compiler
if (c++ = 2)
Automatic!
Parsing steps!

- c++
- c++ = 2
- (c++ = 2)
- If (c++ = 2)

Compiler error

Relational Operator

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

void main()
{
        int c = 1 , a = 0  ;
        if (c++ = 2 )
        {
                ;
        }
}
```

But why ?!

Let's roll back to here

Work as if u are a compiler
if (c++ = 2)
Automatic!
Parsing steps!

- c++
- 2
- c++ = 2
- (c++ = 2)
- If (c++ = 2)

Compiler error

Relational Operator

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

void main()
{
        int c = 1 , a = 0  ;
        if (c++ = 2 )
        {
                ;
        }
}
```

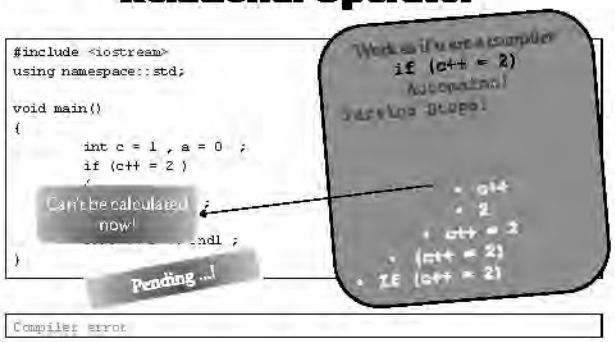
Can't be calculated now!

Work as if u are a compiler
if (c++ = 2)
Automatic!
Parsing steps!

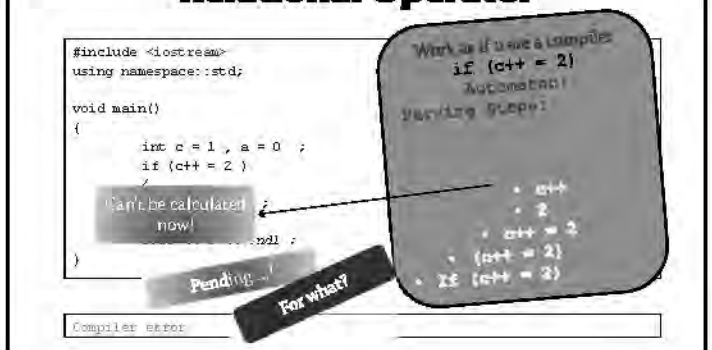
- c++
- 2
- c++ = 2
- (c++ = 2)
- If (c++ = 2)

Compiler error

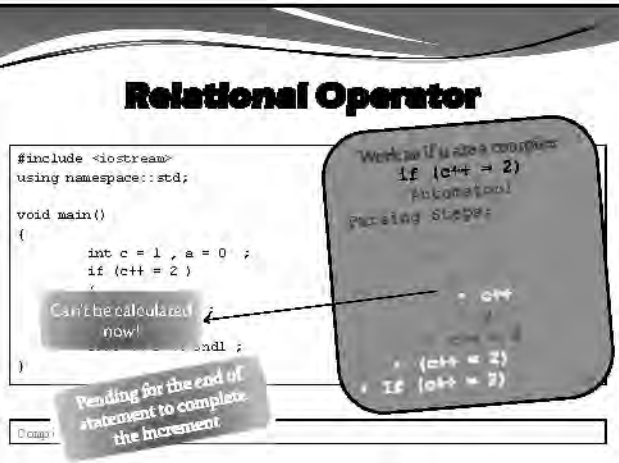
Relational Operator



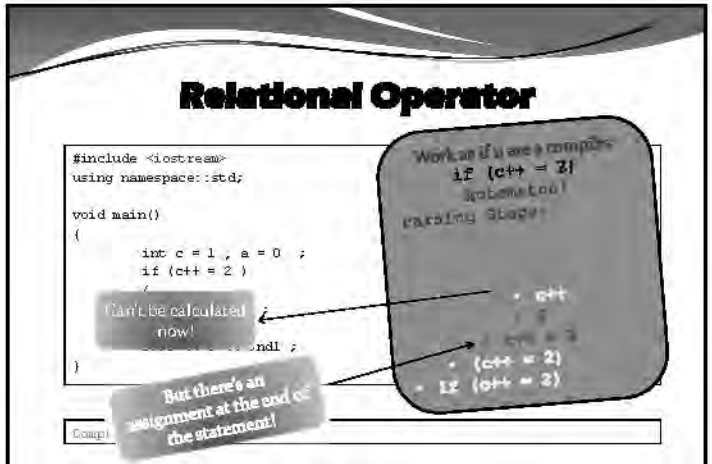
Relational Operator



Relational Operator



Relational Operator



Relational Operator

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

void main()
{
        int c = 1 , a = 0  ;
        if (c++ = 2 )
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

Can't be calculated now!

But there's an assignment at the end of the statement!

We can't assign then increment

Relational Operator

Expression	Value of Expression	Explanation
`' ' < 'a'`	true	The ASCII value of `' '` is 32, and the ASCII value of `'a'` is 97. Because 32 < 97 is `true`, it follows that `' ' < 'a'` is `true`.
`'R' > 'T'`	false	The ASCII value of `'R'` is 82, and the ASCII value of `'T'` is 84. Because 82 > 84 is `false`, it follows that `'R' > 'T'` is `false`.
`'+' < '*'`	false	The ASCII value of `'+'` is 43, and the ASCII value of `'*'` is 42. Because 43 < 42 is `false`, it follows that `'+' < '*'` is `false`.
`'6' <= '>'`	true	The ASCII value of `'6'` is 54, and the ASCII value of `'>'` is 62. Because 54 <= 62 is `true`, it follows that `'6' <= '>'` is `true`.

Relational Operator with strings

- Comparing with *collating sequence*
- If the strings have different lenghts
  - *The shorter one is smaller that the larger*

str1	= "Hello"
str2	
str3	= "Air"
str4	= "Bill"

Expression	Value
str1 < str2	true
str1 > "Han"	false
str3 < "An"	true
str1 == "hello "	false
str3 <= str4	true

Logical Operator

- !, &&, ||

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

void main()
{
        int c = 1 , a = 0  ;
        if (!(a) > 2 )
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

Logical Operator

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

void main()
{
        int c = 1 , a = 0  ;
        if ((a == 0 ) || (c == 3))
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

9

Logical Operator

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

void main()
{
        int c = 1 , a = 0  ;
        if ((a == 2 ) || (c == 3))
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

1

Logical Operator

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

void main()
{
        int c = 1 , a = 0  ;
        if ((a == 2 ) && (c == 3))
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

1

Logical Operator

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

void main()
{
        int c = 1 , a ;
        if ((a == 0 ) && (c == 1))
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

9

Logical Operator

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

void main()
{
        int c = 1 , a ;
        if ((a == 0 ) && (c == 2))
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

```
1
```

Logical Operator

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

void main()
{
        int c = 1  , a ;
        if (a == 0 ) && (c == 2)
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

```
Compiler error
```

Logical Operator

- Not always behave as expected

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

void main()
{
        int c = -1  , a   ;
        if (  0 < c <= 12 )   // Always evaluated true !
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

```
?
```

Logical Operator

- Not always behave as expec

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

void main()
{
        int c = -1  , a   ;
        if (  0 < c <= 12 )   // Al
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

```
?
```

Evaluate Compiler

if (0 < c <= 12)

Parsing Steps:

- if (0 < c <= 12)
- (0 < c <= 12)
- 0 < c

Logical Operator

- Not always behave as expec

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

void main()
{
        int c = -1  , a   ;
        if ( 0 < c <= 12 )    // Alw
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

Evaluate Compiler
if (0 < c <= 12)
Parsing Steps:
- if (0 < c <= 12)
- (0 < c <= 12)
- 0 < c
- Variable(x) <= 0 < c

Logical Operator

- Not always behave as expec

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

void main()
{
        int c = -1  , a   ;
        if ( 0 < c <= 12 )    // Alw
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

Evaluate Compiler
if (0 < c <= 12)
Parsing Steps:
- if (0 < c <= 12)
- (0 < c <= 12)
- 0 < c
- Variable(x) <= 0 < c
- Variable(x) <= 12

Logical Operator

- Not always behave as expec

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

void main()
{
        int c = -1  , a   ;
        if ( 0 < c <= 12 )    // Alw
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

Evaluate Compiler
if (0 < c <= 12)
Parsing Steps:
- if (0 < c <= 12)
- (0 < c <= 12)
- 0 < c
- Variable(x) <= 0 < c
- Variable(x) <= 12
- (Variable(x) <= 12)

Logical Operator

- Not always behave as expec

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

void main()
{
        int c = -1  , a   ;
        if ( 0 < c <= 12 )    // Alw
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

Evaluate Compiler
if (0 < c <= 12)
Parsing Steps:
- if (0 < c <= 12)
- (0 < c <= 12)
- 0 < c
- Variable(x) <= 0 < c
- Variable(x) <= 12
- (Variable(x) <= 12)
- if (Variable(x) <= 12)

Logical Operator

- Not always behave as expec

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

void main()
{
        int c = -1  , a   ;
        if ( 0 < c <= 12 )    // Alw
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

```
9
```

Evaluate Compiler

```
if (0 < c <= 12 )
```

Parsing Steps:

- if (0 < c <= 12)
- (0 < c <= 12)
- 0 < c
- Variable(x) <= 0 < c
- Variable(x) <= 12
- (Variable(x) <= 12)
- if (Variable(x) <= 12)

Now Let's parse!

Logical Operator

- Not always behave as expec

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

void main()
{
        int c = -1  , a   ;
        if ( 0 < c <= 12 )    // Alw
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

```
9
```

Evaluate Compiler

```
if (0 < c <= 12 )
```

Parsing Steps:

- if (0 < c <= 12)
- (0 < c <= 12)
- 0 < c
- Variable(x) <= 0 < c
- Variable(x) <= 12
- (Variable(x) <= 12)
- if (Variable(x) <= 12)

Logical Operator

- Not always behave as expec

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

void main()
{
        int c = -1  , a   ;
        if ( 0 < c <= 12 )    // Alw
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

```
9
```

Evaluate Compiler

```
if (0 < c <= 12 )
```

Parsing Steps:

- 0 < c
- Variable(x) <= 0 < c
- Variable(x) <= 12
- (Variable(x) <= 12)
- if (Variable(x) <= 12)

Logical Operator

- Not always behave as expec

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

void main()
{
        int c = -1  , a   ;
        if ( 0 < c <= 12 )    // Alw
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

```
9
```

Evaluate Compiler

```
if (0 < c <= 12 )
```

Parsing Steps:

- 0 < c
- Variable(x) <= 0 < c
- Variable(x) <= 12
- (Variable(x) <= 12)
- if (Variable(x) <= 12)

Logical Operator

- Not always behave as expec

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

void main()
{
        int c = -1  , a   ;
        if (  0 < c <= 12 )    // Alw
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

```
9
```

Evaluate Compiler

```
if (0 < c <= 12 )
```

Parsing Steps:

- 0 < c
  - 0 < -1
  - false!
- Variable(x) <= 0 < c
- Variable(x) <= 12
- (Variable(x) <= 12)
- if (Variable(x) <= 12)

Logical Operator

- Not always behave as expec

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

void main()
{
        int c = -1  , a   ;
        if (  0 < c <= 12 )    // Alw
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

```
9
```

Evaluate Compiler

```
if (0 < c <= 12 )
```

Parsing Steps:

- 0 < c
  - 0 < -1
  - false!
- Variable(x) <= 0 < c
- Variable(x) <= 12
- (Variable(x) <= 12)
- if (Variable(x) <= 12)

Logical Operator

- Not always behave as expec

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

void main()
{
        int c = -1  , a   ;
        if (  0 < c <= 12 )    // Alw
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

```
9
```

Evaluate Compiler

```
if (0 < c <= 12 )
```

Parsing Steps:

- 0 < c
  - 0 < -1
  - false!
- Variable(x) <= 0 < c
  - Variable(x) <= false
- Variable(x) <= 12
- (Variable(x) <= 12)
- if (Variable(x) <= 12)

Logical Operator

- Not always behave as expec

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

void main()
{
        int c = -1  , a   ;
        if (  0 < c <= 12 )    // Alw
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

```
9
```

Evaluate Compiler

```
if (0 < c <= 12 )
```

Parsing Steps:

- Variable(x) <= 12
- (Variable(x) <= 12)
- if (Variable(x) <= 12)

Logical Operator

- Not always behave as expec

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

void main()
{
        int c = -1  , a   ;
        if (  0 < c <= 12 )    // Alw
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

```
9
```

Evaluate Compiler
if (0 < c <= 12)
Parsing Steps:

- Value <= 12
- (Variable(c) <= 12)
- if (Variable(c) <= 12)

Logical Operator

- Not always behave as expec

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

void main()
{
        int c = -1  , a   ;
        if (  0 < c <= 12 )    // Alw
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

```
1
```

Evaluate Compiler
if (0 < c <= 12)
Parsing Steps:

- Value <= 12
  - 0 <= 12
- (Variable(c) <= 12)
- if (Variable(c) <= 12)

Logical Operator

- Not always behave as expec

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

void main()
{
        int c = -1  , a   ;
        if (  0 < c <= 12 )    // Alw
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

```
9
```

Evaluate Compiler
if (0 < c <= 12)
Parsing Steps:

- Value <= 12
  - 0 <= 12
  - true
- (Variable(c) <= 12)
- if (Variable(c) <= 12)

Logical Operator

- Not always behave as expec

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

void main()
{
        int c = -1  , a   ;
        if (  0 < c <= 12 )    // Alw
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

```
9
```

Evaluate Compiler
if (0 < c <= 12)
Parsing Steps:

- (Variable(c) <= 12)
- if (Variable(c) <= 12)

Logical Operator

- Not always behave as expec

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

void main()
{
        int c = -1  , a  ;
        if (  0 < c <= 12 )   // Alw
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

```
9
```

Evaluate Compiler
if (0 < c <= 12)
Parsing Steps:

- if (Variable(c) <= 12)

Logical Operator

- Not always behave as expec

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

void main()
{
        int c = -1  , a  ;
        if (  0 < c <= 12 )   // Alw
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

Evaluate Compiler
if (0 < c <= 12)
Parsing Steps:

- if (Variable(c) <= 12)

Logical Operator

- Not always behave as expec

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

void main()
{
        int c = -1  , a  ;
        if (  0 < c <= 12 )   // Alw
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

```
9
```

Evaluate Compiler
if (0 < c <= 12)
Parsing Steps:

- if

Logical Operator

- Not always behave as expect

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

void main()
{
        int c = -1  , a  ;
        if (  0 < c <= 12 )   // Alw
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

```
9
```

Evaluate Compiler
if (0 < c <= 12)
Parsing Steps:

Logical Operator

- Not always behave as expected

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

void main()
{
        int c = 0  , a   ;
        if ((  0 < c ) && ( c <= 12 ))
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

```
0
```

Conditional Operator ?

- Code :

```
Condition ? Result1 : Result2
```

- Condition
  - If True
    - Result1
  - Else // condition is false
    - Result 2

Logical Operator

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

void main()
{
        int c ;
        c == 0 ? c = 2 :  c = 1 ;
        cout << c << endl ;

}
```

```
2
```

Logical Operator

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

void main()
{
        int c ;
        c == 2 ? c = 2 :  c = 1 ;
        cout << c << endl ;

}
```

```
2
```

Logical Operator

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

void main()
{
	int c ;
	c != 0 ? c = 2 :  c = 1 ;
	cout << c << endl ;

}
```

Logical Operator

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

void main()
{
	int c ;
	c == 0 ? 2 : 1 ;
	cout << c << endl ;

}
```

Logical Operator

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

void main()
{
	int c ;
	c != 0 ? 2 : 1 ;
	cout << c << endl ;

}
```

Logical Operator

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

void main()
{
	int c ;
	c != 0 ? c=2 :  1 ;
	cout << c << endl ;

}
```

Logical Operator

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

void main()
{
        int c ;
        c = 0 ? c=2 :   c=1 ;
        cout << c << endl ;

}
```

1

1 1

Logical Operator

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

void main()
{
        int c ;
        c = 0 ? c=2 :   1 ;
        cout << c << endl ;

}
```

1

1 1 1

When using "=" the variable takes the values without "=" in the result side ! And takes the last Result

Code Cracking

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

void main()
{
        int c=1 , b=2 , d=3;
        cout << c+(b+c++) << endl ;
}
```

4

Code Cracking

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

void main()
{
        int c=1 , b=2 , d=3;
        cout << c+(b+++c) << endl ;
}
```

Compiler error ^ Output: ?

No compiler error … it means : cout << c+(b + ++c)<<endl;

4

The Same as last time ! c+(b+c++) !!!

Code Cracking

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

void main()
{
        int c=1 , b=2 , d=3;
        cout << c+(++c+b) << endl ;
}
```

6

!"!!! Why [illegible]

[illegible] Who knows [illegible] 100 pts [illegible]

Code Cracking

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

void main()
{
        int c=1 , b=2 , d=3;
        cout << c+(1 + ++c) << endl ;
}
```

5

!!! Not like the slides before when using constant value "1" and not another variable "b" like before

Code Cracking

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

void main()
{
        int c=1 , b=2 , d=3;
        cout << c+(1 + ++c) << endl ;
}
```

Reference Point!

5

!!! Not like the slides before when using constant value "1" and not another variable "b" like before

Code Cracking

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

void main()
{
        int c=1 , b=2 , d=3;
        cout << c+(1+++c) << endl ;
}
```

Compiler error !

!!! Also , Not like the slides before when using constant value "1" and not another variable "b" like before

Code Cracking

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

void main()
{
        int c=1 , b=2 , d=3;
        cout << c+(1+++c) << endl ;
}
```

Compiler error !

Reference Point!

!!! Also , Not like the slides before when using (increment value "1" and put another variable "b" like before

Code Cracking

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

void main()
{
        int c=1 , b=2 , d=3;
        cout << (++c+b)++ << endl ;
}
```

Compiler error !

Code Cracking

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

void main()
{
        enum Cars {Nissan,BMW,Mercedes, Ferrari ,Renault=Nissan-- };
        Cars MyCar = Renault ;
}
```

Compiler error !! Nissan--

Code Cracking

- Another example :

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

void main()
{
        int i1 = 25  ;
        long i2  ;
        float i3 ;
        i2 = static_cast<float>(i3) ;
}
```

Warning ?

Still having a warning

Always note the type we are converting to ;)

Code Cracking

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

void main()
{
        enum Cars(Nissan,BMW,Mercedes ,Ferrari,Renault++)MyCar=Nissan;

}
```

Compiler error .. Renault++

Code Cracking

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

void main()
{
        enum Cars { Nissan = 2.4 ,BMW ,Mercedes ,Ferrari ,Renault
}MyCar = Nissan ;
        MyCar = Renault ;
        int i1 = MyCar ;
}
```

Compiler error ,, Nissan = 2.4 must be integer

Code Cracking

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

void main()
{
        int c = 1 , a ;
        if ((a = 0 ) && (c = 2))
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

0

Watch out for the '=' operator

Code Cracking

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

void main()
{
        int c = 1 , a = 0  ;
        if ((a = 3) > 2 )
        {
                c = 9 ;
        }
        cout << c << endl ;
}
```

9

Code Cracking

```
#include <iostream>
using namespace::std;
const x = 20 ;

void main()
{

}
```

2005 Compiler : Compiler error no default int supported
2008 Compiler : Compile ... Prog... int assumed

Code Cracking

```
#include <iostream>
using namespace::std;
#define MyStringTab =  "\t" ;
#define x = 20 ;

void main()
{

}
```

2005 / 2008 No Compiler error ....

Code Cracking

```
#include <iostream>
using namespace::std;
#define MyStringTab =  "\t" ;
#define x = 20 ;

void main()
{
        cout << x << MyStringTab ;
}
```

2008 Now , it's a compiler error

Code Cracking

```
#include <iostream>
using namespace::std;
void main()
{
        int c ;
        if (c++ == 1)
        {
                cout << "This is wrong !"<<endl ;
        }
        else
        {
                cout << "This is !!! " << endl ;
        }
}
```

This is !!!

Code Cracking

```
#include <iostream>
using namespace::std;
void main()
{
        int c ;
        if (++c == 1)
        {
                cout << "This is wrong !"<<endl ;
        }
        else
        {
                cout << "This is !!! " << endl ;
        }
}
```

```
This is wrong !
```

Code Cracking

```
#include <iostream>
using namespace::std;
void main()
{
        int c ;
        if (++c = 1)
        {
                cout << "This is wrong !"<<endl ;
        }
        else
        {
                cout << "This is !!! " << endl ;
        }
}
```

```
Compiler error
```

Code Cracking

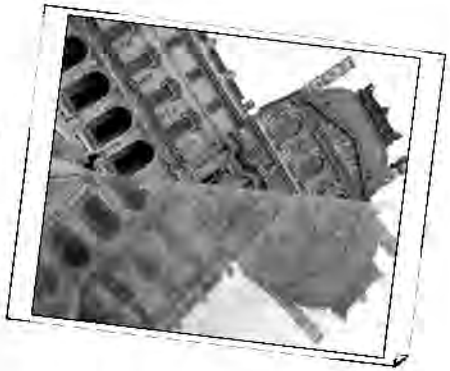
```
#include <iostream>
using namespace::std;

void main()
{
        double b1  , b2 ;
        bool f ;
        if (b1 = !(!!(f)))
        {
                b1 = !(1-1231233) ;
                b2 = b2 + b1 ;
        }
        cout << b1 << b2 << endl ;
}
```

```
Compiler error ? Compile & run ? Output ?
```

```
00
```

Welcome!

How are u ?!
Let's go!



Functions I

Functions

- Why function ?
- functions \ procedures with c++
- What's the function prototype (declaration) \ definition ?
- Function signature

```
int ZeZe(int) ;                    // function prototype
```

- It's *just* the name of the function with its parameters
  - No return type !!!! (*Return type is not considered as part of the function's signature! , guess why? ;)*)

```
ZeZe(int)                          // function signature
```

Functions

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

int ZeZe(int) ;                          // function prototype


int  main()
{

}
```

Functions

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

int ZeZe(int)                    // function prototype


int  main()
{
       return 0 ;
}
```

Compiler error : missing ; after function's declaration

Functions

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

ZeZe(int);                       // function prototype


int  main()
{
       return 0 ;
}
```

Compiler error : missing return type

Functions

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

int ZeZe(int);                          // function prototype


int  main()
{

}
```

Functions

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

int Mult( int x )                       // Hope there's no ; when we
                                        // write the definition here
{
        x = x * 2 ;
        return x ;
}

void main()
{
}
```

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

int Mult( int x );

void main()
{}

int Mult( int x )
{
        x = x * 2 ;
        return x ;
}
```

The same

Functions

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

int Mult( int )
{
        x = x * 2 ;
        return x ;
}

void main()
{
}
```

```
Compiler error , missing ;
```

Functions

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

int Mult( int x )
{
        x = x * 2 ;
        return x ;
}

void main()
{
}
```

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

int Mult(int);              // We didn't write the ;
                            // primshed when prototype only
void main()
{}

int Mult( int x )
{
        x = x * 2 ;
        return x ;
}
```

The same

Functions

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

int Mult(int);

void main()
{
        Mult (3) ;
}

int Mult( int x )
{
        x = x * 2 ;
        return x ;
}
```

Functions

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

int Mult(int);

void main()
{
        cout << Mult(3) ;
}

int Mult( int x )
{
        x = x * 2 ;
        return x ;
}
```

```
6
```

Functions

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

int Mult(int);

void main()
{
        Mult(3) ;
}

int Mult( int x )
{
        x = x * 2 ;
        cout << x ;
        return 0 ;
}
```

```
6
```

Functions

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

int Mult(int);

void main()
{
        int c = 3 ;
        Mult(c) ;
}

int Mult( int x )
{
        x = x * 2 ;
        cout << x ;
        return 0 ;
}
```

```
6
```

Functions

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

int Mult(int);

void main()
{
        int x = 3 ;
        Mult(x) ;
}

int Mult( int x )
{
        x = x * 2 ;
        cout << x ;
        return 0 ;
}
```

```
6
```

Functions

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

int Mult(int);

void main()
{
        int x = 3 ;
        Mult(x) ;
        cout << x ;
}

int Mult( int x )
{
        x = x * 2 ;
        cout << x ;
        return 0 ;
}
```

```
6
3
```

Functions

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

int Mult(int);

void main()
{
        int x = 3 ;
        Mult(x) ;
        cout << endl ;
        cout << x ;
}

int Mult( int x )
{
        x = x * 2 ;
        cout << x ;
        return 0 ;
}
```

```
6
3
```

Functions

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

int Mult(int);

void main()
{
        int x = 3 ;
        Mult(x) ;
        cout << x ;
}

int Mult( int x )
{
        x = x * 2 ;
        cout << x << endl ;
        return 0 ;
}
```

```
6
3
```

Functions

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

int Mult(int);

void main()
{
        int x = 3 ;
        cout << Mult(x) << endl ;
        cout << x ;
}

int Mult( int x )
{
        x = x * 2 ;
        return x ;
}
```

```
6
3
```

Functions

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

void Mult(void);

void main()
{
        int x = 3;
        Mult() << endl ;
        cout << x ;
}

void Mult( void )
{
        int x = 3 ;
        cout << x * 2 << endl ;
}
```

```
Compiler error
```

Functions

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

void Mult(void);

void main()
{
        int c = 3;
        Mult() << endl ;
        cout << c ;
}

void Mult( void )
{
        int x = 3 ;
        cout << x * 2 << endl ;
}
```

Compiler error

Functions

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

void Mult(void);

void main()
{
        cout << x ;
        Mult() << endl ;
}

void Mult( void )
{
        int x = 3 ;
        cout << x*2 << endl ;
}
```

Compiler error : x undeclared

Functions

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

void Mult(void);

void main()
{
        Mult() << endl ;
}

void Mult( void )
{
        int x = 3 ;
        x = x * 2 ;
        return 0 ;
}
```

Compiler error [illegible] return [illegible] void function

Functions

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

void Mult(void);

void main()
{
        Mult() << endl ;
}

void Mult( void )
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error : endl [illegible] Mult()

Functions

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

void Mult(void);

void main()
{
        cout << Mult() << endl ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Functions

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

void Mult(void);

void main(void)
{
        Mult() ;
}

void Mult(void)
{
        int x = 3 ;
        x = x * 2 ;
}
```

Functions

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

void Mult(void);

void main()
{
        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

In the parameter list... void or blank are the same

Functions

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

Mult(void);

void main()
{
        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Functions

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

void Mult(void);

void main()
{
        Mult() ;
}

Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error : Missing return type

Functions

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

Mult(void);

void main()
{
        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error : Missing return type

Functions

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

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}

void main()
{
        Mult() ;
}
```

Functions

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

void main()
{
        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error : Missing function prototype

Functions

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

void main()
{
        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler Parsing

```
#include <iostream>
using namespace::std;
void main()
{
        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error … missing function's prototype

Functions

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

void main()
{
        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler Parsing

```
#include <iostream>
using namespace::std;
{
        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error … missing function's prototype

Functions

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

void main()
{
        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler Parsing

```
#include <iostream>
using namespace::std;
{
        // Enter new scope
        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error … missing function's prototype

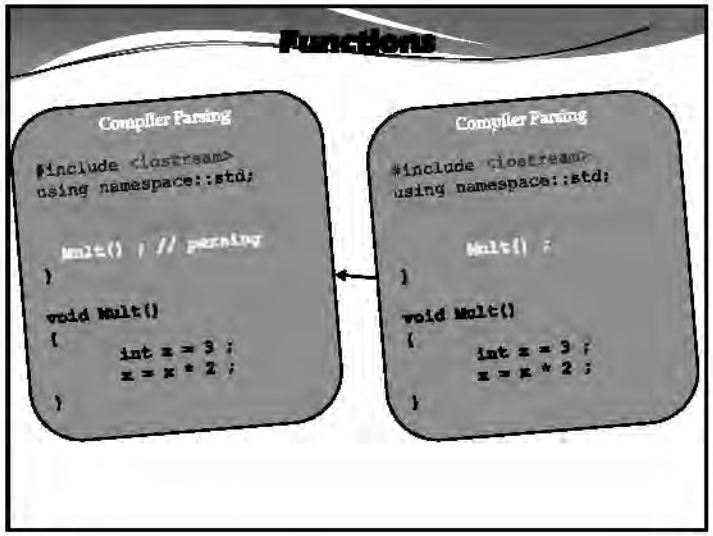
Functions

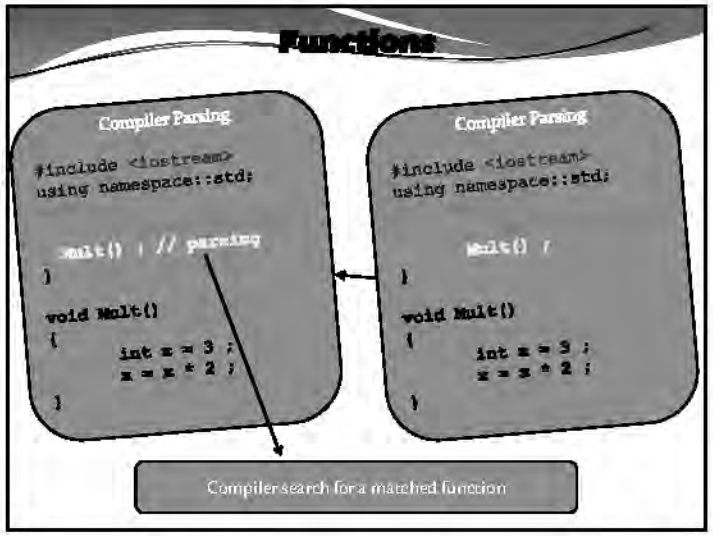
```
#include <iostream>
using namespace::std;

void main()
{
        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

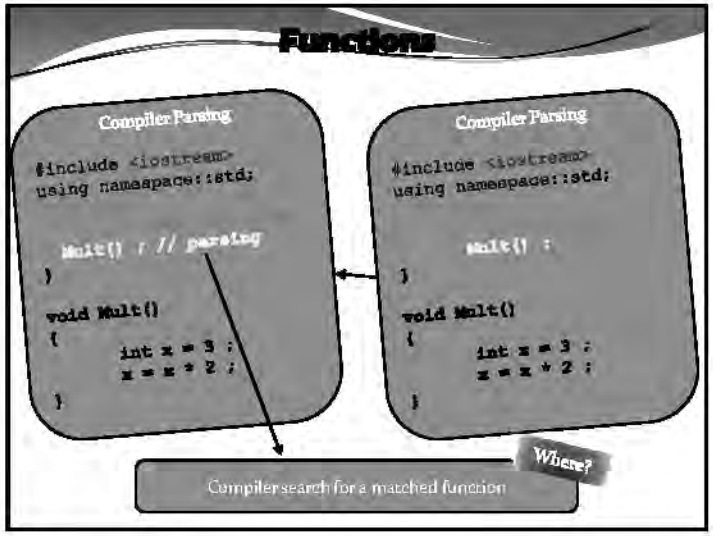
Compiler Parsing

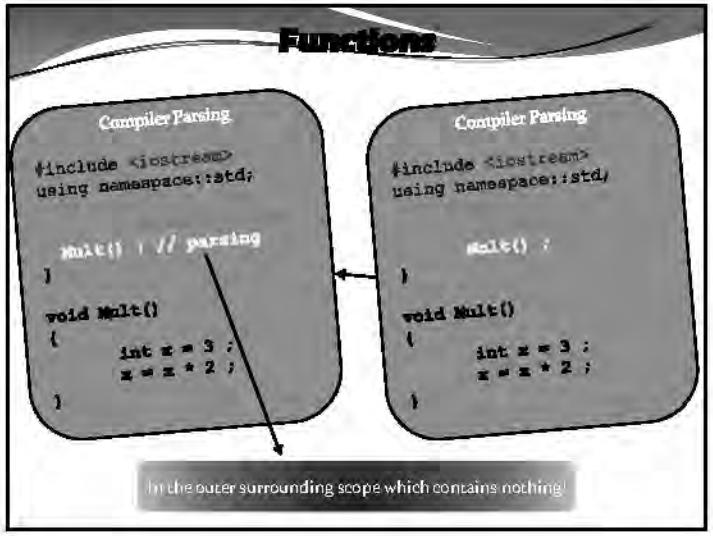
```
#include <iostream>
using namespace::std;

        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error … missing function's prototype


Functions
Compiler Parsing
#include <iostream>
using namespace::std;
Mult() ; // parsing
}
void Mult()
{
int x = 3 ;
x = x * 2 ;
}
Compiler Parsing
#include <iostream>
using namespace::std;
Mult() ;
}
void Mult()
{
int x = 3 ;
x = x * 2 ;
}


Functions
Compiler Parsing
#include <iostream>
using namespace::std;
Mult() ; // parsing
}
void Mult()
{
int x = 3 ;
x = x * 2 ;
}
Compiler Parsing
#include <iostream>
using namespace::std;
Mult() ;
}
void Mult()
{
int x = 3 ;
x = x * 2 ;
}
Compiler search for a matched function


Functions
Compiler Parsing
#include <iostream>
using namespace::std;
Mult() ; // parsing
}
void Mult()
{
int x = 3 ;
x = x * 2 ;
}
Compiler Parsing
#include <iostream>
using namespace::std;
Mult() ;
}
void Mult()
{
int x = 3 ;
x = x * 2 ;
}
Where?
Compiler search for a matched function


Functions
Compiler Parsing
#include <iostream>
using namespace::std;
Mult() ; // parsing
}
void Mult()
{
int x = 3 ;
x = x * 2 ;
}
Compiler Parsing
#include <iostream>
using namespace::std;
Mult() ;
}
void Mult()
{
int x = 3 ;
x = x * 2 ;
}
In the outer surrounding scope which contains nothing!

Functions

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

void Mult(void);

void main(void)
{
        Mult() ;
}

int Mult(void)
{
        int x = 3 ;
        x = x * 2 ;
        return x ;
}
```

Compiler error : conversion type differs

Functions

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

int Mult(void);

void main(void)
{
        Mult() ;
}

void Mult(void)
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error : conversion type differs

Functions

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

int Mult(void);

void main(void)
{
        cout << Mult() << endl ;
}

int Mult(void)
{
        int x = 3 ;
        x = x * 2 ;
        return x ;
}
```

6

Functions

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

int Mult(void);

void main(void)
{
        cout << Mult() << endl ;
}

int Mult(void)
{
        int x = 3 ;
        return x * 2 ;
}
```

6

Functions

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

int Mult(void);
void main(void)
{
        cout << Mult() << endl ;
}

int Mult(void)
{
        int x = 3 ;
        return x * 2 ;
        cout << x << end ;
}
```

6

Warning will be issued after the return statement

Functions

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

int Mult(int x );

void main(void)
{
        cout << Mult() << endl ;
}

int Mult(int x )
{
        return x * 2 ;
}
```

Compiler error : no variable passed to function

Functions

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

int Mult(int x );

void main(void)
{
        cout << Mult(3) << endl ;
}

int Mult(int x )
{
        return x * 2 ;
}
```

6

Functions

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

int Mult(int x );

void main(void)
{
        cout << Mult(3) << endl ;
}

int Mult(int x )
{
        int x = 2 ;
        return x * 2 ;
}
```

Compiler error : redefinition of parameter x

Functions

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

int Mult(int , int );
void main(void)
{
        int i1 , i2 ;
        cout << Mult(i1,i2) << endl ;
}

int Mult(int x , int y)
{
        return x * 2 ;
}
```

0

Functions

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

int Mult(int , int );
void main(void)
{
        int i1 , i2 ;
        cout << Mult(i1,i2) << endl ;
}

int Mult(int x , y)
{
        return x * 2 ;
}
```

Compiler error : missing type for y

Functions

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

int Mult(int , int );
void main(void)
{
        int i1 , i2 ;
        cout << Mult(i1,i2) << endl ;
}

int Mult(int x , int y)
{
        if (x = 2 )
        {
                return 3 ;
        }
        else
        {
                return x * 2 ;
        }
}
```

3

Functions

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

int Mult(int , int );
void main(void)
{
        int i1 , i2 ;
        cout << Mult(i1,i2) << endl ;
}

int Mult(int x , int y)
{
        int MaMe (int z)
        {
                return 0;
        }
}
```

Compiles error : can't define a function inside another one

Functions

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

int Mult(int , int );
void main(void)
{
        float i1= 3.4 , i2 ;
        cout << Mult(i1,i2) << endl ;
}

int Mult(int x , int y)
{
        return 2*x ;
}
```

Compile error

6.8

6

Downcasting

Functions

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

int Mult(int);

void main(void)
{
        for (int i = 0 ; i<=10 ; i++)
        {
                cout << Mult(i) << "-";
        }
        cout << "haha" << endl ;
        cout<< endl ;

}

int Mult(int x )
{
        return 2*x ;
}
```

0-2-4-6-8-10-12-14-16-18-20-haha

Press any key to continue . . .

Functions

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

void Crazy1();
void Crazy2(int);

void main(void)
{       Crazy1();         }

void Crazy1()
{
        int x = 3 ;
        Crazy2(x) ;
        cout << x ;
}

void Crazy2(int x )
{
        x+=6 ;
        cout << x << endl ;
}
```

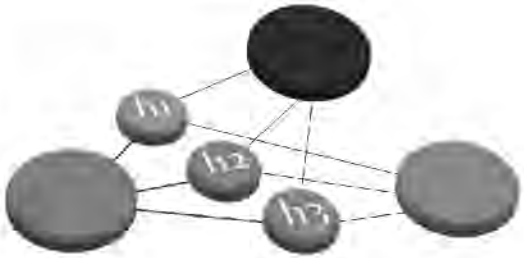
9
3

100 pts

Headers

Headers

- The Concept of headers



Headers File

- What's a header ?
  - Moving functions out side the main program
    - Logical groups
      - Stored in *their own* files
- How to do it ?
  - By Moving ...
    - *function declarations*
      - Into headers files
        - (.h) files // header files
    - *function definitions*
      - Into source files
        - (.cpp) files // source files
      - **Note** : function definitions ***can't*** be split up into multiple files !

