

PROGRAMMING

LECTURE # 2

Flowchart

Flowchart

نستخدم مجموعة من الأشكال الرمزية الاصطلاحية المبوبة في الجدول التالي :

الرمز	العنوان - الترجمة	الوصف
START	خط طيفي (Start)	خط طيفي أو تمهيد (Start) يفتح باب الالغاز.
STOP	(Stop)	خط طيفي (Stop) ينهي باب الالغاز.
PRINT	(Print)	عصبة حسابية (Print).
INPUT	(Input)	اجماع / ادخال / اخراج (Input) ارسال اوصاف / اخراج معلومات من المتصفح.
DEC	Decision (Decision)	اخذ القرار (Decision).
END	Row line (Vertical Line)	خط افقي (Horizontal Line).
FOR I = 1 To 10	Loop Control (Loop)	خط افقي (Horizontal Line).

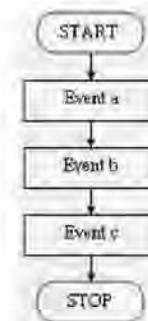
فوائد استخدام flowchart قبل كتابة البرنامج لمسألة ما :

١. تمكن المبرمج من الإلمام الكامل بالمسألة المراد حلها و السيطرة على كل أجزاءها بحيث تساعد في اكتشاف الأخطاء المتبقية (Logic Error) والتي تختبر من أهم الأخطاء التي تجده المبرمج.
٢. تساعد بيسر و سهولة على تعدل البرامج الموضوعة بمجرد النظر.
٣. يختبر الاحتياط برسوم خرائط سير العمليات لحلو 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 صرحت في خطوات واضحة بسيطة و مفهومة.
٤. توفير وسيلة ملائمة ومساعدة في كتابة البرامج ذات التفرعات الكثيرة.

يمكن تصنيف flowcharts بما يلي:

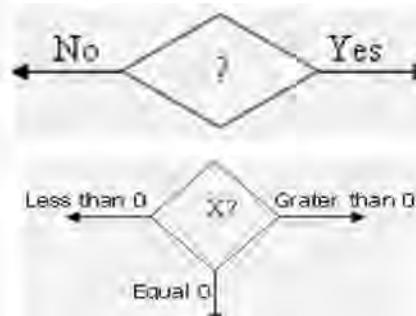
- * خرائط التتابع البسيط (Simple sequential Flowchart)
- * خرائط التفرع (Branched Flowchart)
- * خرائط الدوران البسيط (Loop Flowchart)
- * خرائط الدوران المتداخلة (Nested)

Simple sequential Flowchart



□ يخلو هذا النوع من
النفرعات Branches ..
و الدورانات loops ..

Branched Flowchart



ويحدث التفرع في
البرامح بسبب
الحاجة لاتخاذ قرار
أو مفضلة بين
اختيارات أو أكثر،
وهنالك أسلوبان في
تنفيذ القرار .

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

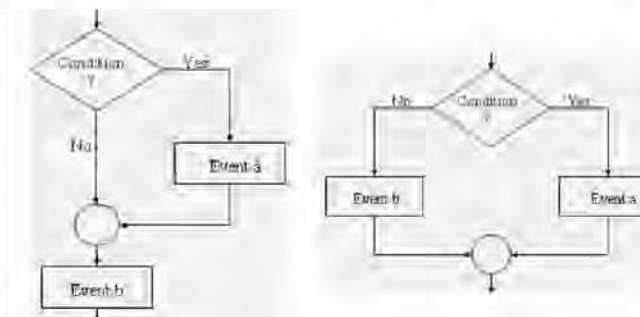


Area= ΠR^2
Circumference= $2\Pi R$
 $\Pi \approx 3.14$

Input: R
Output: R,A,C

- الخطوات :
- ١. ابدأ .
- ٢. اقرّ قيمة R .
- ٣. ضع قيمة PIE = 3,14 .
- ٤. احسب المساحة A من المعادلة A = (PIE)*R*R .
- ٥. احسب المحيط C من المعادلة C = 2*(PIE)*R .
- ٦. اطبع قيمة كل من C, A, R .
- ٧. توقف .