Headers – Math Library

- To use it
  - Include <cmath>

```
#include <iostream>
#include <cmath>
using namespace::std;


void main(void)
{
        cout << sqrt(100.0) << endl;
}
```

```
10
```

Headers – Math Library

```
#include <iostream>

using namespace::std;


void main(void)
{
        cout << sqrt(100.0) << endl;
}
```

```
[Compiler error ... identifier sqrt found]
```

Headers – Math Library

```
#include <iostream>
#include <cmath>
using namespace::std;


void main(void)
{
        cout << sqrt(100) << endl;
}
```

```
Compiler error: "can't use integer "to overloaded function"
```

Headers – Math Library

```
#include <iostream>
#include <cmath>
using namespace::std;


void main(void)
{
        cout << sqrt(sqrt(100.0)) << endl;
}
```

```
3.16113
```

Headers – Math Library

```
#include <iostream>
#include <cmath>
using namespace::std;


void main(void)
{
        int x ;
        cout << sqrt(6*x) << endl;
}
```

```
Compiler error: "can't use integer "no overloaded function"
```

Headers – Math Library

```
#include <iostream>
#include <cmath>
using namespace::std;


void main(void)
{
        double x = 3  ;
        cout << sqrt(3*x) << endl;
}
```

```
3
```

Headers – Math Library

```
#include <iostream>
#include <cmath>
using namespace::std;

void main(void)
{
        float x = 3  ;
        cout << sqrt(3*x) << endl;
}
```

```
:
```

rand()

- Random number
  - `rand()`
  - Needs to include `<cstdlib>`
    - Generates *unsigned integers*
      - Between
        - 0 (Zero :D)
        - `Rand_Max` number (usually 32767)

rand()

```
#include <iostream>
#include <cstdlib>
using namespace::std;

void main(void)
{
        int x = 3  ;
        x = rand();
        cout << x << endl ;
}
```

```
$ (
```

```
Random Number: __
```

Now , when compiling many times ...
Every time we have the same value !

```
$ (
```

```
$ (
```

```
$_
```

rand()

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

void main(void)
{
        int x = 3  ;
        x = rand();
        cout << x << endl ;
}
```

```
$ (
```

```
Needs windows <cstdlib>
```

rand()

```
#include <iostream>
#include <cstdlib>
using namespace::std;


void main(void)
{
        int x = 3  ;
        x = rand();
        cout << x << endl ;
        cout << x << endl ;
}
```

```
16807
282475249
```

Random Number

rand()

```
#include <iostream>
#include <cstdlib>
using namespace::std;


void main(void)
{
        int x = 3  ;
        x = rand();
        cout << x << endl ;
        x = rand();
        cout << x << endl ;
}
```

```
16807
1622650073
```

rand()

- `srand()`
  - Include `<cstdlib>`
  - Give a starting value for the `rand()` function
  - Usually used once in program

rand()

```
#include <iostream>
#include <ctime>
using namespace::std;


void main(void)
{
        int x = 3  ;
        srand (time(0));
        cout << x << endl ;
}
```

```
3
```

rand()

```
#include <iostream>
#include <ctime>
using namespace::std;

void main(void)
{
        int x = 3  ;
        srand (time());
        cout << x << endl ;
}
```

rand()

```
#include <iostream>
#include <ctime>
using namespace::std;

void main(void)
{
        int x = 3  ;
        srand (time(0));
        x = rand () ;
        cout << x << endl ;
}
```

What happened when compiling many times ?

Now , when compiling many times ...
Every time we have a new different value :)

rand()

```
#include <iostream>
#include <ctime>
using namespace::std;

void main(void)
{
        int x = 3  ;
        srand (5);
        x = rand () ;
        cout << x << endl ;
}
```

What happened when compiling many times ?

Not an appropriate srand() parameter leads to insuffiencient using of srand() !!!

rand()

- Scaling & shifting
  - %
    - `x % y // returns a value between 0 to y-1`
    - let's see an example ...

rand()

```
#include <iostream>
#include <cstdlib>
using namespace::std;


void main(void)
{
        int i ;
        i = rand() % 6 + 1 ;
        // rand() % 6 : number between 0 to 5
        // +1 : makes a shift of the range we have
        // 0 to 5 becomes 1 to 6
        cout << i << endl ;
        // here will print a number absolutely between 1 to 6
}
```

```
6
```

```
Or any other number between 1 and 6
```

rand()

```
#include <iostream>
#include <ctime>
using namespace::std;
void main(void)
{
        srand(clock());
        // clock is from <ctime>
        // no needs to value for clock ()
        for (int i = 0 ; i < 10 ; i++)
                cout << rand() << endl ;
}
```

```
744
9416
16377
7407
14053
5490
30850
16183
[illegible]
2140
Press any key to continue . . .
```

Storage Classes

- *Static Storage : (__Very important__)*
  - `static` key word
    - function's local variables
    - *Keeps values between functions call*
      - ***Watch it … Watch it … Watch it … Watch it … Watch it …***
    - Known ***ONLY*** on ***ITS OWN*** function

Storage Classes

- *Static Storage : (__Very important__)*

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

void StaticMyHeadache()
{
        static int x = 8 ;
        cout << "In function , x = " << x << endl ;
        x*=2 ;
        cout << "In function , x = " << x << endl ;
}

int main()
{
        int x = 3 ;
        cout << x << endl ;
        StaticMyHeadache();
        cout << x << endl ;
        StaticMyHeadache();
}
```

Defined and initialized at the same time in the ***first time of calling the function*** then the compiler no longer parse this line of code

Storage Classes

- *Static Storage : (**Very important**)*

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

void StaticMyHeadache()
{
        static int x = 8 ;
        cout << "In function , x = " <<  x << endl ;
        x*=2 ;
        cout << "In function , x = " <<  x << endl ;
}

int main()
{
        int x = 3 ;
        cout << x << endl ;
        StaticMyHeadache();
        cout << x << endl ;
        StaticMyHeadache();
}
```

What's the output ?

Storage Classes

- *Static Storage : (**Very important**)*

```
3
In function , x = 8
In function , x = 16
3
In function , x = 16
In function , x = 32
Press any key to continue
```

Storage Classes

- *Static Storage : (**Very important**)*
  - `extern` key word
    - Global variables accessible to functions in
      - *Same file*
      - *Other files*
    - Must be declared in *each file* which has been used in .
    - Used to access Global variable in another file

Storage Classes - extern

1st file

```
#include <iostream>
using namespace::std;
void main(void)
{
        int MyGlobalVar;
}
```

2nd file

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

extern int MyGlobalVar;
```

Storage Classes - extern

1st file

```
#include <iostream>
using namespace::std;
void main(void)
{
        int MyGlobalVar;
}
```

2nd file

```
#include <iostream>
using namespace::std;
void main(void)
{
        extern int MyGlobalVar;
}
```

```
Compiler error ... why
```

```
2 main's in the same project ... that's wrong
```

Playing with scopes !

- Local Scopes
- Global Scopes

- **<u>The Golden Rule</u>**
  - *Life time of block scope variables is lifetime of block*

Playing with scopes !

```
#include <iostream>

using namespace::std;


void main(void)
{
        int x = 9 , y = 3 ;
        {
                cout << "This is inner block " << endl;
        }
}
```

```
Compiler error
```

```
This is inner block
```

Playing with scopes !

```
#include <iostream>

using namespace::std;


void main(void)
{
        int x = 9 , y = 3 ;
        {
                cout << x << endl ;
        }
}
```

```
9
```

Playing with scopes I

```
#include <iostream>

using namespace::std;


void main(void)
{
        int x = 9 , y = 3 ;
        {
                cout << x << endl ;
        }
        cout << x << endl;
}
```

```
9
3
```

Playing with scopes I

```
#include <iostream>

using namespace::std;


void main(void)
{
        int x = 9 , y = 3 ;
        {
                int x = 4  ;
                cout << x << endl ;
        }
}
```

```
4
```

Playing with scopes I

```
#include <iostream>

using namespace::std;


void main(void)
{
        int x = 9 , y = 3 ;
        {
                int x ;
                x = 4 ;
                cout << x << endl ;
        }
         cout << x << endl ;
}
```

```
4
9
```

Playing with scopes I

```
#include <iostream>

using namespace::std;


void main(void)
{
        int x = 9 , y = 3 ;
        {
                int x = 4  ;
                cout << x << endl ;
        }
         cout << y << endl ;
}
```

```
4
3
```

Playing with scopes !

```
#include <iostream>

using namespace::std;


void main(void)
{
        int x = 9 ;
        {
                int x = 4 , y = 3  ;
                cout << x << endl ;
        }
         cout << y << endl ;
}
```

```
Compiler error , y undeclared identifier
```

Playing with scopes !

```
#include <iostream>

using namespace::std;
int x = 35 ;                                   // Global Variable

void SoSo ()
{
        int x = 3 ;                            // local Variable
}

void main(void)
{
   int x = 9 , y = 3 ;
        {
                int x = 6 ;                    // local Variable
        }
    cout << x << endl ;
}
```

```
9
```

Playing with scopes !

```
#include <iostream>

using namespace::std;
int x = 35 ;                                   // Global Variable

void SoSo ()
{
        int x = 3 ;                            // local Variable
}

void main(void)
{
        int x = 9 , y = 3 ;
        {
                int x = 6 ;                    // local Variable
        }
        cout << ::x << endl ;
}
```

```
35
```

Playing with scopes !

```
#include <iostream>

using namespace::std;
int x = 34 ;                                   // Global Variable

void SoSo ()
{
        int x = 3 ;                            // local Variable
        cout << ::x << endl ;
}

void main(void)
{
        int x = 9 , y = 3 ;
        {
                int x = 6 ;                    // local Variable
        }
}
```

```
Compile & Run but nothing in output
Nothing to output ... !
```

Playing with scopes I

```
#include <iostream>

using namespace::std;
int x = 34 ;                                    // Global Variable

void SoSo ()
{
        int x = 3 ;                             // local Variable
        cout << x << endl ;
}

void main(void)
{
        int x = 9 , y = 3 ;
        {
                int x = 6 ;                     // local Variable
        }
        SoSo();
}
```

3

Playing with scopes I

```
#include <iostream>

using namespace::std;
int x = 34 ;                                    // Global Variable

void SoSo ()
{
        int x = 3 ;                             // local Variable
        cout << ++::x << endl ;
}

void main(void)
{
        int x = 9 , y = 3 ;
        {
                int x = 6 ;                     // local Variable
        }
        SoSo();
}
```

35

Playing with scopes I

```
#include <iostream>

using namespace::std;
int x = 34 ;                                    // Global Variable

void SoSo ()
{
        int x = 3 ;                             // local Variable
        cout << ::x++ << endl ;
}

void main(void)
{
        int x = 9 , y = 3 ;
        {
                int x = 6 ;                     // local Variable
        }
        SoSo();
        cout << ::x << endl ;
}
```

34
35

Inline functions

- `Inline` functions
  - `Inline` key word
    - For small , common-used functions
    - No function call , just copy the code into the program
    - Compile can ***ignore*** `Inline`

Inline functions

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

int IWannaSleep(int);

void main( )
{
        int x;
        cout << "Enter the Input Value: ";
        cin>>x;
        cout<<"\n The Output is: " << IWannaSleep(x) << endl ;
}

inline int IWannaSleep (int x1)
{
        return 5*x1;
}
```

```
Enter the Input Value: 2

 The Output is: 10
Press any key to continue . . .
```

We're finished for NOW !!!

Math library functions

Method	Description	Example
ceil(x)	rounds x to the smallest integer not less than x	ceil(5.2) is 6.0 ceil(-5.2) is -5.0
cos(x)	trigonometric cosine of x (x in radians)	cos(0.0) is 1.0
exp(x)	exponential function ex	exp(1.0) is 2.71828 exp(2.0) is 7.38906
fabs(x)	absolute value of x	fabs(5.1) is 5.1 fabs(0.0) is 0.0 fabs(-5.76) is 5.76
floor(x)	rounds x to the largest integer not greater than x	floor(9.2) is 9.0 floor(-9.8) is -10.0
fmod(x, y)	remainder of x/y as a floating-point number	fmod(13.657, 2.333) is 1.992
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(100.0) is 2.0
pow(x, y)	x raised to power y (xy)	pow(2, 7) is 128 pow(9, .5) is 3
sin(x)	trigonometric sine of x (x in radians)	sin(0.0) is 0
sqrt(x)	square root of x	sqrt(1000.0) is 31.0 sqrt(9.0) is 3.0
tan(x)	trigonometric tangent of x (x in radians)	tan(0.0) is 0

Functions – Part 2

Recursion I

- The most common example
  - The factorial
  - Thinking recursion by knowing the main equations …
    - 0! = 1 ; n = 0
    - n! = n * (n-1)! ; n > 0

Recursion I

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

int factorial(int n)
{
        if (n == 0 )
        {
                return 1 ;
        }
        else
        {
                return n*factorial (n-1) ;
        }
}

void main(void)
{
        cout << factorial(3);
}
```

6

Recursion I

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

int factorial(int n)
{
        if (n == 0 )
                return 1 ;
        else
                return n*factorial (n-1) ;
}

void main(void)
{
        cout << factorial(3);
}
```

6

Recursion I

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

int factorial(int n)
{
        if ( n = 0 )
                return 1 ;
        else
                return n*factorial(n-1) ;
}

void main(void)
{
        cout << factorial(3);
}
```

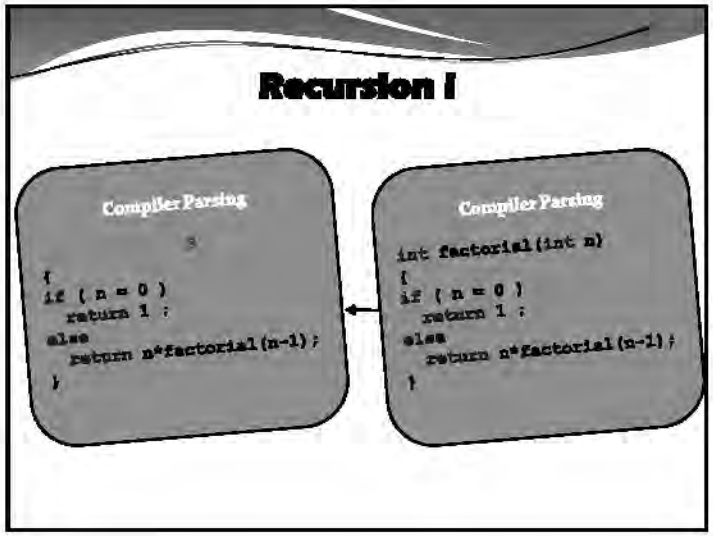
Compile but runtime error ! ... stack over flow ! "=" and not "==" !!!

So the output should 1 right?!!!

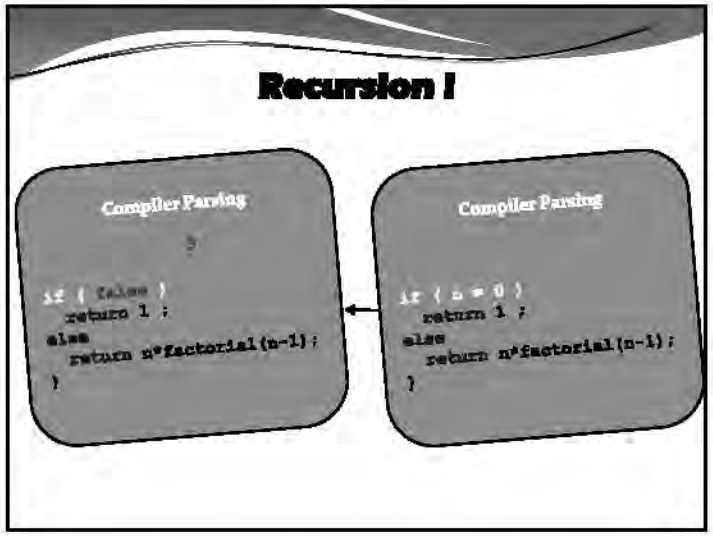
No!!!!! (n=0) is the same as if(false) and that is never enter the scope

250 pts

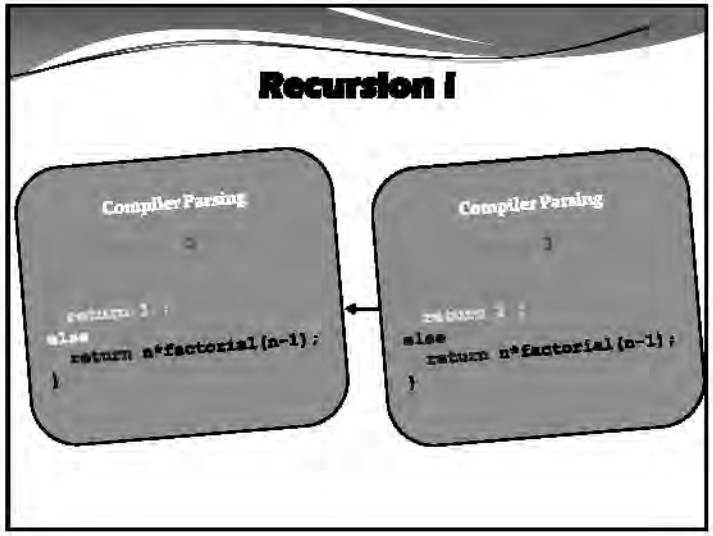
Recursion I



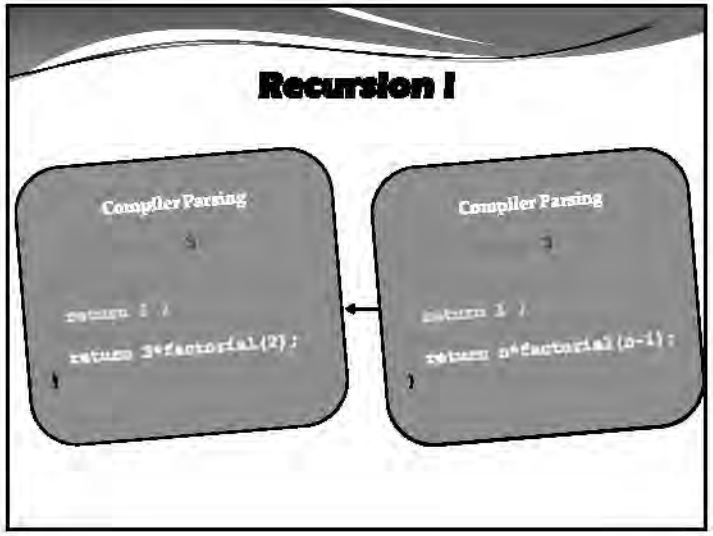
Recursion I



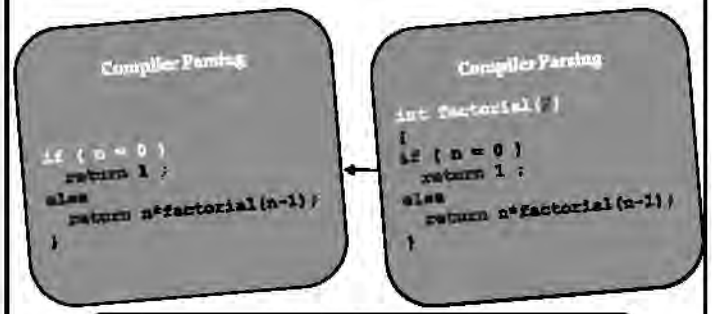
Recursion I



Recursion I

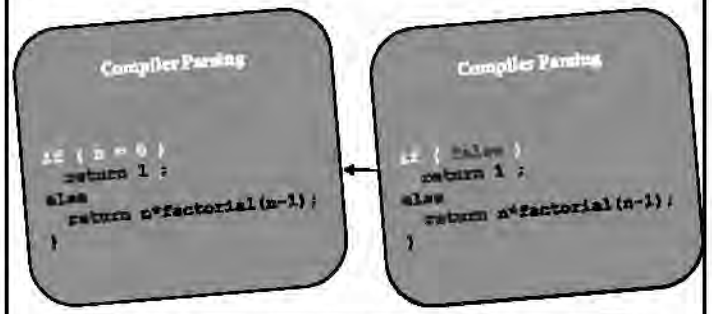


Recursion I



And the same loop over and over again without any stop condition

Recursion I



And the same loop over and over again without any stop condition

Recursion I

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

int xfactorial(int n)
{
        return 234 ;
}

void main(void)
{
        int x =  xfactorial(3);
        cout << x ;
}
```

```
234
```

```
Not a recursive function anymore !
```

Recursion I

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

int xfactorial(int n)
{
        if (n = 0 )
                return 1 ;
}

void main(void)
{
        cout << xfactorial(3);
}
```

```
0
```

Recursion I

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

int xfactorial(int n)
{
        if (n = 0 )
                return 234 ;
}

void main(void)
{
        int x =  xfactorial(3);
        cout << x ;
}
```

```
0
```

Recursion I

- The normal steps for a recursion problem
  - *Every* recursion definition must have
    - One or more "*base case*"
  - Any general case must eventually reduced to *base case*
    - Otherwise, what happened?
  - *base case* stops the Recursion
- Remember
  - Every call to a recursive function has *its own* set of parameters

Recursion I

- How to think recursively ?
  - It's the most enjoyable method when getting used to it
  - Also , sometimes it's the smartest & the hardest to figure out :)
- So , How to Recursive ?
  - *Finding a* **RULE** *that* **RULE** *the problem !*
  - Recursive it !

Recursion I

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

void MyFirstRecFun(int times)
{
        if (times > 0 )
        {
                cout << "this is a Recursive call . \n ";
                MyFirstRecFun(times - 1 );
        }
}

void main(void)
{
        int x ;
        MyFirstRecFun(4);
}
```

```
this is a Recursive call .
this is a Recursive call .
this is a Recursive call .
this is a Recursive call .
press any key to continue
```

Note the space after first line coz we have a space after \n

Recursion I

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

void MyFirstRecFun(int times)
{
        if (times >= 0 )
        {
                cout << "this is a Recursive call . \n ";
                MyFirstRecFun(times - 1 );
        }
}

void main(void)
{
        int x ;
        MyFirstRecFun(4);
}
```

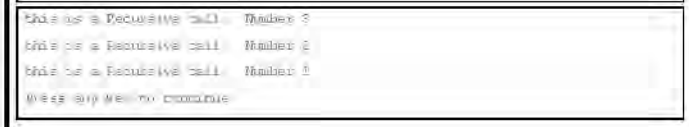
Note the space after first line coz we have a space after \n

Recursion I

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

void MyFirstRecFun(int times)
{
        if (times < 0 )
        {
                cout << "this is a Recursive call . \n ";
                MyFirstRecFun(times - 1 );
        }
}

void main(void)
{
        int x ;
        MyFirstRecFun(4);
}
```

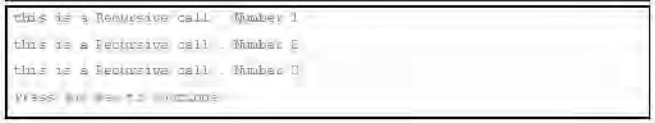
Recursion I

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

void MyFirstRecFun(int times)
{
        if (times > 0 )
        {
                cout << "this is a Recursive call . Number " << times << endl;
                MyFirstRecFun(times  - 1 );
        }
}

void main(void)
{
        int x ;
        MyFirstRecFun(3);
}
```



Recursion I

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

void MyFirstRecFun(int times)
{
        if (times > 0 )
        {
                MyFirstRecFun(times  - 1 );
                cout << "this is a Recursive call . Number " <<times<<endl;
        }
}

void main(void)
{
        int x ;
        MyFirstRecFun(3);
}
```



Recursion !

- Recursive function sum of numbers from 1 to n
- Discussion ...
  - How to do it ?
    - Finding the idea !
      - ***Rule*** to ***Rule*** !
  - We have ...
    - 1 + 2 + 3 + ... + n
    - The *Rule* is :
      - n + sum of the numbers from 1 to n-1($= \sum_{1}^{n-1}$)
        - n + summation (n-1)
      - Now we ***Ruled*** !!

Recursion !

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

int Sum(int n)
{
        if (n == 1 )
                return 1 ;
        else
                return (n + Sum(n-1));
}

void main(void)
{
        cout << Sum(4) << endl ;
}
```

```
10
```

Recursion !

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

int Sum(int n)
{
        if (n == 1 )
                return 0 ;
        else
                return (n + Sum(n-1));
}

void main(void)
{
        cout << Sum(4) << endl ;
}
```

```
6
```

Recursion !

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

int Sum(int n)
{
        if (n == 0 )
                return 1 ;
        else
                return (n + Sum(n-1));
}

void main(void)
{
        cout << Sum(4) << endl ;
}
```

```
11
```

Recursion !

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

int Sum(int n)
{
        if (n == 1 )
                return 1 ;
        else
        {
                int s ;
                return s;
        }
}

void main(void)
{
        cout << Sum(4) << endl ;
}
```

0

Not ... recursion call !!! ... [illegible]

Recursion !

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

int Sum(int n)
{
        int s ;
        if (n == 1 )
                return 1 ;
        else
        {
                s = n + Sum(n-1);
        }
}

void main(void)
{
        cout << Sum(4) << endl ;
        system("pause");
}
```

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

int Sum(int n)
{
    if (n == 1 )
        return 1 ;
    else
    {
        return n + Sum(n-1);
    }
}

void main(void)
{
    cout << Sum(4) << endl ;
    system("pause");
}
```

10 - Visual Studio 2010

Should return 2 in first pass

Recursion !

```
#include <iostream>
using namespace::std;
int s = 0 ;
int Sum(int n)
{
        if (n == 1 )
                return 1 ;
        else
                return n + 2*n*sum(n-1);
}

void main(void)
{
        cout << Sum(4) << endl ;
}
```

compiler error ... sum is small letter ! [illegible]

Recursion !

- Recursive function to find x^n
- Discussion ...
  - How to do it ?
    - Finding the idea !
      - ***Rule*** to ***Rule*** !
  - We have ...
    - 2^0 = 1 // base case
    - 2^1 = 2*2^0 2^2 = 2*2^1; 2^3 = 2*2^2; ...
    - The *Rule* is :
      - *x* * *x*^(n-1)
      - Now we ***Ruled !!***

Recursion I

```
#include <iostream>
using namespace::std;
int s = 0  ;

int power (int x , int n )
{
        if (n == 0 )
                return 1 ;
        else
        {
                return (x*power(x,n-1));
        }
}

void main(void)
{
        cout << power(2,3) << endl ;
}
```

Recursion I

```
#include <iostream>
using namespace::std;
int s = 0  ;

int power (int x , int n )
{
        if (x == 0)
                return 1 ;
        else
        {
                return (x*power(x,n-1));
        }
}

void main(void)
{
        cout << power(2,3) << endl ;
}
```

Recursion I

```
#include <iostream>
using namespace::std;
int s = 0  ;

int power (int x , int n)
{
        if (n == 0 )
                return 1 ;
        else
        {
                return (x*power(x));
        }
}

void main(void)
{
        cout << power(2,3) << endl ;
}
```

Recursion I

```
#include <iostream>
using namespace::std;
int s = 0  ;

void CrazyMessages (int times )
{
 cout << "Msg called with " << times << "\n" ;
 if (times > 0 )
 {
          cout << "This is recursive func .\n" ;
          CrazyMessages(times-1);
 }
 cout << "Msg Returning with " << times << " !!!\n";
}

void main(void)
{
        CrazyMessages (3);
}
```

```
Msg called with 3
This is recursive func .
Msg called with 2
This is recursive func .
Msg called with 1
This is recursive func .
Msg called with 0
Msg Returning with 0 !!!
Msg Returning with 1 !!!
Msg Returning with 2 !!!
Msg Returning with 3 !!!
Press any key to continue .
```

Recursion I

```
#include <iostream>
using namespace::std;
int s = 0  ;

void CrazyMessages (int times )
{
 cout << "Msg called with " << times << "\n" ;
 if (times > 0 )
 {
          cout << "This is recursive func .\n" ;
          CrazyMessages(times-1);
 }
 cout << "Msg Returning with " <<times<<" !\!!\n";
}

void main(void)
{
        CrazyMessages (3);
}
```

```
Msg called with 3
This is recursive func.
Msg called with 2
This is recursive func.
Msg called with 1
This is recursive func.
Msg called with 0
Msg Returning with 0 !!!
Msg Returning with 1 !!!
Msg Returning with 2 !!!
Msg Returning with 3 !!!
Press any key to continue
```

Recursion I

```
#include <iostream>
using namespace::std;
int s = 0  ;

void CrazyMessages (int times )
{
 cout << "Msg called with " << times << "\n" ;
 if (times >= 0 )
 {
  cout << "This is recursive func .\n" ;
  cout << "Times are : " << times << endl ;
  CrazyMessages(times-1);
 }
 cout<<"Msg Returning with "<<times<<" !!!\n";

}

void main(void)
{
        CrazyMessages (3);
}
```

```
Msg called with 3
This is recursive func.
Times are : 3
Msg called with 2
This is recursive func.
Times are : 2
Msg called with 1
This is recursive func.
Times are : 1
Msg called with 0
This is recursive func.
Times are : 0
Msg called with -1
Msg Returning with -1 !!!
Msg Returning with 0 !!!
Msg Returning with 1 !!!
Msg Returning with 2 !!!
Msg Returning with 3 !!!
Press any key to continue
```

Recursion I

```
#include <iostream>
using namespace::std;
int s = 0  ;

int power (int x , n )
{
        if (n == 0 )
                return 1 ;
        else
        {
                return (x*power(x));
        }
}

void main(void)
{
        cout << power(2,3) << endl ;
}
```

```
Compiler Error : missing int before n in function with hidden
```

Recursion VS Iteration I

- To solve a problem , 2 ways :
  - *Iteration*
    - Explicit loop
  - *Recursion*
    - Repeated function calls
- How to stop (Terminate) :
  - *Iteration*
    - Loop's condition fails
  - *Recursion*
    - Base case encountered

Recursion VS Iteration I

- Both , iteration & Recursion can have Infinite loops !
- *Performance*
  - Iteration
- *Software Engineering*
  - Recursion (Not better all the time!)

Recursion VS Iteration I

- *Memory Allocation*
  - Every recursive function call has its own sets of parameters & (automatic) local variables
  - When function is called
    - Memory space allocated
  - When function is terminated
    - Memory space de-allocated
- *Efficiency*
  - Recursive is more slowly than its iterative counterpart

Functions Default parameters

Default parameters

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

void print(int x=0 , int n=4 )
{
        cout << x << endl ;
}

void main(void)
{
        print(2,3);
}
```

```
2
```

Default parameters

```
#include <iostream>
using namespace::std;
int s = 0  ;

void print(int x=0 , int n )
{
        cout << x << endl ;
}

void main(void)
{
        print(2,3) ;
}
```

[illegible]

When defaulting a parameter in a function , all other parameters - if found - on the right side should be defaulted too

Default parameters

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

void print(int x , int n=0 )
{
        cout << x << endl
}

void main(void)
{
        print(2,3);
}
```

2

Default parameters

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

void print(int x , int n , int y )
{
        cout << x << endl ;
}

void main(void)
{
        print(2,3,4) ;
}
```

2

[illegible]

Default parameters

```
#include <iostream>
using namespace::std;
int s = 0  ;

void print(int x , int n , int y = 0   )
{
        cout << x << endl ;
}

void main(void)
{
        print(2,3,4);
}
```

2

[illegible]

Default parameters

```
#include <iostream>
using namespace::std;
int s = 0  ;

void print(int x , int n , int y = 0  )
{
        cout << x << endl ;
}

void main(void)
{
        print(2,3);
}
```

```
2
```

[illegible]

Default parameters

```
#include <iostream>
using namespace::std;
int s = 0  ;

void print (int x , int n , int y=10 )
{
        cout << x << endl ;
        cout << n << endl ;
        cout << y << endl ;
}

void main(void)
{
        print(2,3);
}
```

```
2
3
10
```

Default parameters

```
#include <iostream>
using namespace::std;
int s = 0  ;

void print(int x , int n=10 , int y )
{
        cout << x << endl ;
        cout << n << endl ;
        cout << y << endl ;
}

void main(void)
{
        print(2,3);
}
```

Compile error .. coz we did't default the y

Now .. did u know why ^_^

Default parameters

```
#include <iostream>
using namespace::std;
int s = 0  ;

void print(int x , int n=10 , int y )
{
        cout << x << endl ;
        cout << n << endl ;
        cout << y << endl ;
}

void main(void)
{
        print(2,3);
}
```

Why the compiler couldn't do it?

Compiler error .. coz we did't default the y

Now .. did u know why ^_^

Default parameters

```
#include <iostream>
using namespace::std;
int s = 0  ;

void print(int x , int n=10
{
        cout << x << endl ;
        cout << n << endl ;
        cout << y << endl ;
}

void main(void)
{
        print(2,3);
}
```

Compiler Parsing

```
void print(int x, int n=10, int y);

void main(void)
{
        print(2,3);
}
```

Compiler error .. cos we didn't default the y ..

Now , did u know why ? :)

Default parameters

```
#include <iostream>
using namespace::std;
int s = 0  ;

void print(int x , int n=10
{
        cout << x << endl ;
        cout << n << endl ;
        cout << y << endl ;
}

void main(void)
{
        print(2,3);
}
```

Compiler Parsing

```
print(2,3);
}
```

Compiler error .. cos we didn't default the y ..

Now , did u know why ? :)

Default parameters

```
#include <iostream>
using namespace::std;
int s = 0  ;

void print(int x , int n=10
{
        cout << x << endl ;
        cout << n << endl ;
        cout << y << endl ;
}

void main(void)
{
        print(2,3);
}
```

Compiler Parsing

```
print( ,3);
}
```

Compiler error .. cos we didn't default the y ..

Now , did u know why ? :)

Default parameters

```
#include <iostream>
using namespace::std;
int s = 0  ;

void print(int x , int n=10
{
        cout << x << endl ;
        cout << n << endl ;
        cout << y << endl ;
}

void main(void)
{
        print(2,3);
}
```

Compiler Parsing

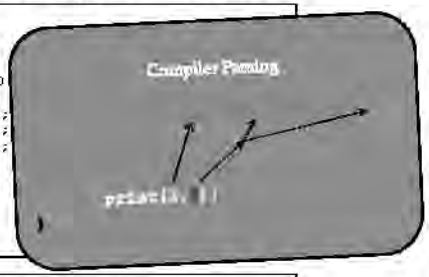
```
print(2, );
}
```

Compiler error .. cos we didn't default the y ..

Now , did u know why ? :)

Default parameters

```
#include <iostream>
using namespace::std;
int s = 0  ;

void print(int x , int n=10
{
        cout << x << endl ;
        cout << n << endl ;
        cout << y << endl ;
}

void main(void)
{
        print(2,3);
}
```

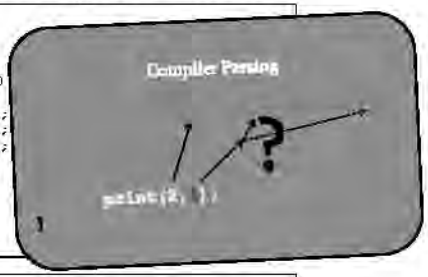


Compiler error … coz we didn't default the y …

Now … did u know why ? :)

Default parameters

```
#include <iostream>
using namespace::std;
int s = 0  ;

void print(int x , int n=10
{
        cout << x << endl ;
        cout << n << endl ;
        cout << y << endl ;
}

void main(void)
{
        print(2,3);
}
```

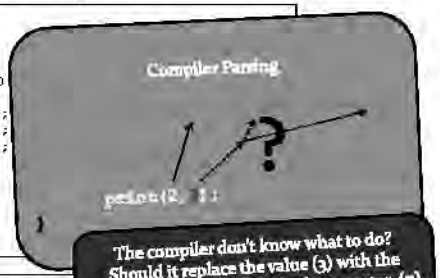


Compiler error … coz we didn't default the y …

Now … did u know why ? :)

Default parameters

```
#include <iostream>
using namespace::std;
int s = 0  ;

void print(int x , int n=10
{
        cout << x << endl ;
        cout << n << endl ;
        cout << y << endl ;
}

void main(void)
{
        print(2,3);
}
```



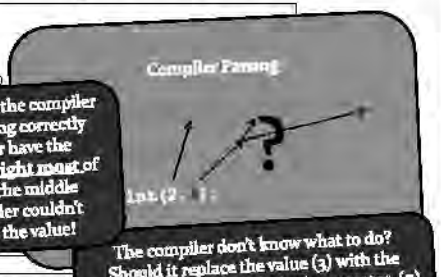
The compiler don't know what to do? Should it replace the value (3) with the default value of the defaulted parameter (n) or go on to the next parameter (y) and pass it through

Compiler error … coz we didn't defau…

Now … did u know why ? :)

Default parameters

```
#include <iostream>
using namespace::std;
int s = 0  ;

void print(int x , int n=10
{
```



And now you can know why the compiler can parse the function calling correctly when the function header have the defaulted parameter at the right most of the prototype and not in the middle coz as you saw, the compiler couldn't determine where to pass the value!

The compiler don't know what to do? Should it replace the value (3) with the default value of the defaulted parameter (n) or go on to the next parameter (y) and pass it through

Compiler error … coz we didn't defau…

Now … did u know why ? :)

Default parameters

```
#include <iostream>
using namespace::std;
int s = 0  ;

void print(int x , int n=10 , int y = 5 )
{
        cout << x << endl ;
        cout << n << endl ;
        cout << y << endl ;
}

void main(void)
{
        print(2,3);
}
```

Calling by value VS calling by reference

Very Important

Used to be ***Massacre***

Calling by value VS calling by reference

- **Calling by value**
  - *Copy* of data
  - Changes to copy don't affect original
  - Prevent an unwanted side effect
- **Calling by reference (&)**
  - Function directly access data
  - Changes affect original (no copy exist here !)
  - Using & *after* data type

```
void Momo(double & x)
```

Calling by value VS calling by reference

```
#include <iostream>
using namespace::std;
int s = 0  ;

void FuncByValue (int x )
{
        cout << "In function before multiplication , x = "<< x << endl ;
        x = x*4 ;
        cout << "In function after multiplication , x = " << x << endl ;
}

void main(void)
{
        int x = 2 ;
        cout << "In main , before calling the function " << x << endl ;
        FuncByValue(x) ;
        cout << "In main , after calling the function " << x << endl ;
}
```

```
In main , before calling the function 2
In function before multiplication , x = 2
In function after multiplication , x = 8
In main , after calling the function 2
Press any key to continue . . .
```

Calling by value VS calling by reference

```
#include <iostream>
using namespace::std;
int s = 0  ;

void FuncByRef (int & x )
{
        cout << "In function before multiplication , x = "<< x << endl ;
        x = x*4 ;
        cout << "In function after multiplication , x = " << x << endl ;
}

void main(void)
{
        int x = 2 ;
        cout << "In main , before calling the function " << x << endl ;
        FuncByRef (x) ;
        cout << "In main , after calling the function " << x << endl ;

}
```

```
In main , before calling the function: 2
In function before multiplication , x = 2
In function after multiplication , x = 8
In main , after calling the function: 8
Press any key to continue
```

Calling by value VS calling by reference

```
#include <iostream>
using namespace::std;
int s = 0  ;

void FuncByRef (& int  x )
{
        cout << "In function before multiplication , x = "<< x << endl ;
        x = x*4 ;
        cout << "In function after multiplication , x = " << x << endl ;
}

void main(void)
{
        int x = 2 ;
        cout << "In main , before calling the function " << x << endl ;
        FuncByRef (x) ;
        cout << "In main , after calling the function " << x << endl ;

}
```

```
Compiler error .. & before type
```

Calling by value VS calling by reference

```
#include <iostream>
using namespace::std;
int s = 0  ;

void FuncByRef ( int  x & )
{
        cout << "In function before multiplication , x = "<< x << endl ;
        x = x*4 ;
        cout << "In function after multiplication , x = " << x << endl ;
}

void main(void)
{
        int x = 2 ;
        cout << "In main , before calling the function " << x << endl ;
        FuncByRef (x) ;
        cout << "In main , after calling the function " << x << endl ;

}
```

```
Compiler error .. & after variable
```

Calling by value VS calling by reference

```
#include <iostream>
using namespace::std;
int s = 0  ;

int FuncByValue (  int x )
{
        cout << "In function before multiplication , x = "<< x << endl ;
        x = x*4 ;
        cout << "In function after multiplication , x = " << x << endl ;
        return x ;
}

void main(void)
{
        int x = 2 ;
        cout << "In main , before calling the function , x = " << x << endl ;
        cout << FuncByValue(x) << endl ;
        cout << "In main , after calling the function , x = " << x << endl ;

}
```

```
In main , before calling the function , x = 2
In function before multiplication , x = 2
In function after multiplication , x = 8
8
In main , after calling the function , x = 2
Press any key to continue
```

Calling by value VS calling by reference

```
#include <iostream>
using namespace::std;
int g = 0  ;

int FuncByRef (  int & x )
{
        cout << "In function before multiplication , x = "<< x << endl ;
        x = x*4 ;
        cout << "In function after multiplication , x = " << x << endl ;
        return x ;
}

void main(void)
{
        int x = 2 ;
        cout << "In main , before calling the function , x = " << x << endl ;
        cout << FuncByRef(x) << endl ;
        cout << "In main , after calling the function , x = " << x << endl ;

}
```

Function Overloading

Very Important

Used to be ***Massacre***

Function Overloading

- Is a function with
  - ***same name***
  - *different* ***parameters*** ***(Not return type !!!!!!!)***
- That means that the overloaded functions are distinguished in **Signature**
  - ***No return type comparing***

Function Overloading

```
#include <iostream>
using namespace::std;
int g = 0  ;

int f (  int &x )
{
        x = x * 2 ;
        return x ;
}

int f (  float &x )
{
        x = x * 2 ;
        return x ;
}

void main(void)
{
        int y1 = 2 ;
        float y2 = 2 ;

        cout << f(y1) << endl ;
        cout << f(y2) << endl ;
}
```

Function Overloading

```
#include <iostream>
using namespace::std;
int z = 0  ;

int f (  int &x )
{
          x = x * 2 ;
          return x ;
}

float f (  float &x )
{
          x = x * 3 ;
          return x ;
}

void main(void)
{
          int y1 = 2 ;
          float y2 = 2 ;

          cout << f(y1) << endl ;
          cout << f(y2) << endl ;
}
```

Function Overloading

```
#include <iostream>
using namespace::std;
int z = 0  ;

int f (  int x )
{
          x = x * 2 ;
          return x ;
}

float f (  float x )
{
          x = x * 3 ;
          return x ;
}

void main(void)
{
          int y1 = 2 ;
          float y2 = 2 ;

          cout << f(y1) << endl ;
          cout << f(y2) << endl ;
}
```

Function Overloading

```
#include <iostream>
using namespace::std;
int z = 0  ;

int f (  int x )
{
          x = x * 2 ;
          return x ;
}

float f (  float x )
{
          x = x * 0 ;
          return x ;
}

void main(void)
{
          int y1 = 2 ;
          float y2 = 2 ;

          cout << f(y1) << endl ;
          cout << f(y2) << endl ;
}
```

Function Overloading

```
#include <iostream>
using namespace::std;
int z = 0  ;

int f (  int x , int z)
{
          x = x * 2 ;
          cout << x << endl ;
          cout << z << endl ;
          return x ;
}

float f (float x , int z)
{
          x = x * 3 ;
          cout << x << endl ;
          cout << z << endl ;
          return x ;
}

void main(void)
{
          int y1 = 2 ;
          float y2 = 2 ;
          int z1 = 3 , z2 = 2 ;

          f(y1,z1);
          f(y2,z2);
}
```

Function Overloading

```
#include <iostream>
using namespace::std;
int z = 0  ;

int f (  int x , int z)
{
        x = x * 2 ;
        cout << x << endl ;
        cout << z << endl ;
        return x ;
}

float f (float x , int z)
{
        x = x * 3 ;
        cout << x << endl ;
        cout << z << endl ;
        return x ;
}

void main(void)
{
        int y1 = 2 ;
        float y2 = 2 ;
        float z1 = 3 , z2 = 2 ;

        f(y1,z1);
        f(y2,z2);
}
```

Function Overloading

```
#include <iostream>
using namespace::std;
int z = 0  ;

int f (  int x , int z)
{
        x = x * 2 ;
        cout << x << endl ;
        cout << z << endl ;
        return x ;
}

float f (int z , float x)
{
        x = x * 3 ;
        cout << x << endl ;
        cout << z << endl ;
        return x ;
}

void main(void)
{
        int y1 = 2 ;
        float y2 = 2 ;
        int z1 = 3 , z2 = 2 ;

        f(y1,z1);
        f(y2,z2);
}
```

Function Overloading

```
#include <iostream>
using namespace::std;
int z = 0  ;

int f (  int x , int z)
{
        x = x * 2 ;
        cout << x << endl ;
        cout << z << endl ;
        return x ;
}

float f (int z , float x)
{
        x = x * 3 ;
        cout << x << endl ;
        cout << z << endl ;
        return x ;
}

void main(void)
{
        int y1 = 2 ;
        float y2 = 2 ;
        float z1 = 3 , z2 = 2 ;

        f(y1,z1);
        f(y2,z2);
}
```

Code Cracking

Code Cracking

```
#include <iostream>
using namespace::std;
int x = 0  ;

float f2 (int z )
{
        cout << z << endl ;
        z*=2 ;
        return z ;
}

void f1 (  int &x )
{
        x+2 ;
        cout << x << endl ;
        f2(x) ;
        cout << x << endl ;
}


void main(void)
{
        f1(::x++) ;
        cout << ++::x << endl ;
}
```

Compiler error ... Can't convert from int to &int

Code Cracking

```
#include <iostream>
using namespace::std;
int x = 0  ;


void f1 (  int &x )
{
        x+2 ;
        cout << x << endl ;
        f2(x) ;
        cout << x << endl ;
}

float f2 (int z )
{
        cout << z << endl ;
        z*=2 ;
        return z ;
}


void main(void)
{
        f1(::x) ;
        cout << ::x << endl ;
}
```

Compiler error ... f1 can't > f2 work

We're finished for today !

Recursion I - #1

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

void MyFirstRecFun(int times)
{
        if (times < 0 )
        {
                cout << "this is a Recursive call . Number " << times
<< endl ;
                MyFirstRecFun(times +1 );
        }
}

void main(void)
{
        int x ;
        MyFirstRecFun(-3);
}
```

```
this is a Recursive call . Number -3
this is a Recursive call . Number -2
this is a Recursive call . Number -1
Press any key to continue . . .
```

Recursion I - #2

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

void MyFirstRecFun(int times)
{
        if (times < 0 )
        {
                cout << "this is a Recursive call , Number " << times
<< endl ;
                MyFirstRecFun(times - 1 );
        }
}

void main(void)
{
        int x ;
        MyFirstRecFun(-3);
}
```

Recursion I - #3 Quiz

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

int Sum(int n)
{
        if (n == 0 )
                return 0 ;
        else
                return (n + Sum(n-1));
}

void main(void)
{
        cout << Sum(4) << endl ;
}
```

10

Recursion I - #4

```
#include <iostream>
using namespace::std;
int s = 0  ;

void CrazyMessages (int times )
{

 cout << "Msg called with " << times << "\n" ;
 if (times >= 0 )
 {
  cout << "This is recursive func .\n" ;
  cout << "Times are : " << times-- << endl ;
        CrazyMessages(times-1);
 }
 cout<<"Msg Returning with "<<times<<" !\!!\n";

}

void main(void)
{
        CrazyMessages (3);
}
```

static I - #5

- Let's see this

```
#include <iostream>

using namespace::std;
int x = 84 ;

void ZeZeCrazySoMuch ()
{
        static int x = 3 ;     // initialized only ONE time
        x = 5 ;
        cout << "first x in ZeZe = " << x <<  endl ;
        x++;
        cout << "second x in ZeZe = " << x <<  endl ;
}
```

Static with scopes I - #5

```
void main(void)
{
        int x = 9 ;
        cout << x << endl ;
                {
        {
                int x = 5 ;
                cout << x++ << endl ;
                }
        cout << x << endl ;
        ::x = x+2 ;
                }
        cout << x << endl ;
        {
        ZeZeCrazySoMuch ();
        ZeZeCrazySoMuch ();}
        cout << x << endl ;
        cout << ::x << endl ;
}
```

What's the output ?

Playing with scopes I

```
9
6
5
9
first x in ZeZe = [illegible]
second x in ZeZe = 6
first x in ZeZe = 5
second x in ZeZe = 6
9
11
Press any key to continue
```

Twisted! #6

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

void TryMe();

void main()
{
    cout << "In main" << endl;
    TryMe() ;
}

void TryMe()
{
    cout << "In function" << endl;
    main();
}
```

```
In main
In function
In main
In function
In main
In function
In main
In function

In main
In function
```

Non definitely

Twisted! #7

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

void TryMe()
{
    cout << "In function" << endl;
    main();
}

void main()
{
    TryMe() ;
    cout << "In main" << endl;
}
```

Compiler error!
TryMe() can't see the main function!

Tweaked Control Structure I

- If \ else
- While
- do \ while
  - *Executed at least once whatever the condition is .*
- for
- switch

Tweaked Control Structure I

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

void main(void)
{
        int x = 4 ;
        if ( x == 4 )
        {
                cout << "True " << endl ;
        }
        else
        {
                cout << "False " << endl ;
        }
}
```

```
True
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 4 ;
        if ( x = 4 )
        {
                cout << "True " << endl ;
        }
        else
        {
                cout << "False " << endl ;
        }
}
```

```
True
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 4 ;
        if ( x = 5 )
        {
                cout << "True " << endl ;
        }
        else
        {
                cout << "False " << endl ;
        }
}
```

```
True
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 4 ;
        if ( x != 4 )
        {
                cout << "True " << endl ;
        }
        else
        {
                cout << "False " << endl ;
        }
}
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 4 ;
        if ( x == 4 )
                cout << "True " << endl ;
        else
                cout << "False " << endl ;
                cout << "GoGo !" << endl ;
}
```

Tweaked Control Structure I

```
#include <iostream>
using namespace::std;
int a = 0  ;

void main(void)
{
     int x = 0;
     if (x = 0)
     {
          cout << "I don't know" << endl ;
     }
     cout << "I know" << endl ;
     system("pause");
}
```

a = 0
Means False!

Remember that every other number other than 0 means true

Tweaked Control Structure I

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

void main(void)
{
        int x = 5 ;
        while (x != 10)
        {
                x++;
                cout << x << endl ;
        }
}
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 5 ;
        while (x != 10)
        {
                cout << x++ << endl ;
        }
}
```

```
5
6
7
8
9
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 5 ;
        while (x != 10)
        {
                cout << ++x << endl ;
        }
}
```

```
6
7
8
9
10
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 5 ;
        while (x != 10)
        {
                ++x;
                cout << x << endl ;
        }
}
```

```
6
7
8
9
10
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 5 ;
        while (x < 10)
        {
                cout << x*x << endl ;
        }
}
```

```
36
25
49
36
64
25
```

Tweaked Control Structure I

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

void main(void)
{
	int x = 5 ;
	while (x*x < 10)
	{
		cout << x*x << endl ;
	}
}
```

```
Nothing to start
```

Tweaked Control Structure I

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

void main(void)
{
	int x = 5 ;
	while (cin)
	{
		cin >> x ;
		cout << x*x << endl ;
	}
}
```

[illegible]

Tweaked Control Structure I

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

void main(void)
{
	int x = 5 ;
	do
	{
		cout << x << endl ;
		x++;
	} while ( x < 10 ) ;
}
```

```
5
6
7
8
9
Press any key to continue
```

Tweaked Control Structure I

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

void main(void)
{
	int x = 5 ;
	do
	{
		cout << x << endl ;
		x++;
	} while ( x < 10 )
}
```

```
Compiler error : missing ';' after the while ( );
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 5 ;
        do
        {
                cout << x << endl ;
        } while ( x < 4 ) ;
}
```

```
5
```

do while execute at lest one , eventhough the condition is not fulfilled .

Tweaked Control Structure I

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

void main(void)
{

int x = 5 ;
do
{
cout << "Enter the x :" << endl ;
cin >> x ;
cout << "You have entered , x = " << x
<< endl ;
cout << "_____"
<< endl ;
} while ( x != 0  ) ;


}
```

```
Enter the x :
5
You have entered , x = 5
_____
Enter the x :
4
You have entered , x = 4
_____
Enter the x :
3
You have entered , x = 3
_____
Enter the x :
0
You have entered , x = 0
_____
Press any key to continue . . .
```

Tweaked Control Structure I

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

void main(void)
{
        int i = 0 ;
        int c = 5 ;
        do
        {
                ++c ;
                cout << "beep ! , number = " << i << endl ;
                i++;
        } while ( ++c < 10 ) ;
}
```

```
beep ! , number = 0
beep ! , number = 1
beep ! , number = 2
Press any key to continue . . .
```

Tweaked Control Structure I

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

void main(void)
{
        int i = 0 ;
        int c = 5 ;
        do
        {
                ++c ;
                cout << "beep ! , number = " << i << endl ;
                i++;
        } while ( c++ < 10 ) ;
}
```

```
beep ! , number = 0
beep ! , number = 1
beep ! , number = 2
Press any key to continue . . .
```

Tweaked Control Structure I

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

void main(void)
{
        int i = 0 ;
        int c = 5 ;
        do
        {
                ++c ;
                cout << "beep ! , number = " << i << endl ;
                i++;
        } while ( ++c <= 10 ) ;
}
```

```
beep ! , number = 0
beep ! , number = 1
beep ! , number = 2
Press any key to continue
```

Tweaked Control Structure I

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

void main(void)
{
        for (int i = 0 ; i < 10 ; i++)
        {
        cout<<"beep ! , number = "<<i<<endl;
        }
        cout << "finished" << endl ;
}
```

```
beep ! , number = 0
beep ! , number = 1
beep ! , number = 2
beep ! , number = 3
beep ! , number = 4
beep ! , number = 5
beep ! , number = 6
beep ! , number = 7
beep ! , number = 8
beep ! , number = 9
finished
Press any key to continue
```

Tweaked Control Structure I

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

void main(void)
{
        int i = 5 ;
        for (int i = 0 ; i < 10 ; i++)
        {
        cout << "beep ! , number = " << i
        << endl ;
        }
        cout << "finished" << endl ;
}
```

```
beep ! , number = 0
beep ! , number = 1
beep ! , number = 2
beep ! , number = 3
beep ! , number = 4
beep ! , number = 5
beep ! , number = 6
beep ! , number = 7
beep ! , number = 8
beep ! , number = 9
finished
Press any key to continue
```

Tweaked Control Structure I

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

void main(void)
{
        int i = 5 ;
        int c = 5 ;
        for (int i = 0 ; i < 10 ; i++)
        {
                cout << "beep ! , number = "
                << i << endl ;
                i++;
        }
        cout << "finished" << endl ;
}
```

```
beep ! , number = 0
beep ! , number = 2
beep ! , number = 4
beep ! , number = 6
beep ! , number = 8
finished
```

Tweaked Control Structure I

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

void main(void)
{
        int i = 66 ;
        int c = 5 ;
        for (int i = 0 ; i < 10 ; i++);
        {
                cout << "beep ! , number = " << i << endl ;
                i++;
        }
        cout << "finished" << endl ;
}
```

```
beep ! , number = 66
finished
Press any key to continue
```

Why is that ?

Watch out for the semi colon ";" after the for statement ..Coz it close it

Tweaked Control Structure I

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

void main(void)
{
        int i = 66 ;
        int c = 5 ;
        for (int i = 0 ; i < 10 ; )
        {
                cout << "beep ! , number = " << i << endl ;
                i++;
        }
        cout << "finished" << endl ;
}
```

```
beep ! , number = 0
beep ! , number = 1
beep ! , number = 2
beep ! , number = 3
beep ! , number = 4
beep ! , number = 5
beep ! , number = 6
beep ! , number = 7
beep ! , number = 8
beep ! , number = 9
finished
Press any key to continue
```

Tweaked Control Structure I

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

void main(void)
{
        int i = 66 ;
        int c = 5 ;
        for (int i = 0 ; i < 10 ; );
        {
                cout << "beep ! , number = " << i << endl ;
                i++;
        }
        cout << "finished" << endl ;
}
```

Nothing to print ... Why ?

Coz he is still working in an un-infinite loop !

Tweaked Control Structure I

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

void main(void)
{
        int i = 66 ;
        int c = 5 ;
        for (int i = 0 ;  ; )
        {
                cout << "beep ! , number = " << i << endl ;
                i++;
        }
        cout << "finished" << endl ;
}
```

```
beep ! , number = 0
beep ! , number = 1
beep ! , number = 2
beep ! , number = 3

beep   number = 9

beep   number = [illegible]



Un-infinite loop !
```

Tweaked Control Structure I

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

void main(void)
{
        int i = 66 ;
        int c = 5 ;
        for (int i = 0 ;  ; i++ )
        {
                cout << "beep ! , number = " << i << endl ;
                i++;
        }
        cout << "finished" << endl ;
}
```

```
beep ! , number = 0
beep ! , number = 2
beep ! , number = 4
beep ! , number = 6
..
beep ! , number = 10

beep ! , number = 65536
..

On-target loop
```

Tweaked Control Structure I

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

void main(void)
{
        int i = 66 ;
        int c = 5 ;
        for (  )
        {
                cout << "beep ! , number = " << i << endl ;
        }
        cout << "finished" << endl ;
}
```

```
Compile error
```

Tweaked Control Structure I

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

void main(void)
{
        int i = 66 ;
        int c = 5 ;
        for (i = 2 ; i<5 ; i++ )
        {
                cout << "beep ! , number = " << i << endl ;
        }
        cout << "finished" << endl ;
}
```

```
beep ! , number = 2
beep ! , number = 3
beep ! , number = 4
finished
Press any key to continue
```

Tweaked Control Structure I

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

void main(void)
{
        int i = 66 ;
        int c = 5 ;
        for (int i = 0 , j = 3 ; i<5 ; i++ )
        {
                cout << "beep ! , number = " << j++ << endl ;
        }
        cout << "finished" << endl ;
}
```

```
beep ! , number = 3
beep ! , number = 4
beep ! , number = 5
beep ! , number = 6
beep ! , number = 7
finished
Press any key to continue
```

Tweaked Control Structure I

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

void main(void)
{
        int i = 66 ;
        int c = 5 ;
        for (int i = 0 ; j = 3 ; i<5 ; i++ )
        {
                cout << "beep ! , number = " << j++ << endl ;
        }
        cout << "finished" << endl ;
}
```

Compiler error : ; unexpected

Tweaked Control Structure I

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

void main(void)
{
        int i = 66 ;
        int c = 5 ;
        for (int i = 0 , int j = 3 ; i<5 ; i++ )
        {
                cout << "beep ! , number = " << j++ << endl ;
        }
        cout << "finished" << endl ;
}
```

Compiler error : j unresolved identifier

Tweaked Control Structure I

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

void main(void)
{
        int i = 66 ;
        int c = 5 ;
        int j = 2 ;
        for (int i = 0 , j ; i<5 ; i++ , j-- )
        {
                cout << j << endl ;
        }
}
```

0
-1
-2
-3
-4

Tweaked Control Structure I

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

void main(void)
{
        int i = 66 ;
        int c = 5 ;
        int j = 4 ;
        for (int i = 0 , j=0 ; i<5 ; i++ , j-- )
        {
                cout << j++ << endl ;
        }
}
```

0
1
0
1
0

Tweaked Control Structure I

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

void main(void)
{
        for (int i = 0  ; i<5 ; i++  )
        {
                cout << i++ << endl ;
                break ;
        }
}
```

```
0
```

Tweaked Control Structure I

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

void main(void)
{
        for (int i = 0  ; i<5 ; i++  )
        {
                cout << i++ << endl ;
                if ( i == 34)
                {
                        break ;
                }
        }
}
```

```
0
2
4
```

Tweaked Control Structure I

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

void main(void)
{
        for (int i = 0  ; i<5 ; i++  )
        {
                cout << i << endl ;
                if ( i == 34)
                {
                        break ;
                }
        }
}
```

```
0
1
2
3
4
```

Tweaked Control Structure I

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

void main(void)
{
        for (int i = 0  ; i<5 ; i++  )
        {
                cout << i << endl ;
                if ( i == 2)
                {
                        break ;
                }
        }
}
```

```
0
1
2
```

Tweaked Control Structure I

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

void main(void)
{
        for (int i = 0  ; i<5 ; i++  )
        {
                if ( i == 2)
                {
                        break ;
                }
                cout << i << endl ;

        }
}
```

```
0
1
```

Tweaked Control Structure I

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

void main(void)
{
        for (int i = 0  ; i<5 ; i++  )
        {
                cout << i << endl ;
                if ( i = = 2)
                {
                        break ;
                }
        }
}
```

```
Compilation error ... = = not ==
```

Tweaked Control Structure I

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

void main(void)
{
        for (int i = 0  ; i<5 ; i++  )
        {
                cout << i << endl ;
                if ( i == 2)
                {
                        cout << "WEWEEEE !!!\";
                        break ;
                }
        }
}
```

```
0
1
2
WEWEEEE !!!Press any key to continue
```

Tweaked Control Structure I

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

void main(void)
{
        for (int i = 0  ; i<5 ; i++  )
        {
                if ( i == 2)
                {
                        continue ;
                }
                cout << i << endl ;
        }
}
```

```
0
1
3
4
```

Tweaked Control Structure I

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

void main(void)
{
        for (int i = 0  ; i<5 ; i++  )
        {
                if ( i == 2)
                {
                        continue ;
                        cout << "WeeWeee" << endl ;
                }
                cout << i << endl ;
        }
}
```

```
0
1
3
4
```

Tweaked Control Structure I

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

void main(void)
{
        for (int i = 0  ; i<5 ; i++  )
        {
                cout << i << endl ;
                if ( i == 2)
                {
                        cout << "WeeeWeee" << endl ;
                        continue ;
                }
        }
}
```

```
0
1
2
WeeeWeee
3
4
Press any key to continue . . .
```

Tweaked Control Structure I

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

void main(void)
{
        for (int i = 0  ; i<5 ; i++  )
        {
                cout << i << endl ;
                if ( i == 2)
                {
                        cout << "WeeeWeee" << endl ;
                }
        }
}
```

```
0
1
2
WeeeWeee
3
4
Press any key to continue . . .
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 3 ;
        cout << "Enter x :  " << endl ;
        cin >> x ;
        cout << "You have entered , x = " << x << endl ;
        switch (x)
        {
        case 1 :
         cout << "WOW , i can't believe it !, you entered x = 1" << endl;
         break ;
        case 2 :
         cout << "WOW , i can't believe it !, you entered x = 2" << endl ;
         break ;
        }
}
```

```
Enter x :
3
You have entered , x = 3
WOW , i can't believe it !, you entered x = 2
Press any key to continue . . .
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 3 ;
        cout << "Enter x : " << endl ;
        cin >> x ;
        cout << "You have entered , x = " << x << endl ;
        switch (x)
        {
        case 1 :
         cout << "WOW , i can't believe it , you entered x = 1" << endl;
         break ;
        case 2 :
         cout << "WOW , i can't believe it , you entered x = 2" << endl ;
         break ;
        }
}
```

```
Enter x :
3
You have entered , x = 3
Press any key to continue . . .
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 3 ;
        cout << "Enter x : " << endl ;
        cin >> x ;
        cout << "You have entered , x = " << x << endl ;
        switch (x)
        {
        case 1 :
                cout << "WOW , i can't believe it , you entered x = 1" << endl ;
                break ;
        case 2 :
                cout << "WOW , i can't believe it , you entered x = 2" << endl ;
                 break ;
        default :
                cout << "Not a 1 or 2 " << endl;
                 break ;
        }
}
```

```
Enter x :
3
You have entered , x = 3
Not a 1 or 2
Press any key to continue . . .
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 3 ;
        cout << "Enter x : " << endl ;
        cin >> x ;
        cout << "You have entered , x = " << x << endl ;
        switch (x)
        {
        case 1 :
                cout << "WOW , i can't believe it , you entered x = 1" << endl ;
                break ;
        case 2 :
                cout << "WOW , i can't believe it , you entered x = 2" << endl ;
                 break ;
        default :
                cout << "Not a 1 or 2 " << endl;
        }
}
```

```
Enter x :
3
You have entered , x = 3
Not a 1 or 2
Press any key to continue . . .
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 3 ;
        cout << "Enter x : " << endl ;
        cin >> x ;
        cout << "You have entered , x = " << x << endl ;
        switch (x)
        {
        case 1 :
                cout << "WOW , i can't believe it , you entered x = 1"
                << endl ;
        case 2 :
                cout << "WOW , i can't believe it , you entered x = 2"
                << endl ;
        default :
                cout << "Not a 1 or 2 " << endl;
        }
}
```

Coz of forgetting break

```
Enter x :
1
You have entered , x = 1
WOW , i can't believe it , you entered x = 1
WOW , i can't believe it , you entered x = 2
Not a 1 or 2
Press any key to continue . . .
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 3 ;
        cout << "Enter x : " << endl ;
        cin >> x ; // 1
        cout << "You have entered , x = " << x << endl ;
        switch (x)
        {
        case 1 :
                cout << "WOW , i can't believe it , you entered x = 1" << endl ;
                cout << x++ << endl;
        case 2 :
                cout << "WOW , i can't believe it , you entered x = 2" << endl ;
        default :
                cout << "Not a 1 or 2 " << endl;
        }
}
```

```
Enter x :
1
You have entered , x = 1
WOW , i can't believe it , you entered x = 1
1
WOW , i can't believe it , you entered x = 2
Not a 1 or 2
Press any key to continue . . .
```

Tweaked Control Structure I

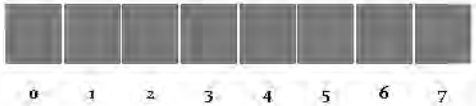
```
#include <iostream>
using namespace::std;

void main(void)
{
        int x = 3 ;
        cout << "Enter x : " << endl ;
        cin >> x ;
        cout << "You have entered , x = " << x << endl ;
        switch (x)
        {
        case 1 :
                cout << "WOW , i can't believe it , you entered x = 1" << endl ;
                cout << x++ << endl;
                break ;
        case 2 :
                cout << "WOW , i can't believe it , you entered x = 2" << endl ;
                break ;
        default :
                cout << "Not a 1 or 2 " << endl;
        }
}
```

```
Enter x :
1
You have entered , x = 1
WOW , i can't believe it , you entered x = 1
1
Press any key to continue . . .
```

Arrays

C++ data types

Arrays

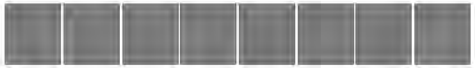


Note : nth element in position **n-1**

Arrays

- Declaring arrays ....

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

void main(void)
{
        int A [1000] ; // means the position are from 0 to 999
}
```



Arrays

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


void main(void)
{
     int intArr [10] = {4,5,56,6} ;
     cout << intArr << endl ;
     system("pause");
}
```

```
0048FD08
Press any key to continue
```

When printing the array it print it's location on memory by default

Just the (char arr []) Don't do that! It prints the char contained in it! We'll see it in minutes ;)

The array "variable" is the location of its first element in memory

Arrays

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


void main(void)
{
     int intArr [10] = {4,5,56,6} ;
     cout << intArr[0] << endl ;
     system("pause");
}
```

```
4
Press any key to continue
```

When running the array it print it's location on memory by default

Just the (char arr []) Don't do that! It prints the char contained in it! We'll see it in minutes ;)

Arrays

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

void main(void)
{
        int A [5] ;
        A[3] = 0 ;
        A[0] = -1 ;
        for (int i = 0 ; i<=4 ; i ++)
        {
                cout << "A[" << i << "] = " << A[i] << endl ;
        }
}
```

```
A[0] = -1
A[1] = 2092880
A[2] = 5040544
A[3] = 0
A[4] = 1523006256
Press any key to continue
```

Note that it's not a compiler error ... i ++

Arrays

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

void main(void)
{
        int A [5] ;
        A[0] = 1;
        A[1] = 3;
        A[2] = 4;
        A[3] = 7;
        A[4] = 89;
        for (int i = 0 ; i<=4 ; i ++)
        {
                cout << "A[" << i << "] = " << A[i] << endl ;
        }
}
```

```
A[0] = 1
A[1] = 3
A[2] = 4
A[3] = 7
A[4] = 89
Press any key to continue
```

Arrays

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

void main(void)
{
        int A [5] ;
        A[0] = 1;
        A[1] = 3;
        A[2] = 4;
        A[3] = 7;
        A[4] = 89;
        A[5] = 34;
        for (int i = 0 ; i<=4 ; i ++)
        {
                cout << "A[" << i << "] = " << A[i] << endl ;
        }
}
```

```
A[0] = 1
A[1] = 3
A[2] = 4
A[3] = 7
A[4] = 89
Press any key to continue
```

Arrays

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

void main(void)
{
        int A [5];
        A[3] = 0 ;
        A[0] = -1 ;
        for (int i = 0 ; i<=10 ; i ++)
        {
                cout << "A[" << i << "] = " << A[i] << endl ;
        }
}
```

Compiler error ?

Arrays

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


void main(void)
{
        int A [5] ;
        A[3] = 0 ;
        A[0] = 34 ;
        A[3] = 2 ;
        A[2] = 1 ;
        A[3+1] = 5;
        A[1] = 7 ;
        A[4] = 6 ;
        for (int i = 0 ; i<=4 ; i ++)
        {
                cout << "A[" << i << "] = " << A[i] << endl ;
        }
}
```

```
A[0] = 34
A[1] = 7
A[2] = 1
A[3] = 2
A[4] = 6
Press any key to continue
```

Arrays

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


void main(void)
{
        int A [5] ;
        A[3] = 0 ;
        A[0] = 34 ;
        A[3] = 2 ;
        A[2] = 1 ;
        A[3+1] = 5;
        A[1] = 7 ;
        A[4] = 6 ;
        for (int i = -1 ; i<=3 ; i ++)
        {
                cout << "A[" << i+1 << "] = " << A[i+1] << endl ;
        }
}
```

```
A[0] = 34
A[1] = 7
A[2] = 1
A[3] = 2
A[4] = 6
Press any key to continue
```

Arrays

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


void main(void)
{
        int A [5] ;
        A[3] = 0 ;
        A[0] = 34 ;
        A[3] = 2 ;
        A[2] = 1 ;
        A[3+1] = 5;
        A[1] = 7 ;
        A[4] = 6 ;
        for (int i = -1 ; i<=3 ; i ++)
        {
                cout << "A[" << i << "] = " << A[i] << endl ;
        }
}
```

```
A[-1] = 1637801184
A[0] = 34
A[1] = 7
A[2] = 1
A[3] = 2
Press any key to continue
```

Arrays

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


void main(void)
{
        int A [5] ;
        A[3] = 0 ;
        A[0] = 34 ;
        A[3] = 2 ;
        A[2] = 1 ;
        A[3+1] = 5;
        A[1] = 7 ;
        A[4] = 6 ;
        for (int i = -1 ; i<4 ; i ++)
        {
                cout << "A[" << i << "] = " << A[i] << endl ;
        }
}
```

```
A[-1] = 1804120084
A[0] = 34
A[1] = 7
A[2] = 1
A[3] = 2
Press any key to continue
```

Arrays

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


void main(void)
{
        int A      [5]  , X[2];
        A[3] = 0 ;
        A[0] = -1 ;
        X[0] = A[3] ;
        X[1] = A[2] ;
        for (int i = 0 ; i<2 ; i ++)
        {
                cout << "X[" << i << "] = " << X[i] << endl ;
        }
}
```

```
X[0] = 0
X[1] = -858993460
Press any key to continue
```

Arrays

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


void main(void)
{
        int a[5] = {3,5,2,4,33} ;// initializing
        for (int i = 0 ; i<5 ; i ++)
        {
                cout << "a[" << i << "] = " << a[i] << endl ;
        }
}
```

```
a[0] = 3
a[1] = 5
a[2] = 2
a[3] = 4
a[4] = 33
Press any key to continue
```

Arrays

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


void main(void)
{
        int a[5] = {3,5,2,4,33,12} ;        // initializing
        for (int i = 0 ; i<5 ; i ++)
        {
                cout << "a[" << i << "] = " << a[i] << endl ;
        }
}
```

```
Compiler error
```

Arrays

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


void main(void)
{
        int a[5] = {3,5,2} ;        // initializing
        for (int i = 0 ; i<5 ; i ++)
        {
                cout << "a[" << i << "] = " << a[i] << endl ;
        }
}
```

```
a[0] =3
a[1] =5
a[2] =2
a[3] =0
a[4] =0
Press any key to continue
```

Arrays

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


void main(void)
{
        int a[5] = {4} ; // initializing
        for (int i = 0 ; i<5 ; i ++)
        {
                cout << "a[" << i << "] = " << a[i] << endl ;
        }
}
```

```
a[0] = 4
a[1] = 0
a[2] = 0
a[3] = 0
a[4] = 0
Press any key to continue
```

Arrays

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


void main(void)
{
        int a[5] = {0} ; // initializing
        for (int i = 0 ; i<5 ; i ++)
        {
                cout << "a[" << i << "] = " << a[i] << endl ;
        }
}
```

```
a[0] = 0
a[1] = 0
a[2] = 0
a[3] = 0
a[4] = 0
Press any key to continue
```

Arrays

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


void main(void)
{
        int a[] = {32,34,12,23} ;
        for (int i = 0 ; i<4 ; i ++)
        {
                cout << "a[" << i << "] = " << a[i] << endl ;
        }
}
```

```
Compiler error !
```

```
int a[] = {32,34,12,23} ; // means an array with 4 elements
```

```
a[0] = 32
a[1] = 34
a[2] = 12
a[3] = 23
Press any key to continue
```

Arrays

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


void main(void)
{
        int a[] = {32,34,12,23} ;
        for (int i = 0 ; i<5 ; i ++)
        {
                cout << "a[" << i << "] = " << a[i] << endl ;
        }
}
```

```
a[0] = 32
a[1] = 34
a[2] = 12
a[3] = 23
a[4] = 1364078976
Press any key to continue . . .
```

Arrays

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

void main(void)
{
        int a[5];
        for (int i = 0 ; i<5 ; i ++)
        {
                a[i] = i+2 ;
        }
        cout << "_____" << endl ;
        cout << "The outputted array is : " << endl;
        for (int i=0 ; i<5 ; i++)
        {
                cout << a[i] << endl ;
        }
}
```

```
_____
The outputted array is :
2
3
4
5
6
Press any key to continue
```

Arrays

```
#include <iostream>
using namespace::std;
const int ArrSize = 4 ;

void main(void)
{
        int a[ArrSize];
        for(int i = 0 ; i<ArrSize ; i++)
        {
                a[i]=i*ArrSize ;
        }
        cout << a[3] ;
}
```

```
12
```

Arrays

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


void main(void)
{
        float a[4] = (30);
        cout << a[0] << endl;
}
```

```
30
```

Arrays

```
#include <iostream>
using namespace::std;
const int GoGo = 0 ;

void main(void)
{
        float a[GoGo] = (30);
        cout << a[0] << endl;
}
```

```
Compiler error : could allocate any array of size Zero
```

Arrays

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


void main(void)
{
        float a[4] = {30. , 32 , 33.4 , 23.0 };
        cout << a[0] << endl;
}
```

```
30
```

```
Run with warning of conversion from double to int
```

Arrays

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


void main(void)
{
        int x = 44. ;
}
```

```
Compile : Run
```

But with warning for converting from double to int ...
Possible loss of data

... then (You should know why) no ...

Arrays

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


void main(void)
{
        float a[4] = {30. , 32 , 33.4 , 23.0 } ;
        int x[2] = {34,0};
        a[x[1]]=3;
        for (int i = 0 ; i < 4 ; i++)
                cout << a[i] << endl;
}
```

```
3
32
3.4
2
Press any key to continue
```

Arrays

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


void main(void)
{
        float a[4] = {30. , 32 , 33.4 , 23.0 } ;
        float x[2] = {34,0};
        a[x[1]]=3;
        for (int i = 0 ; i < 4 ; i++)
                cout << a[i] << endl;
}
```

```
Compiler error ... Index is not of an integral type
```

Tweaked Control Structure I - #1

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

void main(void)
{
        int x = 5 ;
        while (x != 10)
        {
                cout << x*x << endl ;
        )
}
```

```
25
36
49
64
81
100


Press any key to continue
```

Tweaked Control Structure I - #2

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

void main(void)
{
        int i = 0 ;
        int c = 5 ;
        do
        (
                ++c ;
                cout << "beep ! , number = " << i << endl ;
                i++;
        } while ( c++ <= 10 ) ;
)
```

```
beep ! , number = 0
beep ! , number = 1
beep ! , number = 2
beep ! , number = 3
Press any key to continue
```

Tweaked Control Structure I #3

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

void main(void)
{
        int i = 66 ;
        int c = 5 ;
        int j = 2 ;
        for (int i = 0 , j ; i<5 ; i++ , j-- )
        {
                cout << ++j << endl ;
        }
)
```

```
1
0
-1
-2
-3
```

Tweaked Control Structure I - #4

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

void main(void)
{
        for (int i = 0  ; i<5 ; i++  )
        {
                cout << ++i << endl ;
                if (i = 34)
                {
                        break ;
                }
        )
)
```

```
1
```

Code Cracking - 05

```
#include <iostream>
using namespace::std;
int x = 0  ;

float f2 (int z )
{
        cout << z << endl ;
        z*=2 ;
        return z ;
}

void f1 (  int &x )
{
        x+2 ;
        cout << x << endl ;
        f2(x) ;
        cout << x << endl ;
}


void main(void)
{
        int x = 4 ;
        cout << ::x++ << endl ;
        f1(x) ;
        f1(::x) ;
        cout << ++::x << endl ;
}
```

Code Cracking - 21

```
#include <iostream>
using namespace::std;
int s = 0 ;

int Sum(int n)
{
        if (n == 1 )
                return 1 ;
        else
        {
                ::s  = ::s +  n + Sum(n-1);
        }
}

void main(void)
{
        cout << Sum(4) << endl ;
        cout << ::s << endl ;
}
```

Strings!

Strings using Arrays

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

void main(void)
{
    string str = "I AM A STRING!";
    system("pause");
}
```

Strings using Arrays

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

void main(void)
{
    string str = "I AM A STRING!";
    cout << str << endl;
    system("pause");
}
```

```
I AM A STRING!
Press any key to continue
```

Strings using Arrays

- String :
  - It's an ***array*** of ***characters***
- All strings end with `null ('\0')`

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


void main(void)
{
        char str1 [] = "Hello" ;
        // null character implicitly added at the end
        char str2 [] = {'H','e','l','l','o','\0'};
        // null character explicitly added
}
```

Strings using Arrays

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


void main(void)
{
char str1 [] = "hello" ;
char str2 [] = {'h','e','l','l','o','\0'};

        for (int i = 0 ; i < 6 ; i++)
        {
                cout << int(str1[i])
                << endl;
        }

        cout << "----------" << endl ;

        for (int i = 0 ; i < 6 ; i++)
        {
                cout << int(str2[i])
                << endl;
        }

}
```

```
104
101
108
108
111
0
----------
104
101
108
108
111
0
Press any key to continue
```

Strings using Arrays

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


void main(void)
{
        char str1 [] = "hello" ;
        char str2 [] = {'h','e','l','l','o','\0'};
        (str1[0]==str2[0]) ? cout << "Yuppy !" : cout << "No No :(" ;
}
```

```
Yuppy !
```

Strings using Arrays

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


void main(void)
{
        char str1 [] = "hello" ;
        char str2 [] = {'h','e','l','l','o','\0'};
        (str1==str2) ? cout << "Yuppy !" : cout << "No No :(" ;
}
```

```
No No :(
```

Strings using Arrays

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


void main(void)
{
        char str1 [] = "hello" ;
        char str2 [] = {'h','e','l','l','o','\0'};
        (str1==str2) ? cout << "Yuppy !" : cout << "No No :(" ;
}
```

```
No No :(
```

Why it's a no?!

Strings using Arrays

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


void main(void)
{
        char str1 [] = "hello" ;
        char str2 [] = {'h','e','l','l','o','\0'};
        (str1==str2) ? cout << "Yuppy !" : cout << "No No :(" ;
}
```

```
No No :(
```

Why it's a no?!

Cos we are comparing addresses and not values!

Strings using Arrays

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


void main(void)
{
        char str1 [] = "hello" ;
        char str2 [] = {'h','e','l','l','o','\0'};
        (str1==str2) ? cout << "Yuppy !" : cout << "No No :(" ;
        cout << "\n" ;

        for (int i = 0 ; i < 6 ; i++)
        {
                cout << int(str1[i]) << endl;
        }

        cout << "------------" << endl ;

        for (int i = 0 ; i < 6 ; i++)
        {
                cout << int(str2[i]) << endl;
        }
}
```

```
No No :(
104
101
108
108
111
0
------------
104
101
108
108
111
0
```

Strings using Arrays

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


void main(void)
{
        char str1 [] = "hello" ;
        char str2 [] = {'h','e','l','l','o','\0'};
        (str1==str2) ? cout << "Yuppy !" : cout << "No No :(" ;
        cout << "\n" ;

        for (int i = 0 ; i < 6 ; i++)
        {
                cout << str1[i] ;
        }

        cout << "\n------------" << endl

        for (int i = 0 ; i < 6 ; i++)
        {
                cout <<str2[i] ;
        }
        cout << endl ;

}
```

```
No No :(
Hell
------------
Hell
Press any key to
continue
```

Strings using Arrays

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


void main(void)
{
        char str1 [10] ;
        cin >> str1 ;

        cout << "The inputed string as a whole ! , is " << str1 << endl ;


}
```

```
ZeZeZo
The inputed string as a whole ! , is ZeZeZo
Press any key to continue
```

Strings using Arrays

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


void main(void)
{
        char str1 [10] ;
        cin >> str1 ;

        cout << "The inputed string as a whole ! , is " << str1 << endl ;


}
```

```
Geeeeeeeeeeeeeeeeeeeeeeeeeee
The inputed string as a whole ! , is Ceeeeeeeeeeeeeeeeeeeeeeeeeeee
Press any key to continue . . .
```

Compiler and prime but with Run-time error !!! Out of bound !

Strings using Arrays

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


void main(void)
{
        char str1 [10] ;
        cin >> str1 ;

        cout << "The inputed string as a whole ! , is " << str1 << endl ;


}
```

```
WaWe Ed
The inputed string as a whole ! , is WaWe
Press any key to continue . . .
```

Stops at first white space character

Strings using Arrays

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


void main(void)
{
        char strl [10] ;
        cin >> strl ;

        cout << "The inputed string as a whole ! , is " << strl << endl ;


}
```

Printing the string "Array" like this is just for the strings
No other type of arrays can be printed like that

Strings using Arrays

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

void main(void)
{
    int intArr [10] = {0,1,1,12,34};

    cout << "The inputed string as a whole ! , is " << intArr << endl ;
    system("pause");
}
```

```
The inputed string as a whole ! , is 0043F958
Press any key to continue . . .
```

When printing the array it print it's location on memory by default

Strings using Arrays

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

void main(void)
{
    int intArr [10] = {0,1,1,12,34};

    cout << "The inputed string as a whole ! , is " << intArr << endl ;
    system("pause");
}
```

```
The inputed string as a whole ! , is 0047F850
Press any key to continue . . .
```

When printing the array it print it's location on memory by default

Just the char array (char charArr [])
Don't do that!
It prints the char contained in it!

Strings using Arrays

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

void main(void)
{
    char charArr [10] = "Hello!";
    cout << "The inputed string as a whole ! , is "
    << charArr << endl ;
    system("pause");
}
```

```
The inputed string as a whole ! , is Hello!
Press any key to continue . . .
```

When printing the array it print it's location on memory by default

Just the char array (char charArr [])
Don't do that!
It prints the char contained in it!

Strings using Arrays

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


void main(void)
{
    int str1 [10] ;
    cin >> str1 ;

    cout << "The inputed string as a whole ! , is " << str1 << endl ;
    system("pause");
}
```

Compiler error **(cin >> str1 ;)** ***, it's not a char array!***
U can't input it as a whole!

Strings using Arrays

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

void main(void)
{
    int intArr [10] = {0,1,1,12,34};

    cout << "The inputed string as a whole ! , is " << intArr << endl ;
    system("pause");
}
```

```
The inputed string as a whole ! , is 0043F998
Press any key to continue . . .
```

Strings using Arrays – static I

- Let's see this first …

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


void AutoArr()
{
        int Arr1 [3] = {1,2,3} ;
        cout << "Auto Arr" << endl ;
        for (int i=0 ; i<3 ; i++)
        {
                cout << "Arr1[" << i << "]=" <<
Arr1[i] << endl;
        }

}


void main(void)
{
        AutoArr() ;
}
```

```
Auto Arr
Arr1[0]=1
Arr1[1]=2
Arr1[2]=3
```

Strings using Arrays – static I

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

void AutoArr()
{
        int Arr1 [3] = {1,2,3} ;
        cout << "Auto Arr" << endl ;
        for (int i=0 ; i<3 ; i++)
        {
                cout << "Arr1[" << i << "]=" << Arr1[i] << endl;
        }

}


void main(void)
{
        AutoArr ;
}
```

Strings using Arrays – static I

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


void AutoArr()
{
int Arrl [3] = {1,2,3} ;
cout << "Auto Arr : " << endl ;
for (int i=0 ; i<3 ; i++)
{
        cout << "Arrl[" << i << "]="
        << Arrl[i]<< endl;
}
cout << "Auto Arr after +2 to all elemets :"
        <<endl;
for (int i=0 ; i<3 ; i++)
{
        cout << "Arrl[" << i << "]=" <<
        (Arrl[i]=Arrl[i]+2) << endl;
}

}

void main(void)
{
        AutoArr() ;
}
```

```
Auto Arr :
Arrl[0]=1
Arrl[1]=2
Arrl[2]=3
Auto Arr after +2 to all elemets :
Arrl[0]=3
Arrl[1]=4
Arrl[2]=5
Press any key to continue
```

Strings using Arrays – static I

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


void AutoArr()
{
int Arrl [3] = {1,2,3} ;
cout << "Auto Arr : " << endl ;
for (int i=0 ; i<3 ; i++)
{
        cout << "Arrl[" << i << "]="
        << Arrl[i]<< endl;
}
cout << "Auto Arr after +2 to all elemets :"
        <<endl;
for (int i=0 ; i<3 ; i++)
{
        cout << "Arrl[" << i << "]=" <<
        (Arrl[i]+=2) << endl;
}

}

void main(void)
{
        AutoArr() ;
}
```

```
Auto Arr :
Arrl[0]=1
Arrl[1]=2
Arrl[2]=3
Auto Arr after +2 to all elemets :
Arrl[0]=3
Arrl[1]=4
Arrl[2]=5
Press any key to continue
```

Strings using Arrays – static I

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


void AutoArr()
{
int Arrl [3] = {1,2,3} ;
cout << "Auto Arr : " << endl ;
for (int i=0 ; i<3 ; i++)
{
        cout << "Arrl[" << i << "]="
        << Arrl[i]<< endl;
}
cout << "Auto Arr after +2 to all elemets :"
        <<endl;
for (int i=0 ; i<3 ; i++)
{
        cout << "Arrl[" << i << "]:" <<
        (Arrl[i]+=2) << endl;
}

}

void main(void)
{
        AutoArr() ;
        AutoArr() ;
}
```

```
Auto Arr :
Arrl[0]=1
Arrl[1]=2
Arrl[2]=3
Auto Arr after +2 to all elemets :
Arrl[0]=3
Arrl[1]=4
Arrl[2]=5
Auto Arr :
Arrl[0]=1
Arrl[1]=2
Arrl[2]=3
Auto Arr after +2 to all elemets :
Arrl[0]=3
Arrl[1]=4
Arrl[2]=5
Press any key to continue
```

Strings using Arrays – static I

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


void AutoArr()
{
int Arrl [3] = {1,2,3} ;
cout << "Auto Arr : " << endl ;
for (int i=0 ; i<3 ; i++)
{
        cout << "Arrl[" << i << "]="
        << Arrl[i]<< endl;
}
cout << "Auto Arr after +2 to all elemets :"
        <<endl;
for (int i=0 ; i<3 ; i++)
{
        cout << "Arrl[" << i << "]=" <<
        (Arrl[i]=+2) << endl;
}

}

void main(void)
{
        AutoArr() ;
}
```

```
Auto Arr :
Arrl[0]=1
Arrl[1]=2
Arrl[2]=3
Auto Arr after +2 to all elemets :
Arrl[0]=3
Arrl[1]=4
Arrl[2]=5
Press any key to continue
```

Strings using Arrays – static

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


void StaticArr()
{
static int Arrl [3] = {1,2,3} ;
cout << "Arr : " << endl ;
for (int i=0 ; i<3 ; i++)
        {
cout << "Arrl[" << i << "]=" << Arrl[i] << endl;
        }
cout << "Arr after +2 to all elemets :" <<endl;
for (int i=0 ; i<3 ; i++)
        {
cout << "Arrl[" << i << "]=" << (Arrl[i]+2) <<endl;
        }
}

void main(void)
{
        StaticArr() ;
        StaticArr();
}
```

```
Arr :
Arrl[0]=1
Arrl[1]=2
Arrl[2]=3
Arr after +2 to all elemets :
Arrl[0]=3
Arrl[1]=4
Arrl[2]=5
Arr :
Arrl[0]=3
Arrl[1]=4
Arrl[2]=5
Arr after +2 to all elemets :
Arrl[0]=5
Arrl[1]=6
Arrl[2]=7
Press any key to continue . . .
```

Strings using Arrays – static

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


void StaticArr()
{
static int Arrl [3] = {1,2,3} ;
cout << "Arr : " << endl ;
for (int i=0 ; i<3 ; i++)
        {
cout << "Arrl[" << i << "]=" << Arrl[i] << endl;
        }
cout << "Arr after +2 to all elemets :" <<endl;
for (int i=0 ; i<3 ; i++)
        {
cout << "Arrl[" << i << "]=" << (Arrl[i]+2) <<endl;
        }
}

void main(void)
{
        StaticArr() ;
        StaticArr();
}
```

```
Arr :
Arrl[0]=1
Arrl[1]=2
Arrl[2]=3
Arr after +2 to all elemets :
Arrl[0]=3
Arrl[1]=4
Arrl[2]=5
Arr :
Arrl[0]=3
Arrl[1]=4
Arrl[2]=5
Arr after +2 to all elemets :
Arrl[0]=5
Arrl[1]=6
Arrl[2]=7
Press any key to continue . . .
```

Arrays – passing to functions

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

void PassingArr(int a [])
{
        cout << a[0] << endl;
}

void main(void)
{
        int Arr [] = {3,2};
        PassingArr(Arr);
}
```

```
3
```

Arrays

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

void PassingArr(int a [])
{
        cout << a[0] << endl;
}

void main(void)
{
        int Arr [] = {3,2};
        PassingArr(Arr[]);
}
```

```
Compiler error ... Passing the array with subscript []
```

Arrays

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

void PassingArrVal(int a[])
{
        a[0]+=2 ;
}


void main(void)
{
        int Arr [] = {3,2};
        PassingArrVal(Arr);
        cout << Arr[0] << endl ;
        PassingArrVal(Arr);
        cout << Arr[0] << endl ;
}
```

Passing arrays by value is passing it with reference not by value! ....

Arrays

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

void PassingArrVal(int a[])
{
        a[0]+=2 ;
}
void PassingArrRef1(int &a[])
{
        a[0]+=2 ;
}
void main(void)
{
        int Arr [] = {3,2};
        PassingArrVal(Arr);
        cout << Arr[0] << endl ;
        PassingArrRef(Arr);
        cout << Arr[0] << endl ;
}
```

Compiler error .. can't pass arrays with values, it's referenced all by it self

Arrays

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

void PassingArr(const int a [])
{
        cout << a[0] << endl;
}
void main(void)
{
        int Arr[2] = {1,3} ;
        PassingArr(Arr);
}
```

Arrays

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

void PassingArr(const int a [])
{
        a[0]+=2 ;
}

void main(void)
{
        int Arr [2] ;
        PassingArr(Arr);
        cout << Arr[0] << endl ;
}
```

Compiler error .. can't modify const arrays

Multiple subscripted Arrays

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

void main(void)
{
        int Arr[3][4] ;  // 3 rows, 4 columns
}
```

Multiple subscripted Arrays

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

void main(void)
{
        int Arr[2,3] ;
}
```

```
Compiler error .. Should be [][]
```

Multiple subscripted Arrays

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

void main(void)
{
        int Arr [][] ;
}
```

```
Compiler error : missing subscript size
```

Multiple subscripted Arrays

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

void main(void)
{
        int Arr [][3] ;
}
```

```
Compiler error : missing subscript size
```

Multiple subscripted Arrays

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

void main(void)
{
        int Arr[2][3] = { {2,3,5} , {4,1,3} } ;
                        //      row#1     row#2
}
```

2	3	5
4	1	3

Multiple subscripted Arrays

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

void main(void)
{
        int Arr[2][3] = { {2,3} , {4,1,3} } ;
                        //      row#1   row#2
}
```

2	3	0
4	1	3

Multiple subscripted Arrays

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

void main(void)
{
        int Arr[2][3] = { {2} , {4,3} } ;
}
```

2	0	0
4	3	0

Multiple subscripted Arrays

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

void main(void)
{
        int Arr[2][3] = { {2,3,5} , {4,3} } ;
        cout << Arr[2][3] << endl ;
}
```

[illegible]

Not a compiler error ... but the number in that location in memory

Multiple subscripted Arrays

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

void main(void)
{
        int Arr[2][3] = { {2,3,5} , {4,3} } ;
        cout << Arr[1][2] << endl ;
}
```

```
4
```

Multiple subscripted Arrays passing to function

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

void ArrFun(int A [2][3])
{
        cout << A [1][1] ;
}


void main(void)
{
        int Arr[2][3] = { {2,3,5} , {4,3} } ;
        ArrFun(Arr);
}
```

```
3
```

Multiple subscripted Arrays passing to function

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

void ArrFun(int A [2][3])
{
        cout << A [1][1] ;
}


void main(void)
{
        int Arr[2][3] = { {2,3,5} , {4,3} } ;
        ArrFun(Arr[2][3]);
}
```

```
Compiler error ... 2||...
```

Multiple subscripted Arrays passing to function

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

void ArrFun(int A [][3])
{
        cout << A [1][1] ;
}


void main(void)
{
        int Arr[2][3] = { {2,3,5} , {4,3} } ;
        ArrFun(Arr);
}
```

```
3
```

No compiler error for missing the first subscript in the function's prototype ...
like single-scripted arrays

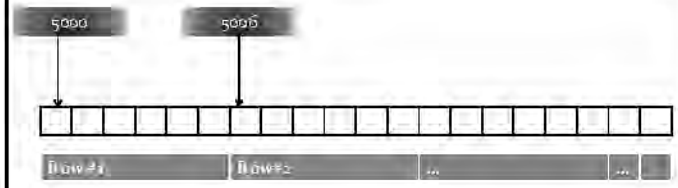
Multiple subscripted Arrays passing to function

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

void ArrFun(int A [][])
{
        cout << A [1][1] ;
}


void main(void)
{
        int Arr[2][3] = { {2,3,5} , {4,3} } ;
        ArrFun(Arr);
}
```

Compiler error ... missing the second subscript in the function's prototype

Multiple subscripted Arrays Memory storage

```
#include <iostream>
using namespace::std;
const int Rows = 2 ;
const int Columns = 3 ;

void main(void)
{
        int Arr[Rows][Columns] ;
}
```



Functions I

Functions

- Why function ?
- functions \ procedures with c++
- What's the function prototype (declaration) \ definition ?
- Function signature

```
int ZeZe(int) ;                 // function prototype
```

  - It's *just* the name of the function with its parameters
    - No return type !!!! (*Return type is not considered as part of the function's signature! , guess why? :)*)

```
ZeZe(int)                       // function signature
```

Functions

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

int ZeZe(int) ;                          // function prototype


int  main()
{

}
```

Functions

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

int ZeZe(int)                  // function prototype


int  main()
{
        return 0 ;
}
```

```
Compiler error - missing ; after function's declaration
```

Functions

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

ZeZe(int);                       // function prototype


int  main()
{
        return 0 ;
}
```

```
Compiler error - missing return type
```

Functions

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

int ZeZe(int);                              / function prototype


int  main()
{

}
```

Functions

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

int Mult( int x )                    // note there's no ; when we
                                     // write the definition here
{
        x = x * 2 ;
        return x ;
}

void main()
{
}
```

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

int Mult( int x ) ;

void main()
{}

int Mult( int x )
{
        x = x * 2 ;
        return x ;
}
```

The same

Functions

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

int Mult( int )
{
        x = x * 2 ;
        return x ;
}

void main()
{
}
```

```
Compiler Error : missing x
```

Functions

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

int Mult( int x )
{
        x = x * 2 ;
        return x ;
}

void main()
{
}
```

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

int Mult(int);               // we didn't write the x
                             // parameter when prototype only
void main()
{}

int Mult( int x )
{
        x = x * 2 ;
        return x ;
 }
```

The same

Functions

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

int Mult(int);

void main()
{
        Mult (3) ;
}

int Mult( int x )
{
        x = x * 2 ;
        return x ;
}
```

Functions

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

int Mult(int);

void main()
{
        cout << Mult(3) ;
}

int Mult( int x )
{
        x = x * 2 ;
        return x ;
}
```

Functions

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

int Mult(int);

void main()
{
        Mult(3) ;
}

int Mult( int x )
{
        x = x * 2 ;
        cout << x ;
        return 0 ;
}
```

Functions

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

int Mult(int);

void main()
{
        int c = 3 ;
        Mult(c) ;
}

int Mult( int x )
{
        x = x * 2 ;
        cout << x ;
        return 0 ;
}
```

Functions

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

int Mult(int);

void main()
{
        int x = 3 ;
        Mult(x) ;
}

int Mult( int x )
{
        x = x * 2 ;
        cout << x ;
        return 0 ;
}
```

Functions

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

int Mult(int);

void main()
{
        int x = 3 ;
        Mult(x) ;
        cout << x ;
}

int Mult( int x )
{
        x = x * 2 ;
        cout << x ;
        return 0 ;
}
```

```
6
3
```

Functions

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

int Mult(int);

void main()
{
        int x = 3 ;
        Mult(x) ;
        cout << endl ;
        cout << x ;
}

int Mult( int x )
{
        x = x * 2 ;
        cout << x ;
        return 0 ;
}
```

```
6
3
```

Functions

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

int Mult(int);

void main()
{
        int x = 3 ;
        Mult(x) ;
        cout << x ;
}

int Mult( int x )
{
        x = x * 2 ;
        cout << x << endl ;
        return 0 ;
}
```

```
6
3
```

Functions

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

int Mult(int);

void main()
{
        int x = 3 ;
        cout << Mult(x) << endl ;
        cout << x ;
}

int Mult( int x )
{
        x = x * 2 ;
        return x ;
}
```

```
6
3
```

Functions

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

void Mult(void);

void main()
{
        int x = 3;
        Mult() << endl ;
        cout << x ;
}

void Mult( void )
{
        int x = 3 ;
        cout << x * 2 << endl ;
}
```

Compiler error

Functions

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

void Mult(void);

void main()
{
        int c = 3;
        Mult() << endl ;
        cout << c ;
}

void Mult( void )
{
        int x = 3 ;
        cout << x * 2 << endl ;
}
```

Compiler error

Functions

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

void Mult(void);

void main()
{
        cout << x ;
        Mult() << endl ;
}

void Mult( void )
{
        int x = 3 ;
        cout << x*2 << endl ;
}
```

Compiler error : x is undeclared

Functions

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

void Mult(void);

void main()
{
        Mult() << endl ;
}

void Mult( void )
{
        int x = 3 ;
        x = x * 2 ;
        return 0 ;
}
```

Compiler error : cannot convert parameter 1 from 'void' to 'int'

Functions

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

void Mult(void);

void main()
{
        Mult() << endl ;
}

void Mult( void )
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error : [illegible] 'endl' [illegible] 'void' [illegible]

Functions

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

void Mult(void);

void main()
{
        cout << Mult() << endl ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error : [illegible] function

Functions

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

void Mult(void);

void main(void)
{
        Mult() ;
}

void Mult(void)
{
        int x = 3 ;
        x = x * 2 ;
}
```

At last ... everything's working [illegible]

Functions

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

void Mult(void);

void main()
{
        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

At last ... everything's working [illegible]

In the parameter list ... void or blank are the same

Functions

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

Mult(void);

void main()
{
        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error : missing return type in function's prototype

Functions

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

void Mult(void);

void main()
{
        Mult() ;
}

Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error : missing return type

Functions

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

Mult(void);

void main()
{
        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error : missing return type

Functions

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

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}

void main()
{
        Mult() ;
}
```

Functions

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

void main()
{
        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error … missing function's prototype

Functions

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

void main()
{
        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler Parsing

```
#include <iostream>
using namespace::std;
void main()
{
        Mult() ;
}
void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error … missing function's prototype

Functions

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

void main()
{
        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler Parsing

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

{
        Mult() ;
}
void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error … missing function's prototype

Functions

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

void main()
{
        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler Parsing

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

{       // Enter new scope
        Mult() ;
}
void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error … missing function's prototype

Functions

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

void main()
{
        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler Parsing

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

        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error … missing function's prototype

Functions

Compiler Parsing

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

  Mult() ; // parsing
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler Parsing

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

        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Functions

Compiler Parsing

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

  Mult() ; // parsing
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler Parsing

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

        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler search for a matched function

Functions

Compiler Parsing

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

  Mult() ; // parsing
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

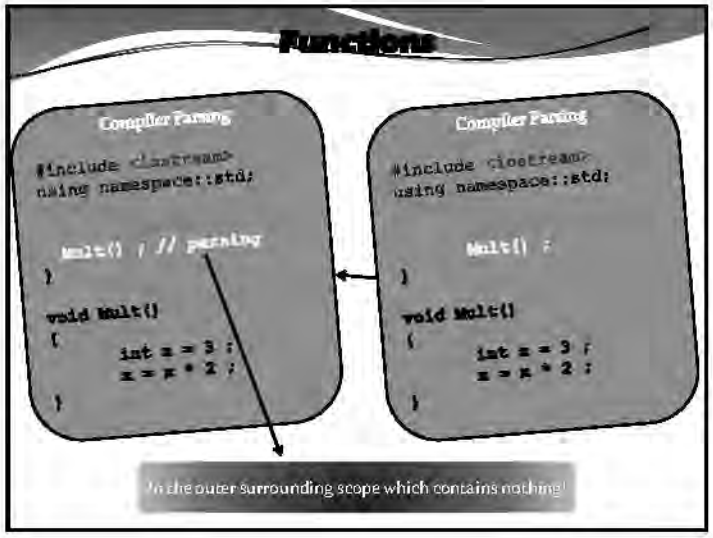
Compiler Parsing

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

        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler search for a matched function

Where?

Functions



Functions

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

void Mult(void);

void main(void)
{
        Mult() ;
}

int Mult(void)
{
        int x = 3 ;
        x = x * 2 ;
        return x ;
}
```

Compiler error ..return type differs

Functions

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

int Mult(void);

void main(void)
{
        Mult() ;
}

void Mult(void)
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error .. return type differs

Functions

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

int Mult(void);

void main(void)
{
        cout << Mult() << endl ;
}

int Mult(void)
{
        int x = 3 ;
        x = x * 2 ;
        return x ;
}
```

6

Functions

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

int Mult(void);

void main(void)
{
        cout << Mult() << endl ;
}

int Mult(void)
{
        int x = 3 ;
        return x * 2 ;
}
```

6

Functions

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

int Mult(void);
void main(void)
{
        cout << Mult() << endl ;
}

int Mult(void)
{
        int x = 3 ;
        return x * 2 ;
        cout << x << end ;
}
```

6

Nothing will be printed after the return statement

Functions

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

int Mult(int x );

void main(void)
{
        cout << Mult() << endl ;
}

int Mult(int x )
{
        return x * 2 ;
}
```

Compiler error : no viable conversion from

Functions

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

int Mult(int x );

void main(void)
{
        cout << Mult(3) << endl ;
}

int Mult(int x )
{
        return x * 2 ;
}
```

6

Functions

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

int Mult(int x );

void main(void)
{
        cout << Mult(3) << endl ;
}

int Mult(int x )
{
        int x = 2 ;
        return x * 2 ;
}
```

Compiler error : redeclaration of variable x

Functions

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

int Mult(int , int );
void main(void)
{
        int i1 , i2 ;
        cout << Mult(i1,i2) << endl ;
}

int Mult(int x , int y)
{
        return x * 2 ;
}
```

0

Functions

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

int Mult(int , int );
void main(void)
{
        int i1 , i2 ;
        cout << Mult(i1,i2) << endl ;
}

int Mult(int x , y)
{
        return x * 2 ;
}
```

Compiler error : missing type for y

Functions

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

int Mult(int , int );
void main(void)
{
        int i1 , i2 ;
        cout << Mult(i1,i2) << endl ;
}

int Mult(int x , int y)
{
        if (x = 2 )
        {
                return 8 ;
        }
        else
        {
                return x * 2 ;
        }
}
```

8

Functions

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

int Mult(int , int );
void main(void)
{
        int i1 , i2 ;
        cout << Mult(i1,i2) << endl ;
}

int Mult(int x , int y)
{
        int MeMe (int z)
        {
                return 0;
        }
}
```

Compiler error: can't define a function inside another one

Functions

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

int Mult(int , int );
void main(void)
{
        float i1= 3.4 , i2 ;
        cout << Mult(i1,i2) << endl ;
}

int Mult(int x , int y)
{
        return 2*x ;
}
```

Compiler error:

No

6

Like casting

Functions

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

int Mult(int);

void main(void)
{
        for (int i = 0 ; i<=10 ; i++)
        {
                cout << Mult(i) << "-";
        }
        cout << "hehe" << endl ;
        cout<< endl ;

}

int Mult(int x )
{
        return 2*x ;
}
```

0-2-4-6-8-10-12-14-16-18-20-hehe

Press any key to continue

Functions

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

void Crazy1();
void Crazy2(int);

void main(void)
{       Crazy1();       }

void Crazy1()
{
        int x = 3 ;
        Crazy2(x) ;
        cout << x ;
}

void Crazy2(int x )
{
        x+=6 ;
        cout << x << endl ;
}
```

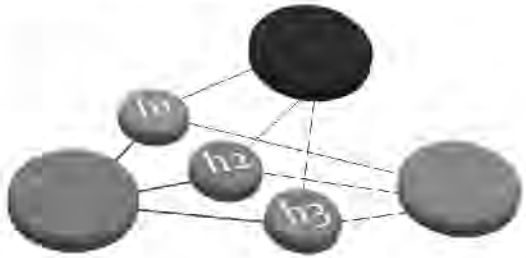
3

100 pts

Headers

Headers

- The Concept of headers



Headers File

- What's a header ?
  - Moving functions out side the main program
    - Logical groups
      - Stored in *their own* files
- How to do it ?
  - By Moving ...
    - *function declarations*
      - Into headers files
        - (. h) files // header files
    - *function definitions*
      - Into source files
        - (. cpp) files // source files
    - **Note** : function definitions ***can't*** be split up into multiple files !

Headers – Math Library

- To use it
  - Include <cmath>

```
#include <iostream>
#include <cmath>
using namespace::std;


void main(void)
{
        cout << sqrt(100.0) << endl;
}
```

```
10
```

Headers – Math Library

```
#include <iostream>

using namespace::std;


void main(void)
{
        cout << sqrt(100.0) << endl;
}
```

```
Compiler error: .. identifier not found
```

Headers – Math Library

```
#include <iostream>
#include <cmath>
using namespace::std;


void main(void)
{
        cout << sqrt(100) << endl;
}
```

```
Compiler error: "sqrt" use integer "no overloaded function"
```

Headers – Math Library

```
#include <iostream>
#include <cmath>
using namespace::std;


void main(void)
{
        cout << sqrt(sqrt(100.0)) << endl;
}
```

```
3.16228
```

Headers – Math Library

```
#include <iostream>
#include <cmath>
using namespace::std;


void main(void)
{
        int x ;
        cout << sqrt(6*x) << endl;
}
```

```
Compiler error: "sqrt" use integer "no overloaded function"
```

Headers – Math Library

```
#include <iostream>
#include <cmath>
using namespace::std;


void main(void)
{
        double x = 3  ;
        cout << sqrt(3*x) << endl;
}
```

```
:
```

Headers – Math Library

```
#include <iostream>
#include <cmath>
using namespace::std;


void main(void)
{
        float x = 3  ;
        cout << sqrt(3*x) << endl;
}
```

```
:
```

rand()

- Random number
  - `rand()`
  - Needs to include `<cstdlib>`
    - Generates *unsigned integers*
      - Between
        - 0 (Zero-D)
        - `Rand_Max` number (usually 32767)

rand()

```
#include <iostream>
#include <cstdlib>
using namespace::std;


void main(void)
{
        int x = 3  ;
        x = rand();
        cout << x << endl ;
}
```

```
:
```

```
Random Number
```

Now, *when compiling many times ...*
Every *time* we *have the same value* !

```
:
```

```
:
```

```
:
```

rand()

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

void main(void)
{
        int x = 3  ;
        x = rand();
        cout << x << endl ;
}
```

```
51
```

```
Random Number ...
```

rand()

```
#include <iostream>
#include <cstdlib>
using namespace::std;

void main(void)
{
        int x = 3  ;
        x = rand();
        cout << x << endl ;
        cout << x << endl ;
}
```

```
155
155
```

```
Random Number ...
```

rand()

```
#include <iostream>
#include <cstdlib>
using namespace::std;

void main(void)
{
        int x = 3  ;
        x = rand();
        cout << x << endl ;
        x = rand();
        cout << x << endl ;
}
```

```
155
16807
```

rand()

- `srand()`
  - Include `<cstdlib>`
  - Give a starting value for the `rand()` function
  - Usually used once in program

rand()

```
#include <iostream>
#include <ctime>
using namespace::std;

void main(void)
{
        int x = 3  ;
        srand (time(0));
        cout << x << endl ;
}
```

3

rand()

```
#include <iostream>
#include <ctime>
using namespace::std;

void main(void)
{
        int x = 3  ;
        srand (time());
        cout << x << endl ;
}
```

Compiler error ... need value for time !!

rand()

```
#include <iostream>
#include <ctime>
using namespace::std;

void main(void)
{
        int x = 3  ;
        srand (time(0));
        x = rand () ;
        cout << x << endl ;
}
```

What happened when compiling many times ?

456

12348

834

Now , when compiling many times ...
Every time we have a new different value :)

rand()

```
#include <iostream>
#include <ctime>
using namespace::std;

void main(void)
{
        int x = 3  ;
        srand (5);
        x = rand () ;
        cout << x << endl ;
}
```

What happened when compiling many times ?

48

46

48

Not an appropriate srand() paramater leads to in-suffiencient using of srand() !!!

rand()

- Scaling & shifting
  - %
    - `x % y // returns a value between 0 to y-1`
    - let's see an example ...

rand()

```
#include <iostream>
#include <cstdlib>
using namespace::std;


void main(void)
{
        int i ;
        i = rand() % 6 + 1 ;
        // rand() % 6 : number between 0 to 5
        // +1 : makes a shift of the range we have
        // 0 to 5 becomes 1 to 6
        cout << i << endl ;
        // here will print a number absolutely between 1 to 6
}
```

```
6
```

```
Or another number between 1 and 6
```

rand()

```
#include <iostream>
#include <ctime>
using namespace::std;
void main(void)
{
        srand(clock());
        // clock is from <ctime>
        // no needs to value for clock ()
        for (int i = 0 ; i < 10 ; i++)
                cout << rand() << endl ;
}
```

```
714
4415
16327
360
14057
5430
30050
16161
30477
3140
Press any key to continue . . .
```

Functions 1

Functions

- Why function ?
- functions \ procedures with c++
- What's the function prototype (declaration) \ definition ?
- Function signature

```
int ZeZe(int) ;                    // function prototype
```

  - It's *just* the name of the function with its parameters
    - No return type !!!! (*Return type is not considered as part of the function's signature! , guess why? :)*)

```
ZeZe(int)                          // function signature
```

Functions

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

int ZeZe(int) ;                          // function prototype


int  main()
{

}
```

Functions

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

int ZeZe(int)                    // function prototype


int  main()
{
        return 0 ;
}
```

```
Compiler error - missing ; after function's declaration
```

Functions

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

ZeZe(int);                       // function prototype


int  main()
{
        return 0 ;
}
```

```
Compiler error - missing return type
```

Functions

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

int ZeZe(int);                          // function prototype

int  main()
{

}
```

Functions

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

int Mult( int x )                       // note there's no ; when we
                                        // write the definition here
{
        x = x * 2 ;
        return x ;
}

void main()
{
}
```

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

int Mult( int x );

void main()
{}

int Mult( int x )
{
        x = x * 2 ;
        return x ;
}
```

The same

Functions

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

int Mult( int )
{
        x = x * 2 ;
        return x ;
}

void main()
{
}
```

```
Compiler: error ... modified ...
```

Functions

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

int Mult( int x )
{
        x = x * 2 ;
        return x ;
}

void main()
{
}
```

The same

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

int Mult(int);              // we didn't write the x
                            // primered when prototype only
void main()
{}

int Mult( int x )
{
        x = x * 2 ;
        return x ;
}
```

Functions

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

int Mult(int);

void main()
{
        Mult (3) ;
}

int Mult( int x )
{
        x = x * 2 ;
        return x ;
}
```

Functions

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

int Mult(int);

void main()
{
        cout << Mult(3) ;
}

int Mult( int x )
{
        x = x * 2 ;
        return x ;
}
```

```
6
```

Functions

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

int Mult(int);

void main()
{
        Mult(3) ;
}

int Mult( int x )
{
        x = x * 2 ;
        cout << x ;
        return 0 ;
}
```

```
6
```

Functions

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

int Mult(int);

void main()
{
        int c = 3 ;
        Mult(c) ;
}

int Mult( int x )
{
        x = x * 2 ;
        cout << x ;
        return 0 ;
}
```

```
6
```

Functions

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

int Mult(int);

void main()
{
        int x = 3 ;
        Mult(x) ;
}

int Mult( int x )
{
        x = x * 2 ;
        cout << x ;
        return 0 ;
}
```

6

Functions

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

int Mult(int);

void main()
{
        int x = 3 ;
        Mult(x) ;
        cout << x ;
}

int Mult( int x )
{
        x = x * 2 ;
        cout << x ;
        return 0 ;
}
```

6
3

Functions

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

int Mult(int);

void main()
{
        int x = 3 ;
        Mult(x) ;
        cout << endl ;
        cout << x ;
}

int Mult( int x )
{
        x = x * 2 ;
        cout << x ;
        return 0 ;
}
```

6
3

Functions

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

int Mult(int);

void main()
{
        int x = 3 ;
        Mult(x) ;
        cout << x ;
}

int Mult( int x )
{
        x = x * 2 ;
        cout << x << endl ;
        return 0 ;
}
```

6
3

Functions

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

int Mult(int);

void main()
{
        int x = 3 ;
        cout << Mult(x) << endl ;
        cout << x ;
}

int Mult( int x )
{
        x = x * 2 ;
        return x ;
}
```

```
6
3
```

Functions

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

void Mult(void);

void main()
{
        int x = 3;
        Mult() << endl ;
        cout << x ;
}

void Mult( void )
{
        int x = 3 ;
        cout << x * 2 << endl ;
}
```

```
Compiler error
```

Functions

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

void Mult(void);

void main()
{
        int c = 3;
        Mult() << endl ;
        cout << c ;
}

void Mult( void )
{
        int x = 3 ;
        cout << x * 2 << endl ;
}
```

```
Compiler error
```

Functions

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

void Mult(void);

void main()
{
        cout << x ;
        Mult() << endl ;
}

void Mult( void )
{
        int x = 3 ;
        cout << x*2 << endl ;
}
```

```
Compiler error  x is undeclared
```

Functions

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

void Mult(void);

void main()
{
        Mult() << endl ;
}

void Mult( void )
{
        int x = 3 ;
        x = x * 2 ;
        return 0 ;
}
```

Compiler error: 'return' with void function

Functions

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

void Mult(void);

void main()
{
        Mult() << endl ;
}

void Mult( void )
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error: 'endl' after 'void' Mult()

Functions

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

void Mult(void);

void main()
{
        cout << Mult() << endl ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error: 'cout' with 'void' return function

Functions

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

void Mult(void);

void main(void)
{
        Mult() ;
}

void Mult(void)
{
        int x = 3 ;
        x = x * 2 ;
}
```

No Error - everything's working :)

Functions

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

void Mult(void);

void main()
{
        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

At last … everything's working :D

In the parameter list … void or blank are the same

Functions

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

Mult(void);

void main()
{
        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error … missing return type in function's prototype

Functions

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

void Mult(void);

void main()
{
        Mult() ;
}

Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error .. missing return type

Functions

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

Mult(void);

void main()
{
        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error … missing return type

Functions

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

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}

void main()
{
        Mult() ;
}
```

Functions

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

void main()
{
        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error .. missing function's prototype

Functions

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

void main()
{
        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler Parsing

```
#include <iostream>
using namespace::std;
void main()
{
        Mult() ;
}
void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error .. missing function's prototype

Functions

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

void main()
{
        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler Parsing

```
#include <iostream>
using namespace::std;
{
        Mult() ;
}
void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error .. missing function's prototype

Functions

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

void main()
{
        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler Parsing

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

{       // Enter new scope
        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error … missing function's prototype

Functions

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

void main()
{
        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler Parsing

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

        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error … missing function's prototype

Functions

Compiler Parsing

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

  Mult() ; // parsing
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler Parsing

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

        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Functions

Compiler Parsing

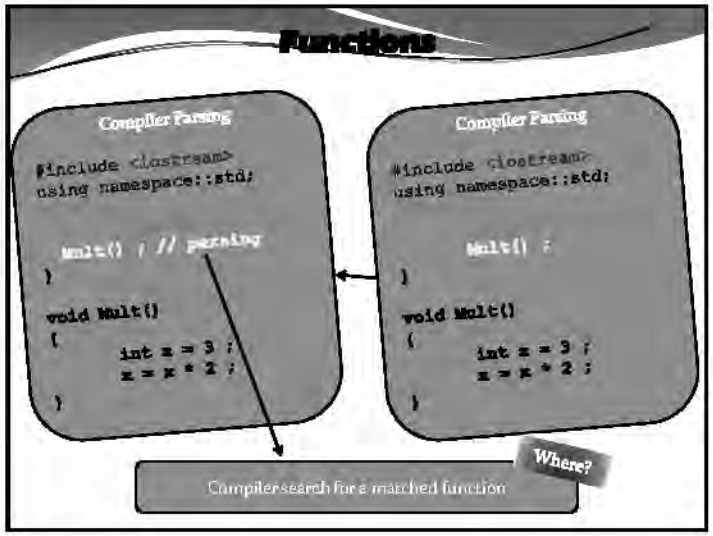
```
#include <iostream>
using namespace::std;

  Mult() ; // parsing
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler Parsing

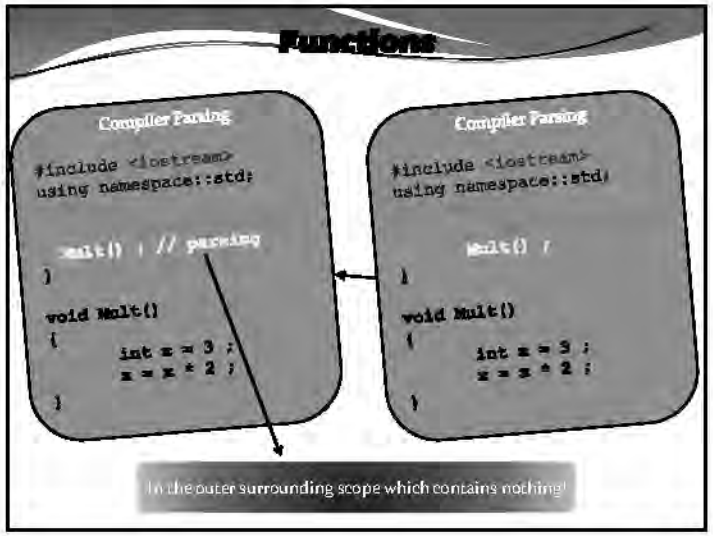
```
#include <iostream>
using namespace::std;

        Mult() ;
}

void Mult()
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler search for a matched function

Functions



Functions



Functions

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

void Mult(void);

void main(void)
{
        Mult() ;
}

int Mult(void)
{
        int x = 3 ;
        x = x * 2 ;
        return x ;
}
```

Compiler error … return type differs

Functions

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

int Mult(void);

void main(void)
{
        Mult() ;
}

void Mult(void)
{
        int x = 3 ;
        x = x * 2 ;
}
```

Compiler error … return type differs

Functions

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

int Mult(void);

void main(void)
{
        cout << Mult() << endl ;
}

int Mult(void)
{
        int x = 3 ;
        x = x * 2 ;
        return x ;
}
```

6

Functions

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

int Mult(void);

void main(void)
{
        cout << Mult() << endl ;
}

int Mult(void)
{
        int x = 3 ;
        return x * 2 ;
}
```

6

Functions

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

int Mult(void);
void main(void)
{
        cout << Mult() << endl ;
}

int Mult(void)
{
        int x = 3 ;
        return x * 2 ;
        cout << x << end ;
}
```

6

Warning will be emitted after the return statement.

Functions

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

int Mult(int x );

void main(void)
{
        cout << Mult() << endl ;
}

int Mult(int x )
{
        return x * 2 ;
}
```

Compiler error: too few arguments to function

Functions

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

int Mult(int x );

void main(void)
{
        cout << Mult(3) << endl ;
}

int Mult(int x )
{
        return x * 2 ;
}
```

```
6
```

Functions

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

int Mult(int x );

void main(void)
{
        cout << Mult(3) << endl ;
}

int Mult(int x )
{
        int x = 2 ;
        return x * 2 ;
}
```

```
Compiler error redefinition of variable
```

Functions

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

int Mult(int , int );
void main(void)
{
        int i1 , i2 ;
        cout << Mult(i1,i2) << endl ;
}

int Mult(int x , int y)
{
        return x * 2 ;
}
```

```
0
```

Functions

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

int Mult(int , int );
void main(void)
{
        int i1 , i2 ;
        cout << Mult(i1,i2) << endl ;
}

int Mult(int x , y)
{
        return x * 2 ;
}
```

```
Compiler error : missing type for y
```

Functions

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

int Mult(int , int );
void main(void)
{
        int i1 , i2 ;
        cout << Mult(i1,i2) << endl ;
}

int Mult(int x , int y)
{
        if (x = 2 )
        {
                return 3 ;
        }
        else
        {
                return x * 2 ;
        }
}
```

3

Functions

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

int Mult(int , int );
void main(void)
{
        int i1 , i2 ;
        cout << Mult(i1,i2) << endl ;
}

int Mult(int x , int y)
{
        int MeMe (int z)
        {
                return 0;
        }
}
```

Compiler error .. can't define a function inside another one

Functions

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

int Mult(int , int );
void main(void)
{
        float i1= 3.4 , i2 ;
        cout << Mult(i1,i2) << endl ;
}

int Mult(int x , int y)
{
        return 2*x ;
}
```

Compiler error ?

No ..

6

Line casting

Functions

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

int Mult(int);

void main(void)
{
        for (int i = 0 ; i<=10 ; i++)
        {
                cout << Mult(i) << "-";
        }
        cout << "hehe" << endl ;
        cout<< endl ;

}

int Mult(int x )
{
        return 2*x ;
}
```

0-2-4-6-8-10-12-14-16-18-20-hehe

Press any key to continue

Functions

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

void Crazy1();
void Crazy2(int);

void main(void)
{       Crazy1();       }

void Crazy1()
{
        int x = 3 ;
        Crazy2(x) ;
        cout << x ;
}

void Crazy2(int x )
{
        x+=6 ;
        cout << x << endl ;
}
```

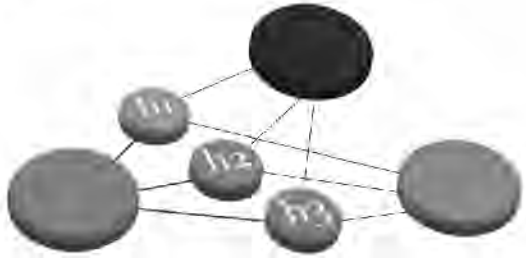
```
?
:
```

100 pt.

Headers

Headers

- The Concept of headers



Headers File

- What's a header ?
  - Moving functions out side the main program
    - Logical groups
      - Stored in *their own* files
- How to do it ?
  - By Moving ...
    - *function declarations*
      - Into headers files
        - (.h) files // header files
    - *function definitions*
      - Into source files
        - (.cpp) files // source files
      - **Note** : function definitions ***can't*** be split up into multiple files !

Headers – Math Library

- To use it
  - Include <cmath>

```
#include <iostream>
#include <cmath>
using namespace::std;


void main(void)
{
        cout << sqrt(100.0) << endl;
}
```

```
10
```

Headers – Math Library

```
#include <iostream>

using namespace::std;


void main(void)
{
        cout << sqrt(100.0) << endl;
}
```

```
Compiler error    identifier not found
```

Headers – Math Library

```
#include <iostream>
#include <cmath>
using namespace::std;


void main(void)
{
        cout << sqrt(100) << endl;
}
```

```
Compiler error    call to 'sqrt' is ambiguous
```

Headers – Math Library

```
#include <iostream>
#include <cmath>
using namespace::std;


void main(void)
{
        cout << sqrt(sqrt(100.0)) << endl;
}
```

```
3.16228
```

Headers – Math Library

```
#include <iostream>
#include <cmath>
using namespace::std;


void main(void)
{
        int x ;
        cout << sqrt(6*x) << endl;
}
```

```
Compiler error : can't use integer to overloaded function
```

Headers – Math Library

```
#include <iostream>
#include <cmath>
using namespace::std;


void main(void)
{
        double x = 3  ;
        cout << sqrt(3*x) << endl;
}
```

```
3
```

Headers – Math Library

```
#include <iostream>
#include <cmath>
using namespace::std;


void main(void)
{
        float x = 3  ;
        cout << sqrt(3*x) << endl;
}
```

```
3
```

rand()

- Random number
  - `rand()`
  - Needs to include `<cstdlib>`
    - Generates *unsigned integers*
      - Between
        - 0 (Zero :D)
        - `Rand_Max` number (usually 32767)

rand()

```
#include <iostream>
#include <cstdlib>
using namespace::std;


void main(void)
{
        int x = 3  ;
        x = rand();
        cout << x << endl ;
}
```

41

Random Number

Now , when compiling many times ...
Every time we have the same value !

41

41

41

rand()

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


void main(void)
{
        int x = 3  ;
        x = rand();
        cout << x << endl ;
}
```

41

Works <cstdlib> <cstdlib>

rand()

```
#include <iostream>
#include <cstdlib>
using namespace::std;


void main(void)
{
        int x = 3  ;
        x = rand();
        cout << x << endl ;
        cout << x << endl ;
}
```

41
41

Random Number ...

rand()

```
#include <iostream>
#include <cstdlib>
using namespace::std;


void main(void)
{
        int x = 3  ;
        x = rand();
        cout << x << endl ;
        x = rand();
        cout << x << endl ;
}
```

41
18467

rand()

- `srand()`
  - Include <cstdlib>
  - Give a starting value for the `rand()` function
  - Usually used once in program

rand()

```
#include <iostream>
#include <ctime>
using namespace::std;


void main(void)
{
        int x = 3  ;
        srand (time(0));
        cout << x << endl ;
}
```

```
3
```

rand()

```
#include <iostream>
#include <ctime>
using namespace::std;


void main(void)
{
        int x = 3  ;
        srand (time());
        cout << x << endl ;
}
```

```
Compiler error: need value for time
```

rand()

```
#include <iostream>
#include <ctime>
using namespace::std;


void main(void)
{
        int x = 3  ;
        srand (time(0));
        x = rand () ;
        cout << x << endl ;
}
```

What happened when compiling many times ?

```
456
12345
130
```

Now , *when compiling many times ...*
Every time we have a new different value :)

rand()

```
#include <iostream>
#include <ctime>
using namespace::std;

void main(void)
{
        int x = 3  ;
        srand (5);
        x = rand () ;
        cout << x << endl ;
}
```

What happened when compiling many times ?

40

40

40

***Not an appropriate srand() paramater** leads to in-suffiencient using of srand() !!!*

rand()

- Scaling & shifting
  - %
    - `x % y // returns a value between 0 to y-1`
    - let's see an example …

rand()

```
#include <iostream>
#include <cstdlib>
using namespace::std;


void main(void)
{
        int i ;
        i = rand() % 6 + 1 ;
        // rand() % 6 : number between 0 to 5
        // +1 : makes a shift of the range we have
        // 0 to 5 becomes 1 to 6
        cout << i << endl ;
        // here will print a number absolutely between 1 to 6
}
```

6

Or any other number between 1 and 6

rand()

```
#include <iostream>
#include <ctime>
using namespace::std;
void main(void)
{
        srand(clock());
        // clock is from <ctime>
        // no needs to value for clock ()
        for (int i = 0 ; i < 10 ; i++)
                cout << rand() << endl ;
}
```

```
714
4415
16337
1507
14053
5430
30050
16164
20477
3146
Press any key to continue . . .
```

Storage Classes

- *Static Storage : (**Very important**)*
  - `static` key word
    - function's local variables
    - *Keeps values between functions call*
      - ***Watch it ... Watch it ... Watch it ... Watch it ... Watch it ...***
    - Known ***ONLY*** on ***ITS OWN*** function

Storage Classes

- *Static Storage : (**Very important**)*

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

void StaticMyHeadache()
{
        static int x = 8 ;
        cout << "In function , x = " <<  x << endl ;
        x*=2 ;
        cout << "In function , x = " <<  x << endl ;
}

int main()
{
        int x = 3 ;
        cout << x << endl ;
        StaticMyHeadache();
        cout << x << endl ;
        StaticMyHeadache();
}
```

Defined and initialized at the same time in the *first time of calling the function* then the compiler no longer parse this line of code

Storage Classes

- *Static Storage : (**Very important**)*

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

void StaticMyHeadache()
{
        static int x = 8 ;
        cout << "In function , x = " <<  x << endl ;
        x*=2 ;
        cout << "In function , x = " <<  x << endl ;
}

int main()
{
        int x = 3 ;
        cout << x << endl ;
        StaticMyHeadache();
        cout << x << endl ;
        StaticMyHeadache();
}
```

What's the output ?

Storage Classes

- *Static Storage : (**Very important**)*

```
3
In function , x = 8
In function , x = 16
3
In function , x = 16
In function , x = 32
Press any key to continue
```

Storage Classes

- *Static Storage : (__Very important__)*
  - `extern` key word
    - Global variables accessible to functions in
      - *__Same file__*
      - *__Other files__*
    - Must be declared in *__each file__* which has been used in .
    - Used to access Global variable in another file

Storage Classes - extern

1st file

```
#include <iostream>
using namespace::std;
void main(void)
{
        int MyGlobalVar;
}
```

2nd file

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

extern int MyGlobalVar;
```

Storage Classes - extern

1st file

```
#include <iostream>
using namespace::std;
void main(void)
{
        int MyGlobalVar;
}
```

2nd file

```
#include <iostream>
using namespace::std;
void main(void)
{
        extern int MyGlobalVar:
}
```

```
Compiler error … why ?
```

```
2 main's in the same project … that's wrong
```

Playing with scopes !

- Local Scopes
- Global Scopes

- **__The Golden Rule__**
  - *Life time of block scope variables is lifetime of block*

Playing with scopes I

```
#include <iostream>

using namespace::std;

void main(void)
{
        int x = 9 , y = 3 ;
        {
                cout << "This is inner block " << endl;
        }
}
```

```
Compiler: ....
```

```
This is inner block
```

Playing with scopes I

```
#include <iostream>

using namespace::std;

void main(void)
{
        int x = 9 , y = 3 ;
        {
                cout << x << endl ;
        }
}
```

```
9
```

Playing with scopes I

```
#include <iostream>

using namespace::std;

void main(void)
{
        int x = 9 , y = 3 ;
        {
                cout << x << endl ;
        }
        cout << x << endl;
}
```

```
9
9
```

Playing with scopes I

```
#include <iostream>

using namespace::std;

void main(void)
{
        int x = 9 , y = 3 ;
        {
                int x = 4  ;
                cout << x << endl ;
        }
}
```

```
4
```

Playing with scopes I

```
#include <iostream>

using namespace::std;


void main(void)
{
        int x = 9 , y = 3 ;
        {
                int x ;
                x = 4 ;
                cout << x << endl ;
        }
         cout << x << endl ;
}
```

```
4
9
```

Playing with scopes I

```
#include <iostream>

using namespace::std;


void main(void)
{
        int x = 9 , y = 3 ;
        {
                int x = 4  ;
                cout << x << endl ;
        }
         cout << y << endl ;
}
```

```
4
3
```

Playing with scopes I

```
#include <iostream>

using namespace::std;


void main(void)
{
        int x = 9 ;
        {
                int x = 4 , y = 3  ;
                cout << x << endl ;
        }
         cout << y << endl ;
}
```

```
Compiler error : 'y' undeclared identifier
```

Playing with scopes I

```
#include <iostream>

using namespace::std;
int x = 35 ;                                    // Global Variable

void SoSo ()
{
        int x = 8 ;                             // local Variable
}

void main(void)
{
   int x = 9 , y = 3 ;
        {
                int x = 6 ;                     // local Variable
        }
    cout << x << endl ;
}
```

```
9
```

Playing with scopes !

```
#include <iostream>

using namespace::std;
int x = 35 ;                                    // Global Variable

void SoSo ()
{
        int x = 3 ;                             // local Variable
}

void main(void)
{
        int x = 9 , y = 3 ;
        {
                int x = 6 ;                     // local Variable
        }
        cout << ::x << endl ;
}
```

```
35
```

Playing with scopes !

```
#include <iostream>

using namespace::std;
int x = 34 ;                                    // Global Variable

void SoSo ()
{
        int x = 3 ;                             // local Variable
        cout << ::x << endl ;
}

void main(void)
{
        int x = 9 , y = 3 ;
        {
                int x = 6 ;                     // local Variable
        }
}
```

```
Compile & Run  has nothing in output
Nothing to output , yet !
```

Playing with scopes !

```
#include <iostream>

using namespace::std;
int x = 34 ;                                    // Global Variable

void SoSo ()
{
        int x = 3 ;                             // local Variable
        cout << x << endl ;
}

void main(void)
{
        int x = 9 , y = 3 ;
        {
                int x = 6 ;                     // local Variable
        }
        SoSo();
}
```

```
3
```

Playing with scopes !

```
#include <iostream>

using namespace::std;
int x = 34 ;                                    // Global Variable

void SoSo ()
{
        int x = 3 ;                             // local Variable
        cout << ++::x << endl ;
}

void main(void)
{
        int x = 9 , y = 3 ;
        {
                int x = 6 ;                     // local Variable
        }
        SoSo();
}
```

```
35
```

Playing with scopes !

```
#include <iostream>

using namespace::std;
int x = 34 ;                                    // Global Variable

void SoSo ()
{
        int x = 3 ;                             // local Variable
        cout << ::x++ << endl ;
}

void main(void)
{
        int x = 9 , y = 3 ;
        {
                int x = 6 ;                     // local Variable
        }
        SoSo();
        cout << ::x << endl ;
}
```

```
34
35
```

Inline functions

- `Inline` functions
  - `Inline` key word
    - For small , common-used functions
    - No function call , just copy the code into the program
    - Compile can ***ignore*** `Inline`

Inline functions

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

int IWannaSleep(int);

void main( )
{
        int x;
        cout << "Enter the Input Value: ";
        cin>>x;
        cout<<"\n The Output is: " << IWannaSleep(x) << endl ;
}

inline int IWannaSleep (int x1)
{
        return 5*x1;
}
```

```
Enter the Input Value: 2

 The Output is: 10

Press any key to continue
```

Math library functions

Method	Description	Example
`ceil( x )`	rounds x to the smallest integer not less than x	`ceil( 9.2 ) is 10.0` `ceil( -9.8 ) is -9.0`
`cos( x )`	trigonometric cosine of x (x in radians)	`cos( 0.0 ) is 1.0`
`exp( x )`	exponential function e^x	`exp( 1.0 ) is 2.71828` `exp( 2.0 ) is 7.38906`

`fabs( x )`	absolute value of x	`fabs( 5.1 ) is 5.1`
		`fabs( 0.0 ) is 0.0`
		`fabs( -8.76 ) is 8.76`

`floor( x )`	rounds x to the largest integer not greater than x	`floor( 9.2 ) is 9.0` `floor( -9.8 ) is -10.0`
`fmod( x, y )`	remainder of x/y as a floating-point number	`fmod( 13.657, 2.333 ) is 1.992`
`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( 100.0 ) is 2.0`
`pow( x, y )`	x raised to power y (x^y)	`pow( 2, 7 ) is 128` `pow( 9, .5 ) is 3`
`sin( x )`	trigonometric sine of x (x in radians)	`sin( 0.0 ) is 0`
`sqrt( x )`	square root of x	`sqrt( 900.0 ) is 30.0` `sqrt( 9.0 ) is 3.0`
`tan( x )`	trigonometric tangent of x (x in radians)	`tan( 0.0 ) is 0`

Functions – Part 2

Recursion I

- The most common example
  - The factorial
  - Thinking recursion by knowing the main equations …
    - $0! = 1$; $n = 0$
    - $n! = n * (n-1)!$; $n > 0$

Recursion I

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

int factorial(int n)
{
        if (n == 0 )
        {
                return 1 ;
        }
        else
        {
                return n*factorial (n-1) ;
        }
}

void main(void)
{
        cout << factorial(3);
}
```

```
C:
```

Recursion I

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

int factorial(int n)
{
        if (n == 0 )
                return 1 ;
        else
                return n*factorial (n-1) ;
}

void main(void)
{
        cout << factorial(3);
}
```

```
C:
```

Recursion I

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

int factorial(int n)
{
        if ( n = 0 )
                return 1 ;
        else
                return n*factorial(n-1) ;
}

void main(void)
{
        cout << factorial(3);
}
```

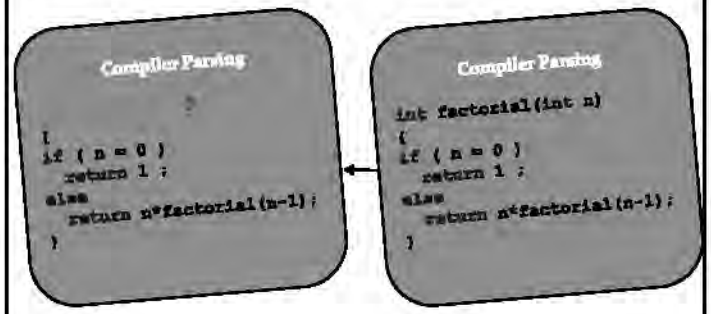
Compile but runtime error ! .. stack over flow ! "=" and not "==" !!!

So the output should 1 right?!!!

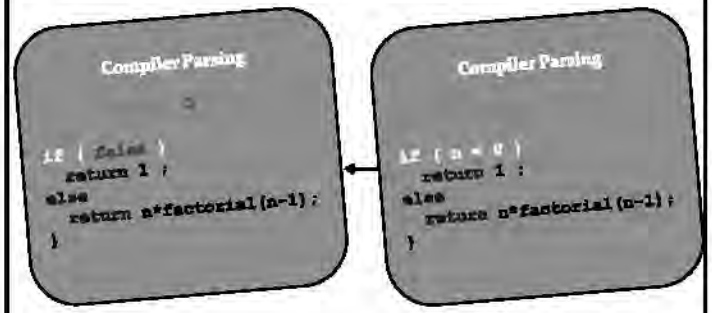
No!!!!! if(n=0) is the same as if(false) and that it never enter the scope

250 pts

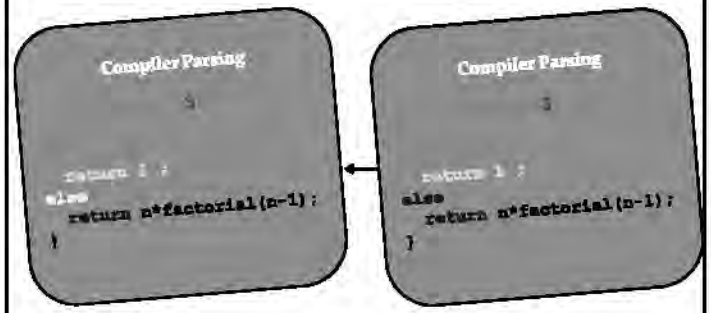
Recursion I



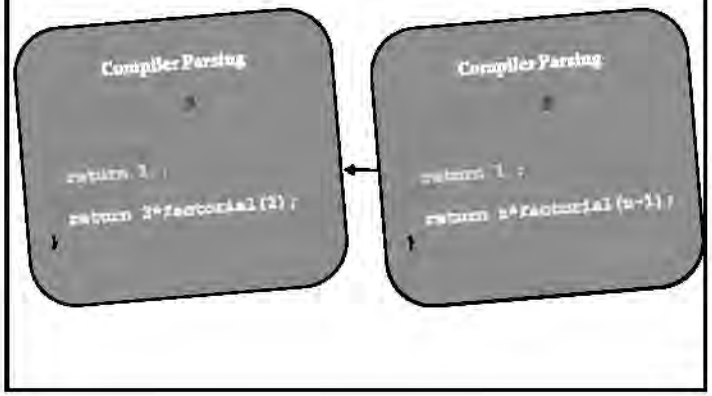
Recursion I



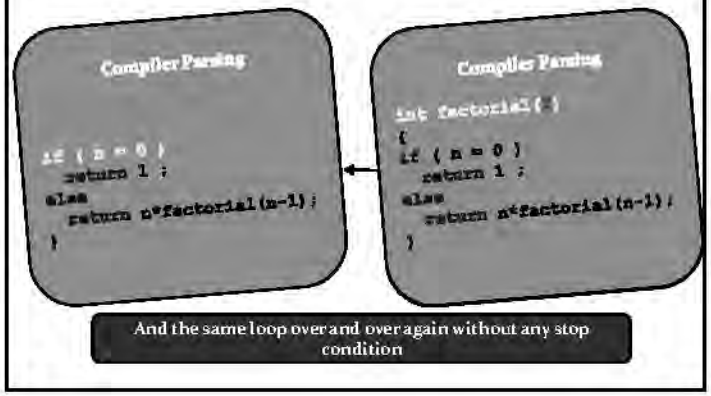
Recursion I



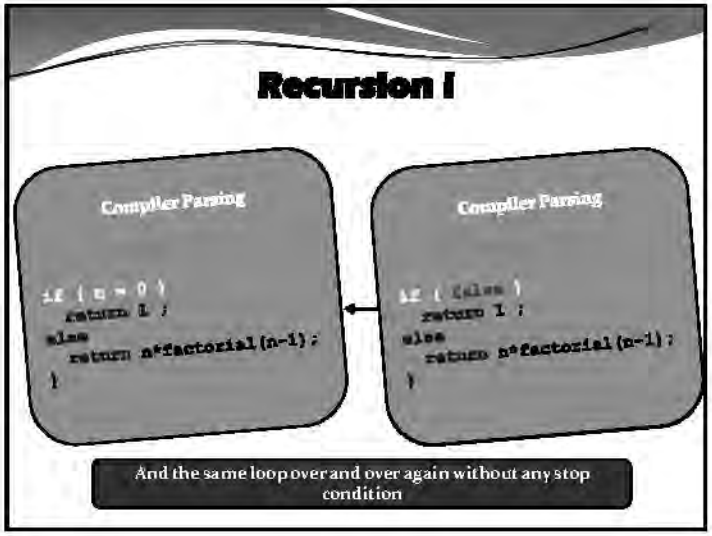
Recursion I



Recursion I



Recursion I



Recursion I

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

int xfactorial(int n)
{
        return 234 ;
}

void main(void)
{
        int x =  xfactorial(3);
        cout << x ;
}
```

```
243
```

```
Not a recursive function anymore !
```

Recursion I

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

int xfactorial(int n)
{
        if (n = 0 )
                return 1 ;
}

void main(void)
{
        cout << xfactorial(3);
}
```

```
0
```

Recursion I

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

int xfactorial(int n)
{
        if (n = 0 )
                return 234 ;
}

void main(void)
{
        int x =  xfactorial(3);
        cout << x ;
}
```

```
0
```

Recursion I

- The normal steps for a recursion problem
  - ***Every*** recursion definition must have
    - One or more "*base case*"
  - Any general case must eventually reduced to *base case*
    - Otherwise , what happened ?
  - *base case* stops the Recursion
- Remember
  - Every call to a recursive function has *its own* set of parameters

Recursion I

- How to think recursively ?
  - It's the most enjoyable method when getting used to it
  - Also , sometimes it's the smartest & the hardest to figure out :)
- So , How to Recursive ?
  - *Finding a* **RULE** *that* **RULE** *the problem !*
  - Recursive it !

Recursion I

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

void MyFirstRecFun(int times)
{
        if (times > 0 )
        {
                cout << "this is a Recursive call . \n ";
                MyFirstRecFun(times - 1 );
        }
}

void main(void)
{
        int x ;
        MyFirstRecFun(4);
}
```

```
this is a Recursive call .
this is a Recursive call .
this is a Recursive call .
this is a Recursive call .
Press any key to continue . . .
```

Note the space after first line coz we have a space after \n

Recursion I

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

void MyFirstRecFun(int times)
{
        if (times >= 0 )
        {
                cout << "this is a Recursive call . \n ";
                MyFirstRecFun(times - 1 );
        }
}

void main(void)
{
        int x ;
        MyFirstRecFun(4);
}
```

Note the space after first line coz we have a space after \n

Recursion I

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

void MyFirstRecFun(int times)
{
        if (times < 0 )
        {
                cout << "this is a Recursive call . \n ";
                MyFirstRecFun(times - 1 );
        }
}

void main(void)
{
        int x ;
        MyFirstRecFun(4);
}
```

```
Press any key to continue . . .
```

Recursion I

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

void MyFirstRecFun(int times)
{
        if (times > 0 )
        {
                cout << "this is a Recursive call . Number " << times << endl;
                MyFirstRecFun(times  - 1 );
        }
}

void main(void)
{
        int x ;
        MyFirstRecFun(3);
}
```

```
this is a Recursive call . Number 3
this is a Recursive call . Number 2
this is a Recursive call . Number 1
Press any key to continue . . .
```

Recursion I

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

void MyFirstRecFun(int times)
{
        if (times > 0 )
        {
                MyFirstRecFun(times - 1 );
                cout << "this is a Recursive call : Number '" <<times<<endl;
        }
}

void main(void)
{
        int x ;
        MyFirstRecFun(3);
}
```

```
this is a Recursive call : Number 1
this is a Recursive call : Number 2
this is a Recursive call : Number 3
Press any key to continue
```

Recursion I

- Recursive function sum of numbers from 1 to n
- Discussion …
  - How to do it ?
    - Finding the idea !
      - ***Rule*** to ***Rule*** !
  - We have …
    - 1 + 2 + 3 + …. + n
    - The *Rule* is :
      - n + sum of the numbers from 1 to n-1($= \sum_{1}^{n-1}$)
        - n + summation (n-1)
      - Now we ***Ruled !!***

Recursion I

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

int Sum(int n)
{
        if (n == 1 )
                return 1 ;
        else
                return (n + Sum(n-1));
}

void main(void)
{
        cout << Sum(4) << endl ;
}
```

```
10
```

Recursion I

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

int Sum(int n)
{
        if (n == 1 )
                return 0 ;
        else
                return (n + Sum(n-1));
}

void main(void)
{
        cout << Sum(4) << endl ;
}
```

```
10
```

Recursion !

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

int Sum(int n)
{
        if (n == 0 )
                return 1 ;
        else
                return (n + Sum(n-1));
}

void main(void)
{
        cout << Sum(4) << endl ;
}
```

11

Recursion !

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

int Sum(int n)
{
        if (n == 1 )
                return 1 ;
        else
        {
                int s ;
                return s;
        }
}

void main(void)
{
        cout << Sum(4) << endl ;
}
```

0

Not a recursion call [illegible] , returns [illegible]

Recursion !

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

int Sum(int n)
{
        int s ;
        if (n == 1 )
                return 1 ;
        else
        {
                s  = n + Sum(n-1);
        }
}

void main(void)
{
        cout << Sum(4) << endl ;
        system("pause");
}
```

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

int Sum(int n)
{
    if (n == 1 )
        return 1 ;
    else
    {
        return n + Sum(n-1);
    }
}

void main(void)
{
    cout << Sum(4) << endl ;
    system("pause");
}
```

10 – Visual Studio 2010

Should return 1 in first test!

Recursion !

```
#include <iostream>
using namespace::std;
int s = 0  ;
int Sum(int n)
{
        if (n == 1 )
                return 1 ;
        else
                return  n + 2*n*sum(n-1);

}

void main(void)
{
        cout << Sum(4) << endl ;
}
```

Compiler error [illegible] sum is small letter !! (Only one S)

Recursion I

- Recursive function to find x^n
- Discussion ...
  - How to do it ?
    - Finding the idea !
      - ***Rule*** to ***Rule*** !
  - We have ...
    - 2^0 = 1 // base case
    - 2^1 = 2*2^0 2^2 = 2*2^1; 2^3 = 2*2^2; ...
  - The *Rule* is :
    - x* x^(n-1)
    - Now we ***Ruled*** !!

Recursion I

```
#include <iostream>
using namespace::std;
int s = 0  ;

int power (int x , int n )
{
        if (n == 0 )
                return 1 ;
        else
        {
                return (x*power(x,n-1));
        }
}

void main(void)
{
        cout << power(2,3) << endl ;
}
```

Recursion I

```
#include <iostream>
using namespace::std;
int s = 0  ;

int power (int x , int n )
{
        if (x == 0)
                return 1 ;
        else
        {
                return (x*power(x,n-1));
        }
}

void main(void)
{
        cout << power(2,3) << endl ;
}
```

Recursion I

```
#include <iostream>
using namespace::std;
int s = 0  ;

int power (int x , int n)
{
        if (n == 0 )
                return 1 ;
        else
        {
                return (x*power(x));
        }
}

void main(void)
{
        cout << power(2,3) << endl ;
}
```

Recursion I

```
#include <iostream>
using namespace::std;
int s = 0  ;

void CrazyMessages (int times )
{
 cout << "Msg called with " << times << "\n" ;
 if (times > 0 )
 {
          cout << "This is recursive func .\n" ;
          CrazyMessages(times-1);
 }
 cout << "Msg Returning with " << times << " !!!\n";
}

void main(void)
{
        CrazyMessages (3);
}
```

```
Msg called with 3
This is recursive func .
Msg called with 2
This is recursive func .
Msg called with 1
This is recursive func .
Msg called with 0
Msg Returning with 0 !!!
Msg Returning with 1 !!!
Msg Returning with 2 !!!
Msg Returning with 3 !!!
Press any key to continue
```

Recursion I

```
#include <iostream>
using namespace::std;
int s = 0  ;

void CrazyMessages (int times )
{
 cout << "Msg called with " << times << "\n" ;
 if (times > 0 )
 {
          cout << "This is recursive func .\n" ;
          CrazyMessages(times-1);
 }
 cout << "Msg Returning with " <<times<<" !\!!\n";
}

void main(void)
{
        CrazyMessages (3);
}
```

```
Msg called with 3
This is recursive func .
Msg called with 2
This is recursive func .
Msg called with 1
This is recursive func .
Msg called with 0
Msg Returning with 0 !!!
Msg Returning with 1 !!!
Msg Returning with 2 !!!
Msg Returning with 3 !!!
Press any key to continue
```

Recursion I

```
#include <iostream>
using namespace::std;
int s = 0  ;

void CrazyMessages (int times )
{
 cout << "Msg called with " << times << "\n" ;
 if (times >= 0 )
 {
  cout << "This is recursive func .\n" ;
  cout << "Times are : " << times << endl ;
  CrazyMessages(times-1);
 }
 cout<<"Msg Returning with "<<times<<" !!!\n";

}

void main(void)
{
        CrazyMessages (3);
}
```

```
Msg called with 3
This is recursive func .
Times are : 3
Msg called with 2
This is recursive func .
Times are : 2
Msg called with 1
This is recursive func .
Times are : 1
Msg called with 0
This is recursive func .
Times are : 0
Msg called with -1
Msg Returning with -1 !!!
Msg Returning with 0 !!!
Msg Returning with 1 !!!
Msg Returning with 2 !!!
Msg Returning with 3 !!!
Press any key to continue . . .
```

Recursion I

```
#include <iostream>
using namespace::std;
int s = 0  ;

int power (int x , n )
{
        if (n == 0 )
                return 1 ;
        else
        {
                return (x*power(x));
        }
}

void main(void)
{
        cout << power(2,3) << endl ;
}
```

```
Compiler error... missing int before n in function header
```

Recursion VS Iteration I

- To solve a problem , 2 ways :
  - *Iteration*
    - Explicit loop
  - *Recursion*
    - Repeated function calls
- How to stop (Terminate) :
  - *Iteration*
    - Loop's condition fails
  - *Recursion*
    - Base case encountered

Recursion VS Iteration I

- Both , iteration & Recursion can have Infinite loops !
- *Performance*
  - Iteration
- *Software Engineering*
  - Recursion (Not better all the time!)

Recursion VS Iteration I

- *Memory Allocation*
  - Every recursive function call has its own sets of parameters & (automatic) local variables
  - When function is called
    - Memory space allocated
  - When function is terminated
    - Memory space de-allocated
- *Efficiency*
  - Recursive is more slowly than its iterative counterpart

Functions Default parameters

Default parameters

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

void print(int x=0 , int n=4 )
{
        cout << x << endl ;
}

void main(void)
{
        print(2,3);
}
```

2

Default parameters

```
#include <iostream>
using namespace::std;
int x = 0 ;

void print(int x=0 , int n )
{
        cout << x << endl ;
}

void main(void)
{
        print(2,3);
}
```

Compiler error: missing default value for parameter 2

When defaulting a parameter in a function , all other parameters - if found - on the right side, should be defaulted too

Default parameters

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

void print(int x , int n=0 )
{
        cout << x << endl ;
}

void main(void)
{
        print(2,3);
}
```

2

Default parameters

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

void print(int x , int n , int y  )
{
        cout << x << endl ;
}

void main(void)
{
        print(2,3,4);
}
```

2

4114 Press

Default parameters

```
#include <iostream>
using namespace::std;
int s = 0  ;

void print(int x , int n , int y = 0  )
{
        cout << x << endl ;
}

void main(void)
{
        print(2,3,4);
}
```

Default parameters

```
#include <iostream>
using namespace::std;
int s = 0  ;

void print(int x , int n , int y = 0  )
{
        cout << x << endl ;
}

void main(void)
{
        print(2,3);
}
```

Default parameters

```
#include <iostream>
using namespace::std;
int s = 0  ;

void print (int x , int n , int y=10 )
{
        cout << x << endl ;
        cout << n << endl ;
        cout << y << endl ;
}

void main(void)
{
        print(2,3);
}
```

```
2
3
10
```

Default parameters

```
#include <iostream>
using namespace::std;
int s = 0  ;

void print(int x , int n=10 , int y )
{
        cout << x << endl ;
        cout << n << endl ;
        cout << y << endl ;
}

void main(void)
{
        print(2,3);
}
```

Default parameters

```
#include <iostream>
using namespace::std;
int s = 0  ;

void print(int x , int n=10 , int y )
{
        cout << x << endl ;
        cout << n << endl ;
        cout << y << endl ;
}

void main(void)
{
        print(2,3);
}
```

Why the compiler couldn't do it?

Compiler error … coz we did'n default the y …

Now , did u know why ? :)

Default parameters

```
#include <iostream>
using namespace::std;
int s = 0  ;

void print(int x , int n=10
{
        cout << x << endl ;
        cout << n << endl ;
        cout << y << endl ;
}

void main(void)
{
        print(2,3);
}
```

Compiler Parsing

```
void print(int x, int n=10, int y);

void main(void)
{
        print(2,3);
}
```

Compiler error … coz we did'n default the y …

Now , did u know why ? :)

Default parameters

```
#include <iostream>
using namespace::std;
int s = 0  ;

void print(int x , int n=10
{
        cout << x << endl ;
        cout << n << endl ;
        cout << y << endl ;
}

void main(void)
{
        print(2,3);
}
```

Compiler Parsing

```
print(2,3);
}
```

Compiler error … coz we did'n default the y …

Now , did u know why ? :)

Default parameters

```
#include <iostream>
using namespace::std;
int s = 0  ;

void print(int x , int n=10
{
        cout << x << endl ;
        cout << n << endl ;
        cout << y << endl ;
}

void main(void)
{
        print(2,3);
}
```

Compiler Parsing

```
print(2,3);
}
```

Compiler error … coz we did'n default the y …

Now , did u know why ? :)

Default parameters

```
#include <iostream>
using namespace::std;
int s = 0  ;

void print(int x , int n=10
{
        cout << x << endl ;
        cout << n << endl ;
        cout << y << endl ;
}

void main(void)
{
        print(2,3);
}
```

Compiler Parsing

print(2, 3);

}

Compiler error .. cos we didn't default the y ..

Now , did u know why ? :)

Default parameters

```
#include <iostream>
using namespace::std;
int s = 0  ;

void print(int x , int n=10
{
        cout << x << endl ;
        cout << n << endl ;
        cout << y << endl ;
}

void main(void)
{
        print(2,3);
}
```

Compiler Parsing

print(2, 3);

}

Compiler error .. cos we didn't default the y ..

Now , did u know why ? :)

Default parameters

```
#include <iostream>
using namespace::std;
int s = 0  ;

void print(int x , int n=10
{
        cout << x << endl ;
        cout << n << endl ;
        cout << y << endl ;
}

void main(void)
{
        print(2,3);
}
```

Compiler Parsing

?

print(2, 3);

}

Compiler error .. cos we didn't default the y ..

Now , did u know why ? :)

Default parameters

```
#include <iostream>
using namespace::std;
int s = 0  ;

void print(int x , int n=10
{
        cout << x << endl ;
        cout << n << endl ;
        cout << y << endl ;
}

void main(void)
{
        print(2,3);
}
```

Compiler Parsing

?

print(2, 3);

}

The compiler don't know what to do? Should it replace the value (3) with the default value of the defaulted parameter (n) or go on to the next parameter (y) and pass it through

Compiler error .. cos we didn't defau...

Now , did u know why ? :)

Default parameters

```
#include <iostream>
using namespace::std;
int s = 0  ;

void print(int x , int n=10
{
```

Compiler Parsing

And now you can know why the compiler can parse the function calling correctly when the function header have the defaulted parameter <u>at the right most</u> of the prototype and not in the middle cox as you saw, the compiler couldn't determine where to pass the value!

The compiler don't know what to do? Should it replace the value (3) with the default value of the defaulted parameter (n) or go on to the next parameter (y) and pass it through

Default parameters

```
#include <iostream>
using namespace::std;
int s = 0  ;

void print(int x , int n=10 , int y = 5 )
{
        cout << x << endl ;
        cout << n << endl ;
        cout << y << endl ;
}

void main(void)
{
        print(2,3);
}
```

Calling by value VS calling by reference

Very Important

Used to be ***Massacre***

Calling by value VS calling by reference

- **Calling by value**
  - *Copy* of data
  - Changes to copy don't affect original
  - Prevent an unwanted side effect
- **Calling by reference (&)**
  - Function directly access data
  - Changes affect original (no copy exist here !)
  - Using & <u>*after*</u> data type

```
void Momo(double & x)
```

Calling by value VS calling by reference

```
#include <iostream>
using namespace::std;
int x = 0  ;

void FuncByValue (int x )
{
        cout << "In function before multiplication , x = "<< x << endl ;
        x = x*4 ;
        cout << "In function after multiplication , x = " << x << endl ;
}

void main(void)
{
        int x = 2 ;
        cout << "In main , before calling the function " << x << endl ;
        FuncByValue(x) ;
        cout << "In main , after calling the function " << x << endl ;

}
```

```
In main , before calling the function: 2
In function before multiplication , x = 2
In function after multiplication , x = 8
In main , after calling the function: 2
Press any key to continue
```

Calling by value VS calling by reference

```
#include <iostream>
using namespace::std;
int x = 0  ;

void FuncByRef (int & x )
{
        cout << "In function before multiplication , x = "<< x << endl ;
        x = x*4 ;
        cout << "In function after multiplication , x = " << x << endl ;
}

void main(void)
{
        int x = 2 ;
        cout << "In main , before calling the function " << x << endl ;
        FuncByRef (x) ;
        cout << "In main , after calling the function " << x << endl ;

}
```

```
In main , before calling the function: 2
In function before multiplication , x = 2
In function after multiplication , x = 8
In main , after calling the function: 8
Press any key to continue
```

Calling by value VS calling by reference

```
#include <iostream>
using namespace::std;
int x = 0  ;

void FuncByRef (& int  x )
{
        cout << "In function before multiplication , x = "<< x << endl ;
        x = x*4 ;
        cout << "In function after multiplication , x = " << x << endl ;
}

void main(void)
{
        int x = 2 ;
        cout << "In main , before calling the function " << x << endl ;
        FuncByRef (x) ;
        cout << "In main , after calling the function " << x << endl ;

}
```

```
Compiler error , & before type
```

Calling by value VS calling by reference

```
#include <iostream>
using namespace::std;
int x = 0  ;

void FuncByRef ( int  x & )
{
        cout << "In function before multiplication , x = "<< x << endl ;
        x = x*4 ;
        cout << "In function after multiplication , x = " << x << endl ;
}

void main(void)
{
        int x = 2 ;
        cout << "In main , before calling the function " << x << endl ;
        FuncByRef (x) ;
        cout << "In main , after calling the function " << x << endl ;

}
```

```
Compiler error , & after variable
```

Calling by value VS calling by reference

```
#include <iostream>
using namespace::std;
int g = 0  ;

int FuncByValue (  int x )
{
        cout << "In function before multiplication , x = "<< x << endl ;
        x = x*4 ;
        cout << "In function after multiplication , x = " << x << endl ;
        return x ;
}

void main(void)
{
        int x = 2 ;
        cout << "In main , before calling the function , x = " << x << endl ;
        cout << FuncByValue(x) << endl ;
        cout << "In main , after calling the function , x = " << x << endl ;

}
```

Calling by value VS calling by reference

```
#include <iostream>
using namespace::std;
int g = 0  ;

int FuncByRef (  int & x )
{
        cout << "In function before multiplication , x = "<< x << endl ;
        x = x*4 ;
        cout << "In function after multiplication , x = " << x << endl ;
        return x ;
}

void main(void)
{
        int x = 2 ;
        cout << "In main , before calling the function , x = " << x << endl ;
        cout << FuncByRef(x) << endl ;
        cout << "In main , after calling the function , x = " << x << endl ;

}
```

Function Overloading

Very Important

Used to be *Massacre*

Function Overloading

- Is a function with
  - ***same name***
  - *different* ***parameters*** ***(Not return type !!!!!!!)***
- That means that the overloaded functions are distinguished in **Signature**
  - ***No return type comparing***

Function Overloading

```
#include <iostream>
using namespace::std;
int z = 0  ;

int f (  int &x )
{
          x = x * 2 ;
          return x ;
}

int f (  float &x )
{
          x = x * 3 ;
          return x ;
}

void main(void)
{
          int y1 = 2 ;
          float y2 = 2 ;

          cout << f(y1) << endl ;
          cout << f(y2) << endl ;
}
```

Function Overloading

```
#include <iostream>
using namespace::std;
int z = 0  ;

int f (  int &x )
{
          x = x * 2 ;
          return x ;
}

float f (  float &x )
{
          x = x * 3 ;
          return x ;
}

void main(void)
{
          int y1 = 2 ;
          float y2 = 2 ;

          cout << f(y1) << endl ;
          cout << f(y2) << endl ;
}
```

Function Overloading

```
#include <iostream>
using namespace::std;
int z = 0  ;

int f (  int x )
{
          x = x * 2 ;
          return x ;
}

float f (  float x )
{
          x = x * 3 ;
          return x ;
}

void main(void)
{
          int y1 = 2 ;
          float y2 = 2 ;

          cout << f(y1) << endl ;
          cout << f(y2) << endl ;
}
```

Function Overloading

```
#include <iostream>
using namespace::std;
int z = 0  ;

int f (  int x )
{
          x = x * 2 ;
          return x ;
}

float f (  float x )
{
          x = x * 0 ;
          return x ;
}

void main(void)
{
          int y1 = 2 ;
          float y2 = 2 ;

          cout << f(y1) << endl ;
          cout << f(y2) << endl ;
}
```

Function Overloading

```
#include <iostream>
using namespace::std;
int s = 0  ;

int f (  int x , int z)
{
        x = x * 2 ;
        cout << x << endl ;
        cout << z << endl ;
        return x ;
}

float f (float x , int z)
{
        x = x * 3 ;
        cout << x << endl ;
        cout << z << endl ;
        return x ;
}

void main(void)
{
        int y1 = 2 ;
        float y2 = 2 ;
        int z1 = 3 , z2 = 2 ;

        f(y1,z1);
        f(y2,z2);
}
```

Function Overloading

```
#include <iostream>
using namespace::std;
int s = 0  ;

int f (  int x , int z)
{
        x = x * 2 ;
        cout << x << endl ;
        cout << z << endl ;
        return x ;
}

float f (float x , int z)
{
        x = x * 3 ;
        cout << x << endl ;
        cout << z << endl ;
        return x ;
}

void main(void)
{
        int y1 = 2 ;
        float y2 = 2 ;
        float z1 = 3 , z2 = 2 ;

        f(y1,z1);
        f(y2,z2);
}
```

Function Overloading

```
#include <iostream>
using namespace::std;
int s = 0  ;

int f (  int x , int z)
{
        x = x * 2 ;
        cout << x << endl ;
        cout << z << endl ;
        return x ;
}

float f (int z , float x)
{
        x = x * 3 ;
        cout << x << endl ;
        cout << z << endl ;
        return x ;
}

void main(void)
{
        int y1 = 2 ;
        float y2 = 2 ;
        int z1 = 3 , z2 = 2 ;

        f(y1,z1);
        f(y2,z2);
}
```

Function Overloading

```
#include <iostream>
using namespace::std;
int s = 0  ;

int f (  int x , int z)
{
        x = x * 2 ;
        cout << x << endl ;
        cout << z << endl ;
        return x ;
}

float f (int z , float x)
{
        x = x * 3 ;
        cout << x << endl ;
        cout << z << endl ;
        return x ;
}

void main(void)
{
        int y1 = 2 ;
        float y2 = 2 ;
        float z1 = 3 , z2 = 2 ;

        f(y1,z1);
        f(y2,z2);
}
```

Control Structure I

Tweaked Control Structure I

- `If \ else`
- `While`
- `do \ while`
  - *Executed at least once whatever the condition is .*
- `for`
- `switch`

Tweaked Control Structure I

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

void main(void)
{
        int x = 4 ;
        if ( x == 4 )
        {
                cout << "True " << endl ;
        }
        else
        {
                cout << "False " << endl ;
        }
}
```

```
True
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 4 ;
        if ( x = 4 )
        {
                cout << "True " << endl ;
        }
        else
        {
                cout << "False " << endl ;
        }
}
```

```
True
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 4 ;
        if ( x = 5 )
        {
                cout << "True " << endl ;
        }
        else
        {
                cout << "False " << endl ;
        }
}
```

```
True
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 4 ;
        if ( x != 4 )
        {
                cout << "True " << endl ;
        }
        else
        {
                cout << "False " << endl ;
        }
}
```

```
False
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 4 ;
        if ( x == 4 )
                cout << "True " << endl ;
        else
                cout << "False " << endl ;
                cout << "GoGo !" << endl ;
}
```

```
True
GoGo !
```

Tweaked Control Structure I

```
#include <iostream>
using namespace::std;
int s = 0  ;

void main(void)
{
    int x = 0;
    if (x = 0)
    {
        cout << "I don't know" << endl ;
    }
    cout << "I know" << endl ;
    system("pause");
}
```

```
I know
```

x = 0 Means False!

Remember that every other number other than 0 means true

Tweaked Control Structure I

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

void main(void)
{
        int x = 5 ;
        while (x != 10)
        {
                x++;
                cout << x << endl ;
        }
}
```

```
6
7
8
9
10
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 5 ;
        while (x != 10)
        {
                cout << x++ << endl ;
        }
}
```

```
5
6
7
8
9
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 5 ;
        while (x != 10)
        {
                cout << ++x << endl ;
        }
}
```

```
6
7
8
9
10
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 5 ;
        while (x != 10)
        {
                ++x;
                cout << x << endl ;
        }
}
```

```
6
7
8
9
10
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 5 ;
        while (x < 10)
        {
                cout << x*x << endl ;
        }
}
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 5 ;
        while (x*x < 10)
        {
                cout << x*x << endl ;
        }
}
```

```
Nothing to print
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 5 ;
        while (cin)
        {
                cin >> x ;
                cout << x*x << endl ;
        }
}
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 5 ;
        do
        {
                cout << x << endl ;
                x++;
        } while ( x < 10 ) ;
}
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 5 ;
        do
        {
                cout << x << endl ;
                x++;
        } while ( x < 10 )
}
```

```
Compiler error : missing ; after the while()
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 5 ;
        do
        {
                cout << x << endl ;
        } while ( x < 4 ) ;
}
```

```
5
```

do while execute at least one , eventhough the condition is not fulfilled .

Tweaked Control Structure I

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

void main(void)
{

int x = 5 ;
do
{
cout << "Enter the x :" << endl ;
cin >> x ;
cout << "You have entered , x = " << x
<< endl ;
cout << "_____"
<< endl ;
} while ( x != 0  ) ;


}
```

```
Enter the x :
2
You have entered , x = 2
_____
Enter the x :
0
You have entered , x = 0
_____
Enter the x :
4
You have entered , x = 4
_____
Enter the x :
23
You have entered , x = 23
_____
Enter the x :
0
You have entered , x = 0
_____
Press any key to continue . . .
```

Tweaked Control Structure I

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

void main(void)
{
        int i = 0 ;
        int c = 5 ;
        do
        {
                ++c ;
                cout << "beep ! , number = " << i << endl ;
                i++;
        } while ( ++c < 10 ) ;
}
```

```
beep ! , number = 0
beep ! , number = 1
beep ! , number = 2
Press any key to continue . . .
```

Tweaked Control Structure I

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

void main(void)
{
        int i = 0 ;
        int c = 5 ;
        do
        {
                ++c ;
                cout << "beep ! , number = " << i << endl ;
                i++;
        } while ( c++ < 10 ) ;
}
```

```
beep ! , number = 0
beep ! , number = 1
beep ! , number = 2
Press any key to continue
```

Tweaked Control Structure I

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

void main(void)
{
        int i = 0 ;
        int c = 5 ;
        do
        {
                ++c ;
                cout << "beep ! , number = " << i << endl ;
                i++;
        } while ( ++c <= 10 ) ;
}
```

```
beep ! , number = 0
beep ! , number = 1
beep ! , number = 2
Press any key to continue
```

Code Cracking

Tweaked Control Structure I

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

void main(void)
{
        for (int i = 0 ; i < 10 ; i++)
        {
        cout<<"beep ! , number = "<<i<<endl;
        }
        cout << "finished" << endl ;
}
```

```
beep ! , number = 0
beep ! , number = 1
beep ! , number = 2
beep ! , number = 3
beep ! , number = 4
beep ! , number = 5
beep ! , number = 6
beep ! , number = 7
beep ! , number = 8
beep ! , number = 9
finished
Press any key to continue
```

Tweaked Control Structure I

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

void main(void)
{
        int i = 5 ;
        for (int i = 0 ; i < 10 ; i++)
        {
        cout << "beep ! , number = " << i
        << endl ;
        }
        cout << "finished" << endl ;
}
```

```
beep ! , number = 0
beep ! , number = 1
beep ! , number = 2
beep ! , number = 3
beep ! , number = 4
beep ! , number = 5
beep ! , number = 6
beep ! , number = 7
beep ! , number = 8
beep ! , number = 9
finished
Press any key to continue
```

Tweaked Control Structure I

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

void main(void)
{
        int i = 5 ;
        int c = 5 ;
        for (int i = 0 ; i < 10 ; i++)
        {
                cout << "beep ! , number = "
                << i << endl ;
                i++;
        }
        cout << "finished" << endl ;
}
```

```
beep ! , number = 0
beep ! , number = 1
beep ! , number = 2
beep ! , number = 3
beep ! , number = 4
finished
```

Tweaked Control Structure I

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

void main(void)
{
        int i = 66 ;
        int c = 5 ;
        for (int i = 0 ; i < 10 ; i++);
        {
                cout << "beep ! , number = " << i << endl ;
                i++;
        }
        cout << "finished" << endl ;
}
```

```
beep ! , number = 66
finished
Press any key to continue
```

Why is that ?

Watch out for the semi colon ";" after the for statement.. Coz it close it

Tweaked Control Structure I

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

void main(void)
{
        int i = 66 ;
        int c = 5 ;
        for (int i = 0 ; i < 10 ; )
        {
                cout << "beep ! , number = " << i << endl ;
                i++;
        }
        cout << "finished" << endl ;
}
```

```
beep ! , number = 0
beep ! , number = 1
beep ! , number = 2
beep ! , number = 3
beep ! , number = 4
beep ! , number = 5
beep ! , number = 6
beep ! , number = 7
beep ! , number = 8
beep ! , number = 9
finished
Press any key to continue
```

Tweaked Control Structure I

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

void main(void)
{
        int i = 66 ;
        int c = 5 ;
        for (int i = 0 ; i < 10 ; );
        {
                cout << "beep ! , number = " << i << endl ;
                i++;
        }
        cout << "finished" << endl ;
}
```

Nothing to print ... Why ?

Or he is still working in an un-infinite loop !

Tweaked Control Structure I

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

void main(void)
{
        int i = 66 ;
        int c = 5 ;
        for (int i = 0 ;  ; )
        {
                cout << "beep ! , number = " << i << endl ;
                i++;
        }
        cout << "finished" << endl ;
}
```

```
beep ! , number = 0
beep ! , number = 1
beep ! , number = 2
beep ! , number = 3
...
beep ! , number = 3
...
beep ! , number = 4[illegible]
```

Un-infinite loop !

Tweaked Control Structure I

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

void main(void)
{
        int i = 66 ;
        int c = 5 ;
        for (int i = 0 ;  ; i++ )
        {
                cout << "beep ! , number = " << i << endl ;
                i++;
        }
        cout << "finished" << endl ;
}
```

```
beep ! , number = 0
beep ! , number = 2
beep ! , number = 4
beep ! , number = 6
...
beep ! , number = 10
...
beep ! , number = [illegible]
```

Un-infinite loop

Tweaked Control Structure I

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

void main(void)
{
        int i = 66 ;
        int c = 5 ;
        for (  )
        {
                cout << "beep ! , number = " << i << endl ;
        }
        cout << "finished" << endl ;
}
```

Compiler error

Tweaked Control Structure I

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

void main(void)
{
        int i = 66 ;
        int c = 5 ;
        for (i = 2 ; i<5 ; i++ )
        {
                cout << "beep ! , number = " << i << endl ;
        }
        cout << "finished" << endl ;
}
```

```
beep ! , number = 2
beep ! , number = 3
beep ! , number = 4
finished
Press any key to continue . . .
```

Tweaked Control Structure I

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

void main(void)
{
        int i = 66 ;
        int c = 5 ;
        for (int i = 0 , j = 3 ; i<5 ; i++ )
        {
                cout << "beep ! , number = " << j++ << endl ;
        }
        cout << "finished" << endl ;
}
```

```
beep ! , number = 3
beep ! , number = 4
beep ! , number = 5
beep ! , number = 6
beep ! , number = 7
finished
Press any key to continue . . .
```

Tweaked Control Structure I

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

void main(void)
{
        int i = 66 ;
        int c = 5 ;
        for (int i = 0 ; j = 3 ; i<5 ; i++ )
        {
                cout << "beep ! , number = " << j++ << endl ;
        }
        cout << "finished" << endl ;
}
```

```
Compiler error ... ; expected
```

Tweaked Control Structure I

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

void main(void)
{
        int i = 66 ;
        int c = 5 ;
        for (int i = 0 , int j = 3 ; i<5 ; i++ )
        {
                cout << "beep ! , number = " << j++ << endl ;
        }
        cout << "finished" << endl ;
}
```

```
Compiler error : j undeclared identifier
```

Tweaked Control Structure I

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

void main(void)
{
        int i = 66 ;
        int c = 5 ;
        int j = 2 ;
        for (int i = 0 , j ; i<5 ; i++ , j-- )
        {
                cout << j << endl ;
        }
}
```

```
2
1
0
-1
-2
```

Tweaked Control Structure I

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

void main(void)
{
        int i = 66 ;
        int c = 5 ;
        int j = 4 ;
        for (int i = 0 , j=0 ; i<5 ; i++ , j-- )
        {
                cout << j++ << endl ;
        }
}
```

```
0
1
2
3
4
```

Tweaked Control Structure I

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

void main(void)
{
        for (int i = 0  ; i<5 ; i++  )
        {
                cout << i++ << endl ;
                break ;
        }
}
```

```
0
```

Tweaked Control Structure I

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

void main(void)
{
        for (int i = 0  ; i<5 ; i++  )
        {
                cout << i++ << endl ;
                if ( i == 34)
                {
                        break ;
                }
        }
}
```

```
0
2
4
```

Tweaked Control Structure I

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

void main(void)
{
        for (int i = 0  ; i<5 ; i++  )
        {
                cout << i << endl ;
                if ( i == 34)
                {
                        break ;
                }
        }
}
```

```
0
1
2
3
4
```

Tweaked Control Structure I

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

void main(void)
{
        for (int i = 0  ; i<5 ; i++  )
        {
                cout << i << endl ;
                if ( i == 2)
                {
                        break ;
                }
        }
}
```

```
0
1
2
```

Tweaked Control Structure I

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

void main(void)
{
        for (int i = 0  ; i<5 ; i++  )
        {
                if ( i == 2)
                {
                        break ;
                }
                cout << i << endl ;

        }
}
```

```
0
1
```

Tweaked Control Structure I

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

void main(void)
{
        for (int i = 0  ; i<5 ; i++  )
        {
                cout << i << endl ;
                if ( i = = 2)
                {
                        break ;
                }
        }
}
```

```
Compiler error ... = = not ==
```

Tweaked Control Structure I

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

void main(void)
{
        for (int i = 0  ; i<5 ; i++  )
        {
                cout << i << endl ;
                if ( i == 2)
                {
                        cout << "WEWEEKK !!!!";
                        break ;
                }
        }
}
```

```
0
1
2
WEWEEKK !!!!Press any key to continue . . .
```

Tweaked Control Structure I

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

void main(void)
{
        for (int i = 0  ; i<5 ; i++  )
        {
                if ( i == 2)
                {
                        continue ;
                }
                cout << i << endl ;
        }
}
```

```
0
1
3
4
```

Tweaked Control Structure I

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

void main(void)
{
        for (int i = 0  ; i<5 ; i++  )
        {
                if ( i == 2)
                {
                        continue ;
                        cout << "WeeWeee" << endl ;
                }
                cout << i << endl ;
        }
}
```

```
0
1
3
4
```

Tweaked Control Structure I

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

void main(void)
{
        for (int i = 0  ; i<5 ; i++  )
        {
                cout << i << endl ;
                if ( i == 2)
                {
                        cout << "WeeeWeee" << endl ;
                        continue ;
                }
        }
}
```

```
0
1
2
WeeeWeee
3
4
Press any key to continue . . .
```

Tweaked Control Structure I

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

void main(void)
{
        for (int i = 0  ; i<5 ; i++ )
        {
                cout << i << endl ;
                if ( i == 2)
                {
                        cout << "WeeeWeee" << endl ;
                }
        }
}
```

```
0
1
2
WeeeWeee
3
4
Press any key to continue . . .
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 3 ;
        cout << "Enter x : " << endl ;
        cin >> x ;
        cout << "You have entered , x = " << x << endl ;
        switch (x)
        {
        case 1 :
         cout << "WOW , i can't believe it , you entered x = 1" << endl;
         break ;
        case 2 :
         cout << "WOW , i can't believe it , you entered x = 2" << endl ;
         break ;
        }
}
```

```
Enter x :
4
You have entered , x = 4
Press any key to continue . . .
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 3 ;
        cout << "Enter x : " << endl ;
        cin >> x ;
        cout << "You have entered , x = " << x << endl ;
        switch (x)
        {
        case 1 :
         cout << "WOW , i can't believe it , you entered x = 1" << endl;
         break ;
        case 2 :
         cout << "WOW , i can't believe it , you entered x = 2" << endl ;
         break ;
        }
}
```

```
Enter x :
2
You have entered , x = 2
WOW , i can't believe it , you entered x = 2
Press any key to continue . . .
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 3 ;
        cout << "Enter x : " << endl ;
        cin >> x ;
        cout << "You have entered , x = " << x << endl ;
        switch (x)
        {
        case 1 :
                cout << "WOW , i can't believe it , you entered x = 1" << endl ;
                break ;
        case 2 :
                cout << "WOW , i can't believe it , you entered x = 2" << endl ;
                 break ;
        default :
                cout << "Not a 1 or 2 " << endl;
                 break ;
        }
}
```

```
Enter x :
5
You have entered , x = 5
Not a 1 or 2
Press any key to continue . . .
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 3 ;
        cout << "Enter x : " << endl ;
        cin >> x ;
        cout << "You have entered , x = " << x << endl ;
        switch (x)
        {
        case 1 :
                cout << "WOW , i can't believe it , you entered x = 1" << endl ;
                break ;
        case 2 :
                cout << "WOW , i can't believe it , you entered x = 2" << endl ;
                 break ;
        default :
                cout << "Not a 1 or 2 " << endl;
        }
}
```

```
Enter x :
3
You have entered , x = 3
Not a 1 or 2
Press any key to continue
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 3 ;
        cout << "Enter x : " << endl ;
        cin >> x ;
        cout << "You have entered , x = " << x << endl ;
        switch (x)
        {
        case 1 :
                cout << "WOW , i can't believe it , you entered x = 1"
                << endl ;
        case 2 :
                cout << "WOW , i can't believe it , you entered x = 2"
                << endl ;
        default :
                cout << "Not a 1 or 2 " << endl;
        }
}
```

Coz of forgetting break

```
Enter x :
1
You have entered , x = 1
WOW , i can't believe it , you entered x = 1
WOW , i can't believe it , you entered x = 2
Not a 1 or 2
Press any key to continue
```

Tweaked Control Structure I

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

void main(void)
{
        int x = 3 ;
        cout << "Enter x : " << endl ;
        cin >> x ; // 1
        cout << "You have entered , x = " << x << endl ;
        switch (x)
        {
        case 1 :
                cout << "WOW , i can't believe it , you entered x = 1" << endl ;
                cout << x++ << endl;
        case 2 :
                cout << "WOW , i can't believe it , you entered x = 2" << endl ;
        default :
                cout << "Not a 1 or 2 " << endl;
        }
}
```

```
Enter x :
1
You have entered , x = 1
WOW , i can't believe it , you entered x = 1
1
WOW , i can't believe it , you entered x = 2
Not a 1 or 2
Press any key to continue
```

Tweaked Control Structure I

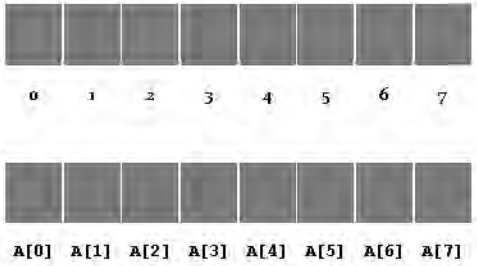
```
#include <iostream>
using namespace::std;

void main(void)
{
        int x = 3 ;
        cout << "Enter x : " << endl ;
        cin >> x ;
        cout << "You have entered , x = " << x << endl ;
        switch (x)
        {
        case 1 :
                cout << "WOW , i can't believe it , you entered x = 1" << endl ;
                cout << x++ << endl;
                break ;
        case 2 :
                cout << "WOW , i can't believe it , you entered x = 2" << endl ;
                break ;
        default :
                cout << "Not a 1 or 2 " << endl;
        }
}
```

```
Enter x :
1
You have entered , x = 1
WOW , i can't believe it , you entered x = 1
1
Press any key to continue
```

Arrays

C++ data types

Arrays



Note : nth element in position **n-1**

Arrays

- Declaring arrays ....

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

void main(void)
{
        int A [1000] ; // means the position ate from 0 to 999
}
```

Arrays

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


void main(void)
{
    int intArr [10] = {4,5,56,6} ;
    cout << intArr << endl ;
    system("pause");
}
```

```
00457B08
Press any key to continue . . .
```

When printing the array it print it's location on memory by default

Just the (char arr []) Don't do that! It prints the char contained in it! We'll see it in minutes ;)

The array "variable" is the location of its first element in memory

Arrays

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


void main(void)
{
    int intArr [10] = {4,5,56,6} ;
    cout << intArr[0] << endl ;
    system("pause");
}
```

```
4
Press any key to continue . . .
```

When printing the array it print it's location on memory by default

Just the (char arr []) Don't do that! It prints the char contained in it! We'll see it in minutes ;)

Arrays

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


void main(void)
{
        int A [5] ;
        A[3] = 0 ;
        A[0] = -1 ;
        for (int i = 0 ; i<=4 ; i ++)
        {
                cout << "A[" << i << "] = " << A[i] << endl ;
        }
}
```

```
A[0] = -1
A[1] = 3092896
A[2] = 5040544
A[3] = 0
A[4] = 1523006656
Press any key to continue . . .
```

Note that it's not a compiler error ... `i ++`

Arrays

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


void main(void)
{
        int A [5] ;
        A[0] = 1;
        A[1] = 3;
        A[2] = 4;
        A[3] = 7;
        A[4] = 89;
        for (int i = 0 ; i<=4 ; i ++)
        {
                cout << "A[" << i << "] = " << A[i] << endl ;
        }
}
```

```
A[0] = 1
A[1] = 3
A[2] = 4
A[3] = 7
A[4] = 89
Press any key to continue . . .
```

Arrays

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


void main(void)
{
        int A [5] ;
        A[0] = 1;
        A[1] = 3;
        A[2] = 4;
        A[3] = 7;
        A[4] = 89;
        A[5] = 34;
        for (int i = 0 ; i<=4 ; i ++)
        {
                cout << "A[" << i << "] = " << A[i] << endl ;
        }
}
```

```
A[0] = [illegible]
A[1] = [illegible]
A[2] = [illegible]
A[3] = [illegible]
A[4] = [illegible]
Press any key to continue
```

Arrays

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


void main(void)
{
        int A [5];
        A[3] = 0 ;
        A[0] = -1 ;
        for (int i = 0 ; i<=10 ; i ++)
        {
                cout << "A[" << i << "] = " << A[i] << endl ;
        }
}
```

Compiler error ?

```
A[0] = -1
A[1] = [illegible]
A[2] = [illegible]
A[3] = 0
A[4] = [illegible]
A[5] = [illegible]
A[6] = [illegible]
A[7] = [illegible]
A[8] = [illegible]
A[9] = [illegible]
A[10] = [illegible]
Press any key to continue...
```

Arrays

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


void main(void)
{
        int A [5] ;
        A[3] = 0 ;
        A[0] = 34 ;
        A[3] = 2 ;
        A[2] = 1 ;
        A[3+1] = 5;
        A[1] = 7 ;
        A[4] = 6 ;
        for (int i = 0 ; i<=4 ; i ++)
        {
                cout << "A[" << i << "] = " << A[i] << endl ;
        }
}
```

```
A[0] = 34
A[1] = [illegible]
A[2] = [illegible]
A[3] = [illegible]
A[4] = [illegible]
Press any key to continue . . .
```

Arrays

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


void main(void)
{
        int A [5] ;
        A[3] = 0 ;
        A[0] = 34 ;
        A[3] = 2 ;
        A[2] = 1 ;
        A[3+1] = 5;
        A[1] = 7 ;
        A[4] = 6 ;
        for (int i = -1 ; i<=3 ; i ++)
        {
                cout << "A[" << i+1 << "] = " << A[i+1] << endl ;
        }
}
```

```
A[0] = [illegible]
A[1] = [illegible]
A[2] = 1
A[3] = [illegible]
A[4] = [illegible]
Press any key to continue...
```

Arrays

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


void main(void)
{
        int A [5] ;
        A[3] = 0 ;
        A[0] = 34 ;
        A[2] = 2 ;
        A[2] = 1 ;
        A[2+1] = 5;
        A[1] = 7 ;
        A[4] = 6 ;
        for (int i = -1 ; i<=2 ; i ++)
        {
                cout << "A[" << i << "] = " << A[i] << endl ;
        }
}
```

```
A[-1] = 1637984161
A[0] = 34
A[1] = 7
A[2] = 1
A[3] = 0
Press any key to continue
```

Arrays

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


void main(void)
{
        int A [5] ;
        A[3] = 0 ;
        A[0] = 34 ;
        A[2] = 2 ;
        A[2] = 1 ;
        A[2+1] = 5;
        A[1] = 7 ;
        A[4] = 6 ;
        for (int i = -1 ; i<4 ; i ++)
        {
                cout << "A[" << i << "] = " << A[i] << endl ;
        }
}
```

```
A[-1] = 1800930049
A[0] = 34
A[1] = 7
A[2] = 1
A[3] = 0
Press any key to continue
```

Arrays

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


void main(void)
{
        int A      [5]  , X[2];
        A[3] = 0 ;
        A[0] = -1 ;
        X[0] = A[3] ;
        X[1] = A[2] ;
        for (int i = 0 ; i<2 ; i ++)
        {
                cout << "X[" << i << "] = " << X[i] << endl ;
        }
}
```

```
X[0] = 0
X[1] = -858993...
Press any key to continue
```

Arrays

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


void main(void)
{
        int a[5] = {3,5,2,4,33} ; // initialising
        for (int i = 0 ; i<5 ; i ++)
        {
                cout << "a[" << i << "] = " << a[i] << endl ;
        }
}
```

```
a[0] = 3
a[1] = 5
a[2] = 2
a[3] = 4
a[4] = 33
Press any key to continue
```

Arrays

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


void main(void)
{
        int a[5] = {3,5,2,4,33,12} ;        // initializing
        for (int i = 0 ; i<5 ; i ++)
        {
                cout << "a[" << i << "] = " << a[i] << endl ;
        }
}
```

```
Compile::error
```

Arrays

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


void main(void)
{
        int a[5] = {3,5,2} ;        // initializing
        for (int i = 0 ; i<5 ; i ++)
        {
                cout << "a[" << i << "] = " << a[i] << endl ;
        }
}
```

```
a[0] =3
a[1] =5
a[2] =2
a[3] =0
a[4] =0
Press any key to continue . . .
```

Arrays

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


void main(void)
{
        int a[5] = (4) ; // initializing
        for (int i = 0 ; i<5 ; i ++)
        {
                cout << "a[" << i << "] = " << a[i] << endl ;
        }
}
```

```
a[0] = 4
a[1] = 0
a[2] = 0
a[3] = 0
a[4] = 0
Press any key to continue . . .
```

Arrays

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


void main(void)
{
        int a[5] = {0} ; // initializing
        for (int i = 0 ; i<5 ; i ++)
        {
                cout << "a[" << i << "] = " << a[i] << endl ;
        }
}
```

```
a[0] = 0
a[1] = 0
a[2] = 0
a[3] = 0
a[4] = 0
Press any key to continue . . .
```

Arrays

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


void main(void)
{
        int a[] = {32,34,12,23} ;
        for (int i = 0 ; i<4 ; i ++)
        {
                cout << "a[" << i << "] = " << a[i] << endl ;
        }
}
```

```
Compiler error
```

```
int a[] = {32,34,12,23} ; // means an array size is 4 element
```

```
a[0] = 32
a[1] = 34
a[2] = 12
a[3] = 23
Press any key to continue
```

Arrays

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


void main(void)
{
        int a[] = {32,34,12,23} ;
        for (int i = 0 ; i<5 ; i ++)
        {
                cout << "a[" << i << "] = " << a[i] << endl ;
        }
}
```

```
a[0] = 32
a[1] = 34
a[2] = 12
a[3] = 23
a[4] = 4196176
Press any key to continue
```

Arrays

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


void main(void)
{
        int a[5];
        for (int i = 0 ; i<5 ; i ++)
        {
                a[i] = i+2 ;
        }
        cout << "_____" << endl ;
        cout << "The outputted array is; " << endl;
        for (int i=0 ; i<5 ; i++)
        {
                cout << a[i] << endl ;
        }
}
```

```
_____
The outputted array is;
2
3
4
5
6
Press any key to continue
```

Arrays

```
#include <iostream>
using namespace::std;
const int ArrSize = 4 ;

void main(void)
{
        int a[ArrSize];
        for(int i = 0 ; i<ArrSize ; i++)
        {
                a[i]=i*ArrSize ;
        }
        cout << a[3] ;
}
```

```
12
```

Arrays

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


void main(void)
{
	float a[4] = {30};
	cout << a[0] << endl;
}
```

```
30
```

Arrays

```
#include <iostream>
using namespace::std;
const int GoGo = 0 ;

void main(void)
{
	float a[GoGo] = {30};
	cout << a[0] << endl;
}
```

```
Compiler error, can't allocate any array of size 0
```

Arrays

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


void main(void)
{
	float a[4] = {30. , 32 , 33.4 , 23.0 };
	cout << a[0] << endl;
}
```

```
30
```

```
but with warning of truncation from double to int
```

Arrays

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


void main(void)
{
	int x = 44. ;
}
```

```
Compile & Run
```

But with warning for converting from double to int ...
Possible loss of data

Now (You should know why no…

Arrays

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


void main(void)
{
        float a[4] = {30. , 32 , 33.4 , 23.0 } ;
        int x[2] = {34,0};
        a[x[1]]=3;
        for (int i = 0 ; i < 4 ; i++)
                cout << a[i] << endl;
}
```

```
3
32
33.4
23
Press any key to continue . . .
```

Arrays

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

void main(void)
{
        float a[4] = {30. , 32 , 33.4 , 23.0 } ;
        float x[2] = {34,0};
        a[x[1]]=3;
        for (int i = 0 ; i < 4 ; i++)
                cout << a[i] << endl;
}
```

```
Compiler error ... Index is not of an integral type
```

Strings!

Strings using Arrays

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

void main(void)
{
    string str = "I AM A STRING!";
    system("pause");
}
```

Strings using Arrays

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

void main(void)
{
    string str = "I AM A STRING!";
    cout << str << endl;
    system("pause");
}
```

```
I AM A STRING!

Press any key to continue
```

Strings using Arrays

- String :
  - It's an ***array*** of ***characters***
- All strings end with `null ('\0')`

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


void main(void)
{
        char str1 [] = "Hello" ;
        // null character implicitly added at the end
        char str2 [] = {'H','e','l','l','o','\0'};
        // null character explicitly added
}
```

Strings using Arrays

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


void main(void)
{
char str1 [] = "hello" ;
char str2 [] = {'h','e','l','l','o','\0'};

        for (int i = 0 ; i < 6 ; i++)
        {
                cout << int(str1[i])
                << endl;
        }

        cout << "----------" << endl ;

        for (int i = 0 ; i < 6 ; i++)
        {
                cout << int(str2[i])
                << endl;
        }

}
```

```
104
101
108
108
111
0
----------
104
101
108
108
111
0
Press any key to continue
```

Strings using Arrays

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


void main(void)
{
        char str1 [] = "hello" ;
        char str2 [] = {'h','e','l','l','o','\0'};
        (str1[0]==str2[0]) ? cout << "Yuppy !" : cout << "No No :(" ;
}
```

```
Yuppy !
```

Strings using Arrays

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


void main(void)
{
        char str1 [] = "hello" ;
        char str2 [] = {'h','e','l','l','o','\0'};
        (str1==str2) ? cout << "Yuppy !" : cout << "No No :(" ;
}
```

```
No No :(
```

Strings using Arrays

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


void main(void)
{
        char str1 [] = "hello" ;
        char str2 [] = {'h','e','l','l','o','\0'};
        (str1==str2) ? cout << "Yuppy !" : cout << "No No :(" ;
}
```

```
No No :(
```

Why it's a no?!

Strings using Arrays

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


void main(void)
{
        char str1 [] = "hello" ;
        char str2 [] = {'h','e','l','l','o','\0'};
        (str1==str2) ? cout << "Yuppy !" : cout << "No No :(" ;
}
```

```
No No :(
```

Why it's a no?!

Cos we are comparing addresses and not values!

Strings using Arrays

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


void main(void)
{
        char str1 [] = "hello" ;
        char str2 [] = {'h','e','l','l','o','\0'};
        (str1==str2) ? cout << "Yuppy !" : cout << "No No :(" ;
        cout << "\n" ;

        for (int i = 0 ; i < 6 ; i++)
        {
                cout << int(str1[i]) << endl;
        }

        cout << "------------" << endl ;

        for (int i = 0 ; i < 6 ; i++)
        {
                cout << int(str2[i]) << endl;
        }
}
```

```
No No :(
104
101
108
108
111
0
------------
104
101
108
108
111
0
```

Strings using Arrays

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


void main(void)
{
        char str1 [] = "hello" ;
        char str2 [] = {'h','e','l','l','o','\0'};
        (str1==str2) ? cout << "Yuppy !" : cout << "No No :(" ;
        cout << "\n" ;

        for (int i = 0 ; i < 6 ; i++)
        {
                cout << str1[i] ;
        }

        cout << "\n------------" << endl

        for (int i = 0 ; i < 6 ; i++)
        {
                cout <<str2[i] ;
        }
        cout << endl ;

}
```

```
No No :(
hello
------------
hello
Press any key to
continue
```

Strings using Arrays

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


void main(void)
{
        char str1 [10] ;
        cin >> str1 ;

        cout << "The inputed string as a whole ! , is " << str1 << endl ;


}
```

```
HeHello
The inputed string as a whole ! , is HeHello
Press any key to continue
```

Strings using Arrays

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


void main(void)
{
        char str1 [10] ;
        cin >> str1 ;

        cout << "The inputed string as a whole ! , is " << str1 << endl ;


}
```

```
Ceeeeeeeeeeeeeeeeeeeeeeeeeeeee
The inputed string as a whole ! , is Ceeeeeeeeeeeeeeeeeeeeeeeeeeeee
Press any key to continue . . .
```

Compiler and print but with Run-time error !!! Out of bound !

Strings using Arrays

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


void main(void)
{
        char str1 [10] ;
        cin >> str1 ;

        cout << "The inputed string as a whole ! , is " << str1 << endl ;


}
```

```
Name Ed
The inputed string as a whole ! , is Name
Press any key to continue . . .
```

Stops at first white space character

Strings using Arrays

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


void main(void)
{
        char strl [10] ;
        cin >> strl ;

        cout << "The inputed string as a whole ! , is " << strl << endl ;


}
```

Printing the string "Array" like this is just for the strings
No other type of arrays can be printed like that

Strings using Arrays

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

void main(void)
{
    int intArr [10] = {0,1,1,12,34};

    cout << "The inputed string as a whole ! , is " << intArr << endl ;
    system("pause");
}
```

```
The inputed string as a whole ! , is 0043F958
Press any key to continue . . .
```

When printing the array it print it's location on memory by default

Strings using Arrays

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

void main(void)
{
    int intArr [10] = {0,1,1,12,34};

    cout << "The inputed string as a whole ! , is " << intArr << endl ;
    system("pause");
}
```

```
The inputed string as a whole ! , is 0047F850
Press any key to continue . . .
```

When printing the array it print it's location on memory by default

Just the char array (char charArr [])
Don't do that!
It prints the char contained in it!

Strings using Arrays

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

void main(void)
{
    char charArr [10] = "Hello!";
    cout << "The inputed string as a whole ! , is "
    << charArr << endl ;
    system("pause");
}
```

```
The inputed string as a whole ! , is Hello!
Press any key to continue . . .
```

When printing the array it print it's location on memory by default

Just the char array (char charArr [])
Don't do that!
It prints the char contained in it!

Strings using Arrays

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


void main(void)
{
    int str1 [10] ;
    cin >> str1 ;

    cout << "The inputed string as a whole ! , is " << str1 << endl ;
    system("pause");
}
```

Compiler error **(cin >> str1 ;)** ***, it's not a char array!***
U can't input it as a whole!

Strings using Arrays

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

void main(void)
{
    int intArr [10] = {0,1,1,12,34};

    cout << "The inputed string as a whole ! , is " << intArr << endl ;
    system("pause");
}
```

```
The inputed string as a whole ! , is 0043F998
Press any key to continue . . .
```

Strings using Arrays – static !

- Let's see this first …

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


void AutoArr()
{
        int Arr1 [3] = {1,2,3} ;
        cout << "Auto Arr" << endl ;
        for (int i=0 ; i<3 ; i++)
        {
                cout << "Arr1[" << i << "]=" <<
Arr1[i] << endl;
        }

}


void main(void)
{
        AutoArr() ;
}
```

```
Auto Arr
Arr1[0]=1
Arr1[1]=2
Arr1[2]=3
```

Strings using Arrays – static !

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

void AutoArr()
{
        int Arr1 [3] = {1,2,3} ;
        cout << "Auto Arr" << endl ;
        for (int i=0 ; i<3 ; i++)
        {
                cout << "Arr1[" << i << "]=" << Arr1[i] << endl;
        }

}


void main(void)
{
        AutoArr ;
}
```

Strings using Arrays – static I

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


void AutoArr()
{
int Arrl [3] = {1,2,3} ;
cout << "Auto Arr : " << endl ;
for (int i=0 ; i<3 ; i++)
{
        cout << "Arrl[" << i << "]="
        << Arrl[i]<< endl;
}
cout << "Auto Arr after +2 to all elemets :"
        <<endl;
for (int i=0 ; i<3 ; i++)
{
        cout << "Arrl[" << i << "]=" <<
        (Arrl[i]=Arrl[i]+2) << endl;
}

}

void main(void)
{
        AutoArr() ;
}
```

```
Auto Arr :
Arrl[0]=1
Arrl[1]=2
Arrl[2]=3
Auto Arr after +2 to all elemets :
Arrl[0]=3
Arrl[1]=4
Arrl[2]=5
Press any key to continue
```

Strings using Arrays – static I

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


void AutoArr()
{
int Arrl [3] = {1,2,3} ;
cout << "Auto Arr : " << endl ;
for (int i=0 ; i<3 ; i++)
{
        cout << "Arrl[" << i << "]="
        << Arrl[i]<< endl;
}
cout << "Auto Arr after +2 to all elemets :"
        <<endl;
for (int i=0 ; i<3 ; i++)
{
        cout << "Arrl[" << i << "]=" <<
        (Arrl[i]+=2) << endl;
}

}

void main(void)
{
        AutoArr() ;
}
```

```
Auto Arr :
Arrl[0]=1
Arrl[1]=2
Arrl[2]=3
Auto Arr after +2 to all elemets :
Arrl[0]=3
Arrl[1]=4
Arrl[2]=5
Press any key to continue
```

Strings using Arrays – static I

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


void AutoArr()
{
int Arrl [3] = {1,2,3} ;
cout << "Auto Arr : " << endl ;
for (int i=0 ; i<3 ; i++)
{
        cout << "Arrl[" << i << "]="
        << Arrl[i]<< endl;
}
cout << "Auto Arr after +2 to all elemets :"
        <<endl;
for (int i=0 ; i<3 ; i++)
{
        cout << "Arrl[" << i << "]=" <<
        (Arrl[i]+=2) << endl;
}

}

void main(void)
{
        AutoArr();
        AutoArr();
}
```

```
Auto Arr :
Arrl[0]=1
Arrl[1]=2
Arrl[2]=3
Auto Arr after +2 to all elemets :
Arrl[0]=3
Arrl[1]=4
Arrl[2]=5
Auto Arr :
Arrl[0]=1
Arrl[1]=2
Arrl[2]=3
Auto Arr after +2 to all elemets :
Arrl[0]=3
Arrl[1]=4
Arrl[2]=5
Press any key to continue
```

Strings using Arrays – static I

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


void AutoArr()
{
int Arrl [3] = {1,2,3} ;
cout << "Auto Arr : " << endl ;
for (int i=0 ; i<3 ; i++)
{
        cout << "Arrl[" << i << "]="
        << Arrl[i]<< endl;
}
cout << "Auto Arr after +2 to all elemets :"
        <<endl;
for (int i=0 ; i<3 ; i++)
{
        cout << "Arrl[" << i << "]=" <<
        (Arrl[i]=+2) << endl;
}

}

void main(void)
{
        AutoArr() ;
}
```

```
Auto Arr :
Arrl[0]=1
Arrl[1]=2
Arrl[2]=3
Auto Arr after +2 to all elemets :
Arrl[0]=3
Arrl[1]=4
Arrl[2]=5
Press any key to continue
```

Strings using Arrays – static

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


void StaticArr()
{
static int Arrl [3] = {1,2,3} ;
cout << "Arr : " << endl ;
for (int i=0 ; i<3 ; i++)
        {
cout << "Arrl[" << i << "]=" << Arrl[i] << endl;
        }
cout << "Arr after +2 to all elemets :" <<endl;
for (int i=0 ; i<3 ; i++)
        {
cout << "Arrl[" << i << "]=" << (Arrl[i]+2) <<endl;
        }
}

void main(void)
{
        StaticArr() ;
        StaticArr();
}
```

```
Arr :
Arrl[0]=1
Arrl[1]=2
Arrl[2]=3
Arr after +2 to all elemets :
Arrl[0]=3
Arrl[1]=4
Arrl[2]=5
Arr :
Arrl[0]=3
Arrl[1]=4
Arrl[2]=5
Arr after +2 to all elemets :
Arrl[0]=5
Arrl[1]=6
Arrl[2]=7
Press any key to continue . . .
```

Strings using Arrays – static

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


void StaticArr()
{
static int Arrl [3] = {1,2,3} ;
cout << "Arr : " << endl ;
for (int i=0 ; i<3 ; i++)
        {
cout << "Arrl[" << i << "]=" << Arrl[i] << endl;
        }
cout << "Arr after +2 to all elemets :" <<endl;
for (int i=0 ; i<3 ; i++)
        {
cout << "Arrl[" << i << "]=" << (Arrl[i]+2) <<endl;
        }
}

void main(void)
{
        StaticArr() ;
        StaticArr();
}
```

```
Arr :
Arrl[0]=1
Arrl[1]=2
Arrl[2]=3
Arr after +2 to all elemets :
Arrl[0]=3
Arrl[1]=4
Arrl[2]=5
Arr
Arrl[0]=3
Arrl[1]=4
Arrl[2]=5
Arr after +2 to all elemets :
Arrl[0]=5
Arrl[1]=6
Arrl[2]=7
Press any key to continue . . .
```

Arrays – passing to functions

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

void PassingArr(int a [])
{
        cout << a[0] << endl;
}

void main(void)
{
        int Arr [] = {3,2};
        PassingArr(Arr);
}
```

```
3
```

Arrays

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

void PassingArr(int a [])
{
        cout << a[0] << endl;
}

void main(void)
{
        int Arr [] = {3,2};
        PassingArr(Arr[]);
}
```

```
Compiler error ... Passing the array with subscript []
```

Arrays

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

void PassingArrVal(int a[])
{
	a[0]+=2 ;
}


void main(void)
{
	int Arr [] = {3,2};
	PassingArrVal(Arr);
	cout << Arr[0] << endl ;
	PassingArrVal(Arr);
	cout << Arr[0] << endl ;
}
```

Passing arrays by value is passing it with reference not by value! ....

Arrays

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

void PassingArrVal(int a[])
{
	a[0]+=2 ;
}
void PassingArrRef1(int &a[])
{
	a[0]+=2 ;
}
void main(void)
{
	int Arr [] = {3,2};
	PassingArrVal(Arr);
	cout << Arr[0] << endl ;
	PassingArrRef(Arr);
	cout << Arr[0] << endl ;
}
```

Compiler error .. can't pass arrays with values , it's referenced all by it self

Arrays

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

void PassingArr(const int a [])
{
	cout << a[0] << endl;
}
void main(void)
{
	int Arr[2] = {1,3} ;
	PassingArr(Arr);
}
```

Arrays

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

void PassingArr(const int a [])
{
	a[0]+=2 ;
}

void main(void)
{
	int Arr [2] ;
	PassingArr(Arr);
	cout << Arr[0] << endl ;
}
```

Compiler error .. can't modify const arrays

Multiple subscripted Arrays

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

void main(void)
{
        int Arr[3][4] ;  // 3 rows, 4 columns
}
```

Multiple subscripted Arrays

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

void main(void)
{
        int Arr[2,3] ;
}
```

```
Compiler error .. Should be [][]
```

Multiple subscripted Arrays

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

void main(void)
{
        int Arr [][] ;
}
```

```
Compiler error : missing subscript size
```

Multiple subscripted Arrays

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

void main(void)
{
        int Arr [][3] ;
}
```

```
Compiler error : missing subscript size
```

Multiple subscripted Arrays

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

void main(void)
{
        int Arr[2][3] = { {2,3,5} , {4,1,3} } ;
                        //      row#1     row#2
}
```

2	3	5
4	1	3

Multiple subscripted Arrays

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

void main(void)
{
        int Arr[2][3] = { {2,3} , {4,1,3} } ;
                        //      row#1   row#2
}
```

2	3	0
4	1	3

Multiple subscripted Arrays

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

void main(void)
{
        int Arr[2][3] = { {2} , {4,3} } ;
}
```

2	0	0
4	3	0

Multiple subscripted Arrays

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

void main(void)
{
        int Arr[2][3] = { {2,3,5} , {4,3} } ;
        cout << Arr[2][3] << endl ;
}
```

[illegible]

Not a compiler error ... but the number in that location in memory

Multiple subscripted Arrays

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

void main(void)
{
        int Arr[2][3] = { {2,3,5} , {4,3} } ;
        cout << Arr[1][2] << endl ;
}
```

```
4
```

Multiple subscripted Arrays passing to function

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

void ArrFun(int A [2][3])
{
        cout << A [1][1] ;
}


void main(void)
{
        int Arr[2][3] = { {2,3,5} , {4,3} } ;
        ArrFun(Arr);
}
```

```
3
```

Multiple subscripted Arrays passing to function

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

void ArrFun(int A [2][3])
{
        cout << A [1][1] ;
}


void main(void)
{
        int Arr[2][3] = { {2,3,5} , {4,3} } ;
        ArrFun(Arr[2][3]);
}
```

```
Compiler error ... 2||...
```

Multiple subscripted Arrays passing to function

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

void ArrFun(int A [][3])
{
        cout << A [1][1] ;
}


void main(void)
{
        int Arr[2][3] = { {2,3,5} , {4,3} } ;
        ArrFun(Arr);
}
```

```
3
```

No compiler error for missing the first subscript in the function's prototype ...
like single-scripted arrays

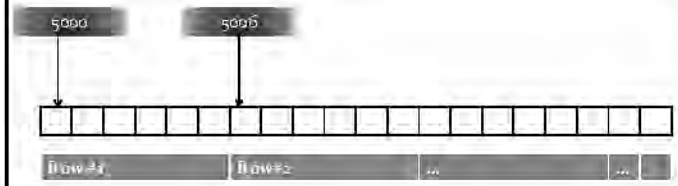
Multiple subscripted Arrays passing to function

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

void ArrFun(int A [][])
{
        cout << A [1][1] ;
}


void main(void)
{
        int Arr[2][3] = { (2,3,5) , (4,3) } ;
        ArrFun(Arr);
}
```

Compiler error ... missing the second subscript in the function's prototype

Multiple subscripted Arrays Memory storage

```
#include <iostream>
using namespace::std;
const int Rows = 2 ;
const int Columns = 3 ;

void main(void)
{
        int Arr[Rows][Columns] ;
}
```



Strings - Structs

Strings - Structs

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

void main(void)
{
        char c ;
        while ( (c = cin.get()) != EOF)
        {
                cout << "c = " << c << endl ;
        }
}
```

- cin.get()
  - Returns one character from stram "even white space"
  - Returns EOF is end-of-file encountered

- EOF
  - indicate end of input
    - ctrl+z IBM
    - ctrl+d Unix-Mac
- cin.eof
  - Returns 1 (True) if EOF occured

Strings - Structs

```
#include <iostream>
#include <stdio.h>
using namespace::std;

void main(void)
{
        char c ; // qwegwe        qwerty
        while ( (c = cin.get()) != EOF)
        {
                cout << "c = " << c << endl ;
        }
        if (cin.eof())
        {
                cout << "We reached the end of input " << endl ;
        }
}
```

Strings - Structs

```
qwegwe
c = q
c = w
c = e
c = g
□ = w
c = e
c =

qwerty
□ = q
c = w
□ = e
c = r
c = t
□ = y
c =

^Z
We reached the end of input
Press any key to continue . . .
```

Strings - Structs

```
#include <iostream>
#include <stdio.h>
using namespace::std;

void main(void)
{
        char c ;        // qwegwe   qwerty
        while ( (c = cin.get()) != EOF)
        {
                cout << "c = " << c << endl ;
        }
        if (cin.eof() == 1 )
        {
                cout << "We reached the end of input " << endl ;
        }
}
```

Strings - Structs

```
qwegwe
c = q
c = w
c = e
c = g
c = w
c = e
c =

qwerty
c = q
c = w
c = e
c = r
c = t
c = y
c =

^Z
We reached the end of input
Press any key to continue . . .
```

Strings - Structs

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

void main(void)
{
        char c ;        // qweqwe   qwerty
        while ( (c = cin.get()) != EOF)
        {
                cout << "c = " << c << endl ;
        }
        if (cin.eof() = 1 )
        {
                cout << "We reached the end of input " << endl ;
        }
}
```

```
Compiler error: == not =
```

Strings - Structs

```
#include <iostream>

using namespace::std;

void main(void)
{
        char c = 1 ;
        cout << c << endl ;
        c = -1 ;
        cout << c << endl ;
        c = 3 ;
        cout << c << endl ;
        c= 67;
        cout << c << endl ;
        c= 68;
        cout << c << endl ;
        c = -10 ;
        cout << c << endl ;
        c= - 324234;
        cout << c << endl ;
}
```

Strings - Structs

- Character handling library
  - <cctype>
    - Functions to perform tests and manipulations on characters
- As long as char doesn't allow negative values
  - EOF usually has value -1

Strings - Structs

isalnum	Check if character is alphanumeric
isalpha	Check if character is alphabetic
iscntrl	Check if character is a control character
isdigit	Check if character is decimal digit
isgraph	Check if character has graphical representation using locale
islower	Check if character is lowercase letter
isprint	Check if character is printable
ispunct	Check if character is a punctuation character
isspace	Check if character is a white-space
isupper	Check if character is uppercase letter
isxdigit	Check if character is hexadecimal digit

And secondly, two functions to convert between letter cases:

tolower	Convert uppercase letter to lowercase
toupper	Convert lowercase letter to uppercase

Fundamentals of Strings

Fundamentals of Strings

- A string :
  - Array of characters , end with null "\0"
  - An empty string "null string"
    - Contains no characters , represented as ""
      - Not a single space !
- Length of string is the number of characters in it
- String is a constant pointer \ pointer to const ?
  - It's surely a constant pointer
    - Pointer to string's first element "character" .

Fundamentals of Strings

```
#include <iostream>
#include <cctype>

using namespace::std;

void main(void)
{
        char Str1 [] = "Smile" ;
        char Str2 [] = {'S' , 'm' , 'i' , 'l' , 'e' , '\0'};
        if (Str1 == Str2)
        {
                cout << "Yeeeaaah !" << endl ;
        }
        else
        {
                cout << "Nopeeee ! " << endl ;
        }
}
```

```
Nopeeee !
Press any key to continue
```

So , how to compare strings ?

Fundamentals of Strings

- To compare strings ... it's like comparing the arrays you already know !!!!

```
#include <iostream>
#include <cctype>

using namespace::std;

void main(void)
{
        bool match = true ;
        char Str1 [] = "Smile" ;
        char Str2 [] = {'S' , 'm' , 'i' , 'l' , 'e' , '\0'};
        for (int i = 0 ; i<6 ; i++)
        {
                if(Str1[i] != Str2[i])
                {
                        match = false ;
                }
        }
        match == 1 ? cout << "Yes" : cout << "No" ;
        cout << endl;
}
```

```
Yes
```

Fundamentals of Strings

```
#include <iostream>
#include <cctype>

using namespace::std;

void main(void)
{
        bool match = true ;
        char Str1 [] = "Smile" ;
        char Str2 [] = {'S' , 'm' , 'i' , 'l' , 'e' , '\0'};
        for (int i = 0 ; i<7 ; i++)
        {
                if(Str1[i] != Str2[i])
                {
                        match = false ;
                }
        }
        match == 1 ? cout << "Yes" : cout << "No" ;
        cout << endl;
}
```

```
No
```

Fundamentals of Strings

- Better way of thinking ?
  - Sure , using `while`

```
#include <iostream>
#include <cctype>

using namespace::std;

void main(void)
{       bool match = true ;
        int i = 0 ;
        char Str1 [] = "Smile" ;
        char Str2 [] = {'S' , 'm' , 'i' , 'l' , 'e' , '\0'};
        while ((match == true) && (i<6))
        {
                if(Str1[i] != Str2[i])
                {
                        match = false ;
                }
                i++;
        }
        match == 1 ? cout << "Yes" : cout << "No" ; cout <<"\n";
```

```
Yes
```

Fundamentals of Strings

```
#include <iostream>
#include <cctype>

using namespace::std;

void main(void)
{
        bool match = true ;
        int i = 0 ;
        char Str1 [] = "Smile" ;
        char Str2 [] = {'S' , 'm' , 'i' , 'l' , 'e' };
        while ((match == true) && (i<6))
        {
                if(Str1[i] != Str2[i])
                {
                        match = false ;
                }
                i++;
        }
        match == 1 ? cout << "Yes" : cout << "No" ;
        cout << endl;
}
```

```
No
```

Fundamentals of Strings

```
#include <iostream>
#include <cctype>

using namespace::std;

void main(void)
{
        bool match = true ;
        int i = 0 ;
        char Str1 [] = "Smile" ;
        char Str2 [] = {'S' , 'm' , 'i' , 'l' , 'e' , '\0'};
        while (match == true)
        {
                if(Str1[i] != Str2[i])
                {
                        match = false ;
                }
                i++;
        }
        match == 1 ? cout << "Yes" : cout << "No" ;
        cout << endl;
}
```

```
No
```

Fundamentals of Strings

```
#include <iostream>
#include <cctype>

using namespace::std;

void main(void)
{
        bool match = true ;
        int i = 0 ;
        char Str1 [] = "Smile" ;
        char Str2 [] = ('S' , 'm' , 'i' , 'l' , 'e' , ) ;
        while ((match = true) && (i<6))
        {
                if(Str1[i] != Str2[i])
                {
                        match = false ;
                }
                i++;
        }
        match == 1 ? cout << "Yes" : cout << "No" ;
        cout << endl;
}
```

```
Yes
```

Fundamentals of Strings

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

void main(void)
{
        char TwoDimStr[4][4] = {"SoSo" , "hee", "GoGo"};
}
```

```
Compiler error... why ?
```

```
Array bounds over flow
```

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

void main(void)
{
        char TwoDimStr[4][5] = {"SoSo" , "hee", "GoGo"};
}
```

Fundamentals of Strings

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

void main(void)
{
        char TwoDimStr[][5] = {"SoSo" , "hee", "GoGo"};
}
```

```
Compile and run...
```

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

void main(void)
{
        char TwoDimStr[][] = {"SoSo" , "hee", "GoGo"};
}
```

```
Compiler error
```

Fundamentals of Strings

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

void main(void)
{
        char TwoDimStr[3][5] = {"SoSo" , "hee", "GoGo"} ;
        for (int i = 0 ; i<3  ; i++)
        {
                cout << TwoDimStr[i] ;
        }
        for (int i = 0 ; i<3  ; i++)
        {
                cout << TwoDimStr[i] <<endl;
        }
}
```

```
SoSoheeGoGoSoSo
hee
GoGo
Press any key to continue
```

Fundamentals of Strings

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

void main(void)
{
        char TwoDimStr[3][5] = ("SoSo" , "hee") ;
        TwoDimStr[1] = "LaLa" ;
        TwoDimStr[2] = "Pooo";
        for (int i = 0 ; i<3  ; i++)
        {
                cout << TwoDimStr[i] ;
        }
        for (int i = 0 ; i<3  ; i++)
        {
                cout << TwoDimStr[i] <<endl;
        }
}
```

Compiler error

So how can we do it?

Fundamentals of Strings

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

void Inserter ( int RowIndex , char sl [5] , char Arr [][5])
{
        for (int i = 0 ; i<5 ; i++)
        {
                Arr[RowIndex][i]=sl[i];
        }
}

void main(void)
{
        char TwoDimStr[3][5] = ("SoSo" , "hee") ;
        char TempStr1[] = "LaLa" ;
        char TempStr2[] = "Pooo" ;
        cout << "Before Inserter" << endl ;
        for (int i = 0 ; i<3  ; i++)
        {       cout << TwoDimStr[i] <<endl;       }
        Inserter (1,TempStr1,TwoDimStr);
        Inserter (2,TempStr2,TwoDimStr);
        cout << "After Inserter" << endl ;

        for (int i = 0 ; i<3  ; i++)
        {       cout << TwoDimStr[i] <<endl;       }
}
```

```
Before Inserter
SoSo
hee

After Inserter
SoSo
LaLa
Pooo
```

Fundamentals of Strings

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

void Inserter ( int RowIndex , char sl [5] , char Arr [][5])
{
        for (int i = 0 ; i<5 ; i++)
        {
                Arr[RowIndex][i]=sl[i];
        }
}

void main(void)
{
        char TwoDimStr[3][5] = ("SoSo" , "hee") ;
        char TempStr1[] = "LaLa" ;
        char TempStr2[] = "Poooo0000" ;
        cout << "Before Inserter" << endl ;
        for (int i = 0 ; i<3  ; i++)
        {       cout << TwoDimStr[i] <<endl;       }
        Inserter (1,TempStr1,TwoDimStr);
        Inserter (2,TempStr2,TwoDimStr);
        cout << "After Inserter" << endl ;

        for (int i = 0 ; i<3  ; i++)
        {       cout << TwoDimStr[i] <<endl;       }
}
```

```
Before Inserter
SoSo
hee

After Inserter
SoSo
LaLa
Poooo0000
```

Fundamentals of Strings

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

void Inserter ( int RowIndex , char sl [5] , char Arr [][5])
{
        for (int i = 0 ; i<5 ; i++)
        {
                Arr[RowIndex][i]=sl[i];
        }
}

void main(void)
{
        char TwoDimStr[3][5] = ("SoSo" , "Poooo0000") ;
        char TempStr1[] = "LaLa" ;

        cout << "Before Inserter" << endl ;
        for (int i = 0 ; i<3  ; i++)
        {       cout << TwoDimStr[i] <<endl;}
        Inserter (1,TempStr1,TwoDimStr);
        cout << "After Inserter" << endl ;
        for (int i = 0 ; i<3  ; i++)
        {       cout << TwoDimStr[i] <<endl;}
}
```

Compiler error

Fundamentals of Strings

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

void Inserter ( int RowIndex , char sl [5] , char Arr [][5])
{
        for (int i = 0 ; i<5 ; i++)
        {
                Arr[RowIndex][i]=sl[i];
        }
}

void main(void)
{
        char TwoDimStr[3][5] = {"SoSo" , "bee"} ;
        char TempStr1[] = "LaLa" ;
        char TempStr2[] = {'I' , 'v'} ;
        cout << "Before Inserter" << endl ;
        for (int i = 0 ; i<3  ; i++)
        {       cout << TwoDimStr[i] <<endl;      }
        Inserter (1,TempStr1,TwoDimStr);
        Inserter (2,TempStr2,TwoDimStr);
        cout << "After Inserter" << endl ;

        for (int i = 0 ; i<3  ; i++)
        {       cout << TwoDimStr[i] <<endl;      }
}
```

```
Before Inserter
SoSo
bee

After Inserter
SoSo
LaLa
Iv
```

Fundamentals of Strings

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

void Inserter ( int RowIndex , char sl [5] , char Arr
[][5])
{
        for (int i = 0 ; i<5 ; i++)
        {
                Arr[RowIndex][i]=sl[i];
        }
}

void main(void)
{       char TwoDimStr[3][5] = {"SoSo" , "bee"} ;
        char TempStr1[] = {'I','v'};
        char TempStr2[] = "LaLa" ;
        cout << "Before Inserter" << endl ;
        for (int i = 0 ; i<3  ; i++)
        {       cout << TwoDimStr[i] <<endl;      }
        Inserter (1,TempStr1,TwoDimStr);
        Inserter (2,TempStr2,TwoDimStr);
        cout << "After Inserter" << endl ;

        for (int i = 0 ; i<3  ; i++)
        {                cout << TwoDimStr[i] ;    }
        cout << endl;}
```

```
Before Inserter
SoSo
bee

After Inserter
SoSo
bee
```

Fundamentals of Strings

- Input Stream ...
  - get (CharArray , Size , Delimiter)
  - *1st parameter :*
    - Pointer to a constant array
  - *2nd parameter :*
    - Maximum number of characters to read
      - Reads until Size-1 or delimiter encountered (default EOF)
  - *3rd parameter :*
    - Termination character , default '\n'
- Note :
  - If the termination character is reached before the maximum number of characters is read ...
    - a null is written
    - And the delimiter is not placed in the character array

Fundamentals of Strings

- One unfortunate side effect of the get() function is that it leaves unused characters in the input stream
  - Remain waiting in the input stream
- A subsequent call to get() retrieves the next "unused" characters , whether or not that retrieval is intended

Fundamentals of Strings

```
#include <iostream>
using namespace::std;
const int Length = 256 ;

void main(void)
{

        char str1[Length] ;
        char str2[Length] ;

        cout << "Enter String " << endl ;
        cin.get(str1,Length);     // i wanna go
        cout << "str1 = " << str1 << endl ;


}
```

```
Enter String
i wanna go
str1 = i wanna go
Press any key to continue . . .
```

Fundamentals of Strings

```
#include <iostream>
using namespace::std;
const int Length = 256 ;

void main(void)
{

        char str1[Length] ;
        char str2[Length] ;

        cout << "Enter String " << endl ;
        cin.get(str1,Length);     // i wanna go
        cout << "str1 = " << str1 << endl ;

        cout << "Enter String " << endl ;
        cin.get(str2,Length);
        cout << "str2 = " << str2 << endl ;

}
```

```
Enter String
i wanna go
str1 = i wanna go
Enter String
str2 =
Press any key to continue . . .
```

Fundamentals of Strings

```
#include <iostream>
using namespace::std;
const int Length = 256 ;

void main(void)
{

        char str1[Length] ;
        char str2[Length] ;

        cout << "Enter String " << endl ;
        cin.get(str1,Length);     // i wanna go
        cout << "str1 = " << str1 << endl ;

        cout << "Enter String " << endl ;
        cin >> str2;      // he wants to go
        cout << "str2 = " << str2 << endl ;

}
```

```
Enter String
i wanna go
str1 = i wanna go
Enter String
he wants to go
str2 = he
Press any key to continue . . .
```

Fundamentals of Strings

```
#include <iostream>
using namespace::std;
const int Length = 256 ;
void main(void)
{       char str1[Length] ;
        char str2[Length] ;
        char str3[Length] ;

        cout << "Enter String " << endl ;
        cin.get(str1,Length);     // i wanna go
        cout << "str1 = " << str1 << endl ;

        cout << "Enter String " << endl ;
        cin >> str2;      // he wants to go
        cout << "str2 = " << str2 << endl ;

        cout << "Enter String " << endl ;
        cin >> str3;      // she wants to go
        cout << "str3 = " << str3 << endl ;}
```

```
Enter String
i wanna go
str1 = i wanna go
Enter String
he wants to go
str2 = he
Enter String
she wants to go
str3 = she
Press any key to continue . . .
```

Fundamentals of Strings

```
#include <iostream>
using namespace::std;
const int Length = 256 ;
void main(void)
{       char str1[Length] ;
        char str2[Length] ;

        cout << "Enter String " << endl ;
        cin.get(str1,Length);     // i wanna go
        cout << "str1 = " << str1 << endl ;

        cout << "Enter String " << endl ;
        cin >> str2;      // he wants to go
        cout << "str2 = " << str2 << endl ;

        cout << "Enter String " << endl ;
        cin >> str2;      // she wants to go
        cout << "str2 = " << str2 << endl ;}
```

```
Enter String
i wanna go
str1 = i wanna go
Enter String
he wants to go
str2 = he
Enter String
str2 = wants
Press any key to continue
```

Fundamentals of Strings

```
#include <iostream>
using namespace::std;
const int Length = 256 ;

void main(void)
{       char str1[Length] ;
        char str2[Length] ;
        char str3[Length] ;

        cout << "Enter String " << endl ;
        cin.get(str1,Length);     // i wanna go
        cout << "str1 = " << str1 << endl ;

        cout << "Enter String " << endl ;
        cin >> str2;              // he wants to go
        cout << "str2 = " << str2 << endl ;

        cout << "Enter String " << endl ;
        cin.get(str3,Length);     // she wants to go
        cout << "str3 = " << str3 << endl ; }
```

```
Enter String
i wanna go
str1 = i wanna go
Enter String
he wants to go
str2 = he
Enter String
str3 = wants to go
Press any key to continue . . .
```

Fundamentals of Strings

```
#include <iostream>
using namespace::std;
const int Length = 256 ;

void main(void)
{       char str1[Length] ;
        char str2[Length] ;
        char str3[Length] ;

        cout << "Enter String " << endl ;
        cin.get(str1,Length);  // i wanna go
        cout << "str1 = " << str1 << endl ;

        cout << "Enter String " << endl ;
        cin.get(str2,Length);  // he wants to go
        cout << "str2 = " << str2 << endl ;}
```

```
Enter String
i wanna go
str1 = i wanna go
Enter String
str2 =
Press any key to continue . . .
```

Fundamentals of Strings

```
#include <iostream>
using namespace::std;
const int Length = 256 ;

void main(void)
{       char str1[Length] ;
        char str2[Length] ;
        char str3[Length] ;

        cout << "Enter String " << endl ;
        cin>>str1;        // i wanna go
        cout << "str1 = " << str1 << endl ;

        cout << "Enter String " << endl ;
        cin.get(str2,Length);     // he wants to go
        cout << "str2 = " << str2 << endl ; }
```

```
Enter String
i wanna go
str1 = i
Enter String
str2 =  wanna go
Press any key to continue
```

Fundamentals of Strings

```
#include <iostream>
using namespace::std;
const int Length = 256 ;

void main(void)
{       char str1[Length] ;
        char str2[Length] ;
        char str3[Length] ;

        cout << "Enter String " << endl ;
        cin.get(str1,Length);     // i Wanna go
        cout << "str1 = " << str1 << endl ;

        cout << "Enter String " << endl ;
        cin>>str2;        // We Wants to go
        cout << "str2 = " << str2 << endl ;}
```

```
Enter String
I Wanna go
str1 = I Wanna go
Enter String
We Wants to go
str2 = We
Press any key to continue . . .
```

Fundamentals of Strings

```
#include <iostream>
using namespace::std;
const int Length = 30 ;

void main(void)
{
        int num ;
        char str1[Length] ;
        char str2[Length] ;
        char str3[Length] ;

        cout << "Enter String " << endl ;
        cin >> num ;                    // 30
        cin.get(str1 , Length , '\n');

        cout << "str1 = " << str1 << endl ;
}
```

```
Enter String
30
str1 =
Press any key to continue . . .
```

Rule:
When numeric values are input, the following string is not processed.

Fundamentals of Strings

```
#include <iostream>
using namespace::std;
const int Length = 30 ;

void main(void)
{
        int num ;
        char str1[Length] ;
        char str2[Length] ;
        char str3[Length] ;

        cout << "Enter String " << endl ;
        cin >> num ;                    // 30
        cin.get(str1 , Length , '\n');

        cout << "str1 = " << str1 << endl ;
}
```

```
Enter String
30
str1 =
Press any key to continue . . .
```

Rule:
When numeric values are input, the following string is not processed.

Fundamentals of Strings

```
#include <iostream>
using namespace::std;
const int Length = 30 ;

void main(void)
{
        int num ;
        char str1[Length] ;
        char str2[Length] ;
        char str3[Length] ;

        cout << "Enter String " << endl ;
        cin >> num ;            // 30
        cin.get();
        cin.get(str1 , Length , '\n');// Wewe

        cout << "str1 = " << str1 << endl ;}
```

```
Enter String
30
Wewe
str1 = Wewe
Press any key to continue . . .
```

Solutions:
Putting a "cin.get();" between string inputs

Fundamentals of Strings

```
#include <iostream>
using namespace::std;
const int Length = 80 ;

void main(void)
{
        int num ;
        char str1[Length] ;
        char str2[Length] ;
        char str3[Length] ;

        cout << "Enter String " << endl ;
        cin >> num ;                        // 30 sad
        cin.get(str1 , Length , '\n');

        cout << "str1 = " << str1 << endl ;
}
```

```
Enter String
30 sad
str1 =  sad
Press any key to continue
```

Fundamentals of Strings

```
#include <iostream>
using namespace::std;
const int Length = 30 ;

void main(void)
{
        int num ;
        char str1[Length] ;
        char str2[Length] ;
        char str3[Length] ;

        cout << "Enter String " << endl ;
        cin >> num ;                        // 30sad
        cin.get(str1 , Length , '\n');

        cout << "str1 = " << str1 << endl ;
}
```

```
Enter String
30sad
str1 =  sad
Press any key to continue . . .
```

Fundamentals of Strings

```
#include <iostream>
using namespace::std;     const int Length = 80 ;

void main(void)
{       char c ;
        char str1[Length] ;
        char str2[Length] ;
        char str3[Length] ;

        cout << "Enter String " << endl ;

        cin.get(str1 , Length) ;
        cin.get() ;

        cout << "Enter a char " << endl ;
        cin.get(c);

        cout << "str1 = " << str1 << endl ;
        cout << "c = " << c << endl ; }
```

```
Enter String
i am happy
Enter a char
d
str1 = i am happy
c = d
Press any key to continue . . .
```

Fundamentals of Strings

```
#include <iostream>
using namespace::std;     const int Length = 30 ;

void main(void)
{       char c ;
        char str1[Length] ;
        char str2[Length] ;
        char str3[Length] ;

        cout << "Enter String " << endl ;

        cin.get(str1 , Length) ;

        cout << "Enter a char " << endl ;
        cin.get(c);

        cout << "str1 = " << str1 << endl ;
        cout << "c = " << c << endl ; }
```

```
Enter String
i am happy
Enter a char
str1 = i am happy
c =

Press any key to continue . . .
```

Fundamentals of Strings

```
#include <iostream>
using namespace::std;     const int Length = 80 ;

void main(void)
{       char c ;
        char str1[Length] ;
        char str2[Length] ;
        char str3[Length] ;

        cout << "Enter String " << endl ;

        cin.get(str1 , Length) ;

        cout << "Enter a char " << endl ;
        cin.get(c);

        cout << "str1 = " << str1 << endl ;
        cout << "c = " << c << endl ; }
```

```
Enter String
i wanna go and play ! :D
Enter a char
str1 = i wanna g
c = a
Press any key to continue
```

Fundamentals of Strings

```
#include <iostream>
using namespace::std;     const int Length = 30 ;

void main(void)
{       char c ;
        char str1[Length] ;
        char str2[Length] ;
        char str3[Length] ;

        cout << "Enter String " << endl ;

        cin.get(str1 , Length) ;
        cin.get();

        cout << "Enter a char " << endl ;
        cin.get(c);

        cout << "str1 = " << str1 << endl ;
        cout << "c = " << c << endl ; }
```

```
Enter String
i wanna go and play :D
Enter a char
str1 = i wanna g
c = a
Press any key to continue
```

So it's impossible to guess how many `cin.get()` **are needed to consume all potintional additional letters**

Fundamentals of Strings

```
#include <iostream>
using namespace::std;
const int Length = 10 ;

void main(void)
{
        char c ;
        char str1[Length] ;
        char str2[Length] ;
        char str3[Length] ;

        cout << "Enter String " << endl ;

        cin.get(str1 , Length) ;
        cin.get();      cin.get();      cin.get();      cin.get();

        cout << "Enter a char " << endl ;
        cin.get(c);

        cout << "str1 = " << str1 << endl ;
        cout << "c = " << c << endl ;
}
```

```
Enter String
i wanna go and play :D
Enter a char
str1 = i wanna g
c = a
Press any key to continue
```

Fundamentals of Strings

```
#include <iostream>
using namespace::std;
const int Length = 256;

void main(void)
{
        char c ;
        char str1[Length] ;
        cout << "Enter String " << endl ;
        cin.get(str1 , Length) ;
        cin.get();      cin.get();      cin.get();      cin.get();
        cout << "Enter a char " << endl ;
        cin.get(c);
        cout << "str1 = " << str1 << endl ;
        cout << "c = " << c << endl ;
}
```

```
Enter String
i wanna go and play :D


Enter a char
a
str1 = i wanna g
c = a
Press any key to continue
```

Fundamentals of Strings

- Solution ...
  - `ignore()`
  - `ignore(int Length=1 ,char c = 'n');`
    - ***Extract*** and ***discard*** characters from stream (default = 1)
    - 1st parameter:
      - Maximum number of characters to ignore
    - 2nd parameter
      - Termination character
    - Stop discarding once delimiter found
      - Default (EOF)

```
#include <iostream>
using namespace::std;
const int Length = 50;
void main(void)
{               char c ;
        char str1[Length] ;
        char str2[Length] ;
        char str3[Length] ;

        cout << "Enter String " << endl ;

        cin.get(str1 , Length) ;// i wanna go and eat now !
        cin.ignore();

        cout << "Enter a char " << endl ;
        cin.get(c);
        cout << "str1 = " << str1 << endl ;
        cout << "c = " << c << endl ; }
```

```
Enter String
i wanna go and eat now !
Enter a char
n
str1 = i wanna go and eat now !
c = n
Press any key to continue
```

```
#include <iostream>
using namespace::std;
const int Length = 50;
void main(void)
{               char c ;
        char str1[Length] ;
        char str2[Length] ;
        char str3[Length] ;

        cout << "Enter String " << endl ;

        cin.get(str1 , Length) ;// i wanna go and eat now !
        cin.ignore(100,'\n');

        cout << "Enter a char " << endl ;
        cin.get(c);
        cout << "str1 = " << str1 << endl ;
        cout << "c = " << c << endl ; }
```

```
Enter String
i wanna go and eat now !
Enter a char
i
str1 = i wanna go and eat now !
c = i
Press any key to continue
```

Fundamentals of Strings

- `cin.getline(Arr , Size , delimiter);`
  - Reads line of text
  - Copies input into specified array , till
    - One less than size is reached (size-1)
    - Delimiter character is input
    - Removes delimiter from input stream
    - Puts `null` at the end of the array

Fundamentals of Strings

```
#include <iostream>
using namespace::std;
const int Length = 50;

void main(void)
{
        int IWannaEat ;
        char str1[Length] ;


        cout << "Enter String :" << endl ;
        cin.getline(str1 , 80 ,'\n') ;
        cout << "str1 = " << str1 << endl ;

        cout << "Enter a num :" << endl ;
        cin >> IWannaEat ;

        cout << "str1 = " << str1 << endl ;
        cout << "IWannaEat = " << IWannaEat << endl ;
}
```

```
Enter String :
I am happy
str1 = I am happy
Enter a num :
23
str1 = I am happy
IWannaEat = 23
Press any key to continue
```

Fundamentals of Strings

- String function conversion
  - Converting , searching , comparsion
  - <cstlib>

Some of the functions

- **String conversion:**
  atofConvert string to double (function)
- **atoi**Convert string to integer (function)
- **atol**Convert string to long integer (function)
- **strtod**Convert string to double (function)
- **strtol**Convert string to long integer (function)
- **strtoul**Convert string to unsigned long integer (function)

C++ data types

Structs

Structs

- Differentiate :
  - Array :
    - Is a collection of variables of the *same* type
  - Struct "Structure"
    - Is a collection of variable of *one or more* type

Structs

- Struct is definition not a declaration
  - No memory allocation
    - "Memory is allocated just when we declare a variable"
- Examples :
  - Student record , banks , players , addresses

Structs

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


void main(void)
{
        struct MyFirstS
        {
                int key ;
                float Salary ;
                char Name[20];
        };
}
```

Structs

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


void main(void)
{
        struct MyFirstS
        {
                int key ;
                float Salary ;
                char Name[20];
        }
}
```

```
Compiler error : Missing ; after defining the struct
```

Structs

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


Void main(void)
{
        struct MyInner
        {
                int Key ;
                int Salary;
        };

        struct MyFirstS
        {
                MyInner info ;
                char Name[20];
        };
}
```

Structs

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


void main(void)
{
        int i = 0;
        struct MyFirstS
        {
                int a , b , c ;
        } TempStruct [2] = {{2,4,5} , {1,6,8}};

        cout << TempStruct[i].a << endl ;
        cout << TempStruct[i].b << endl ;
        cout << TempStruct[i].c << endl ;
}
```

```
2
4
5
Press any key to continue . . .
```

Structs

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


void main(void)
{
        int i = 0;
        struct MyFirstS
        {
                int a , b , c ;
        } TempStruct [2] = {{2,4} , {1,6,8}};

        cout << TempStruct[i].a << endl ;
        cout << TempStruct[i].b << endl ;
        cout << TempStruct[i].c << endl ;
}
```

```
2
4
0
Press any key to continue . . .
```

Structs

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


void main(void)
{
        int i = 1;
        struct MyFirstS
        {
                int a , b , c ;
        } TempStruct [2] = {{2,4} };

        cout << TempStruct[i].a << endl ;
        cout << TempStruct[i].b << endl ;
        cout << TempStruct[i].c << endl ;
}
```

```
0
0
0
Press any key to continue . . .
```

Structs

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


void main(void)
{
        int i = 1;
        struct MyFirstS
        {
                int a , b , c ;
        } ;
        MyFirstS TempStruct [2] = {{2,4} };

        cout << TempStruct[i].a << endl ;
        cout << TempStruct[i].b << endl ;
        cout << TempStruct[i].c << endl ;
}
```

```
0
0
0
Press any key to continue
```

Structs

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


void main(void)
{
        int i = 1;
        struct MyFirstS
        {
                int a , b , c ;
        } TempStruct [2] = {{2,4} }

        cout << TempStruct[i].a << endl ;
        cout << TempStruct[i].b << endl ;
        cout << TempStruct[i].c << endl ;
}
```

```
Compiled error, missing semi colon before cout ';' (After TempStruct)
```

Structs

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


void main(void)
{
        int i = 0;
        struct
        {
                int a , b , c ;
        } TempStruct [2] = {{2,8} };

        cout << TempStruct[i].a << endl ;
        cout << TempStruct[i].b << endl ;
        cout << TempStruct[i].c << endl ;
}
```

```
2
8
0
Press any key to continue . . .
```

Structs

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


void main(void)
{
        int i = 0;
        struct
        {
                int a , b , c ;
        } ;
        TempStruct [2] = {{2,8} };

        cout << TempStruct[i].a << endl ;
        cout << TempStruct[i].b << endl ;
        cout << TempStruct[i].c << endl ;
}
```

Compiler Error: When not writing the name of the struct we should declare variable immediately

Structs

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


void main(void)
{
        int i = -1;
        struct
        {
                int a , b , c ;
        } TempStruct [2] = {{2,8} };
        cout << TempStruct[i].a << endl ;
        cout << TempStruct[i].b << endl ;
        cout << TempStruct[i].c << endl ;
}
```

```
Press any key to continue
```

Structs

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


void main(void)
{
        int i = -1;
        struct
        {
                int a , b , c ;
        } TempStruct [2] = {{2,8} };
        TempStruct[i].c = 67 ;
        cout << TempStruct[i].a << endl ;
        cout << TempStruct[i].b << endl ;
        cout << TempStruct[i].c << endl ;
}
```

```
Press any key to continue . . .
```

Structs

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


void main(void)
{
        int i = 1;
        struct
        {
                int a , b , c ;
        } TempStruct [2] = {{2,8} };
        TempStruct[i].c = 67 ;
        cout << TempStruct[i].a << endl ;
        cout << TempStruct[i].b << endl ;
        cout << TempStruct[i].c << endl ;
}
```

```
0
0
67
Press any key to continue . . .
```

Structs

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


void main(void)
{
        int i = 1;
        struct
        {
                int a , b , c ;
        } TempStruct [2] = {{2,8} };
        TempStruct.c = 67 ;
        cout << TempStruct[i].a << endl ;
        cout << TempStruct[i].b << endl ;
        cout << TempStruct[i].c << endl ;
}
```

Compiler Error

Structs

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


void main(void)
{
        int i = 1;
        struct
        {
                int a , b , c ;
        } TempStruct [2] = {{2,8} };
        TempStruct[i].c = 67 ;
        cout << TempStruct << endl  ;
}
```

0013F048
Press any key to continue

Structs

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


void main(void)
{
        struct student
        {
                char Name[20] ;
                int Num ;
                int id ;
        };
        student st1 = {"Mohammad" , 196 , 7};
}
```

Compile and run

Structs

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


void main(void)
{
        struct student
        {
                char Name[20] ;
                int Num ;
                int id ;
        };
        student st1 = {"Mohammad" , 196 , 7};
        st1 = {"ZeeZee" , 111 , 4};
}
```

Compiler error

Just in definition time

Structs

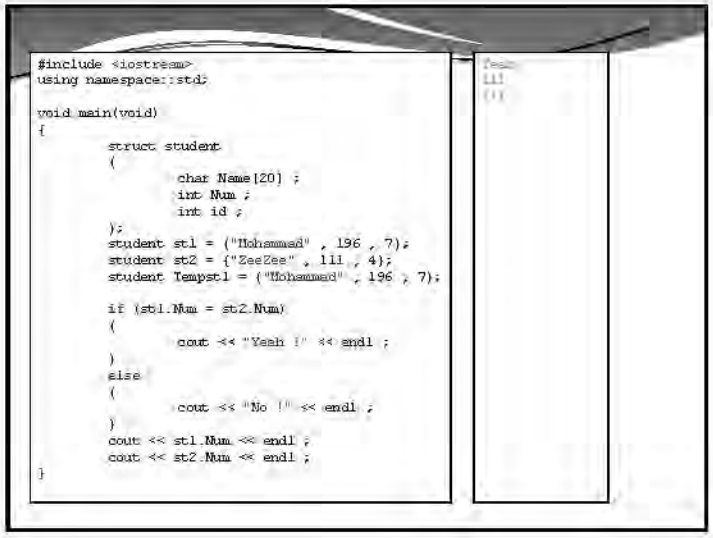
```
#include <iostream>
using namespace::std;


void main(void)
{
        struct student
        {
                char Name[20] ;
                int Num ;
                int id ;
        };
        student st1 = {"Mohammad" , 196 , 7};
        student st2 = {"ZeeZee" , 111 , 4};
        student Tempst1 = {"Mohammad" , 196 , 7};

        if (st1 == Tempst1)
        {
                cout << "Yeah !" << endl ;
        }
        else
        {
                cout << "No !" << endl ;
        }
}
```

Structs

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


void main(void)
{
        struct student
        {
                char Name[20] ;
                int Num ;
                int id ;
        };
        student st1 = {"Mohammad" , 196 , 7};
        student st2 = {"ZeeZee" , 111 , 4};
        student Tempst1 = {"Mohammad" , 196 , 7};

        if (st1 == Tempst1)
        {
                cout << "Yeah !" << endl ;
        }
        else
        {
                cout << "No !" << endl ;
        }
}
```

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


void main(void)
{
        struct student
        {
                char Name[20] ;
                int Num ;
                int id ;
        };
        student st1 = {"Mohammad" , 196 , 7};
        student st2 = {"ZeeZee" , 111 , 4};
        student Tempst1 = {"Mohammad" , 196 , 7};

        if (st1.Name == Tempst1.Name)
        {
                cout << "Yeah !" << endl ;
        }
        else
        {
                cout << "No !" << endl ;
        }
}
```

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


void main(void)
{
        struct student
        {
                char Name[20] ;
                int Num ;
                int id ;
        };
        student st1 = {"Mohammad" , 196 , 7};
        student st2 = {"ZeeZee" , 111 , 4};
        student Tempst1 = {"Mohammad" , 196 , 7};

        if (st1.Num == Tempst1.Num)
        {
                cout << "Yeah !" << endl ;
        }
        else
        {
                cout << "No !" << endl ;
        }
}
```



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

void main(void)
{
        struct student
        {
                char Name[20] ;
                int Num ;
                int id ;
        };
        student st1 = ("Mohammad" , 196 , 7);
        student st2 = {"ZeeZee" , 111 , 4};
        student Tempst1 = ("Mohammad" , 196 , 7);

        if (st1.Num = st2.Num)
        {
                cout << "Yeah !" << endl ;
        }
        else
        {
                cout << "No !" << endl ;
        }
        cout << st1.Num << endl ;
        cout << st2.Num << endl ;
}
```

```
Yeah !
111
196
```