يشكل عام فلن Branched Flowchart يمكن أن تأخذ إحدى
الصورتين الآتتين :

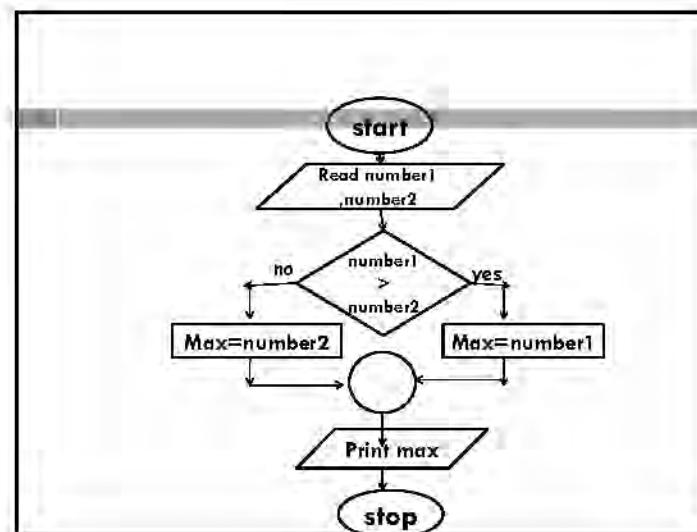


Example: compare between tow number and print the max :

Input : number1 ,number2
Output: max

الخطوات :

١. ابدأ
٢. ادخل العدد الاول number1 والعدد الثاني number2
٣. إذا كان number1 > number2 جعل max=number1 وإلا
- max=number2 لا اجعل
٤. اطبع max
٥. توقف



Nested

نحتاج إليها لإعادة عملية أو مجموعة من العمليات في البرنامج عدداً محدوداً أو غير محدود من المرات، ويكون الشكل العام لمثل هذه الخرائط كالتالي:

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

Input: R
Output: R, A

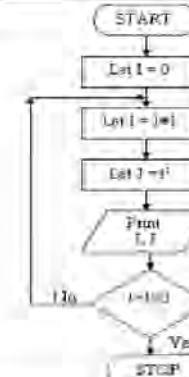
خطوات الحل :

١. ابدأ
٢. اقرأ صيف قطر الدائرة R
٣. أوجد مساحة الدائرة $A = 3.14 R^2$
٤. اطبع قيم كل من A, R
٥. هل هناك مزيد من الدوائر؟
- فإن كان نعم فعد إلى الخطوة (٢).
- وإن كان لا فعد إلى الخطوة (٤).
٦. توقف.

العداد : Counter

- كثير من الأحيان نحتاج في برامج الحاسوب الإلكتروني إلى العد Counting، فقد تريه مثلاً أن بعد عدد كل من الطلاب والطالبات ضمن الشعبة، وقد تكون هذه العملية سهلة للإنسان لأنها أصبحت ضمن قدراته الحركية التي يكتسبها من الطفولة، إلا أن الحاسوب يحتاج إلى تصميم خوارزمية للعد Counting Algorithm تتضمن خطوات معينة إذا أتيحت لها اسقاطها على العد.
- ويمكن تحديد الخطوات التي يتبعها الحاسوب حتى يمكن من العد في الخطوات الأساسية:
 ١. أجعل العدد مساوياً لـ ٠.
 ٢. أجعل 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: لم يتم إدخال أي مدخلات.

- خطوات الحل :
١. ابدأ.
 ٢. أجعل $I = 0$.
 ٣. أجعل $J = 1$.
 ٤. أطبع I, J .
 ٥. إذا كانت $I = 100$ = أذهب إلى الخطوة ٧ ولا أذهب إلى الخطوة ٣.
 ٦. توقف.

Summers Algorithm

المجاميع الإجمالية:

- في كثير من الأحيان نحتاج في برامج الحاسوب الإلكتروني إلى جمع مجموعة كبيرة من الأعداد التي تمثل معلومات ظاهرة معينة، فمثلاً قد نرغب في إيجاد الوسط الحسابي لأعمار طلاب الجامعة، ولتحقيق هذا أو لا يجب أن نحسب مجموع أعمار الطلاب، وطبعاً ليس عملياً إعطاء رمز أبيجي لكل عمر طالب فقد تحتاج لأكثر من عشرة الآلاف رمز، في مثل هذه الحالات نصمم خوارزمية معينة للتجميع تسمى خوارزمية التجميع summers Algorithm تتضمن خطوات محددة إذا أتبعها الحاسوب استطاع أن يجمع أي كمية من البيانات باستخدام متغيرين اثنين إحداهما هو المتغير الذي نجمعه والأخر هو الجمع الإجمالي (المجموع).

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

١. أجعل المجمع مساوياً الصفر.
٢. ادخل قيمة واحدة للمتغير.
٣. أجعل 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.

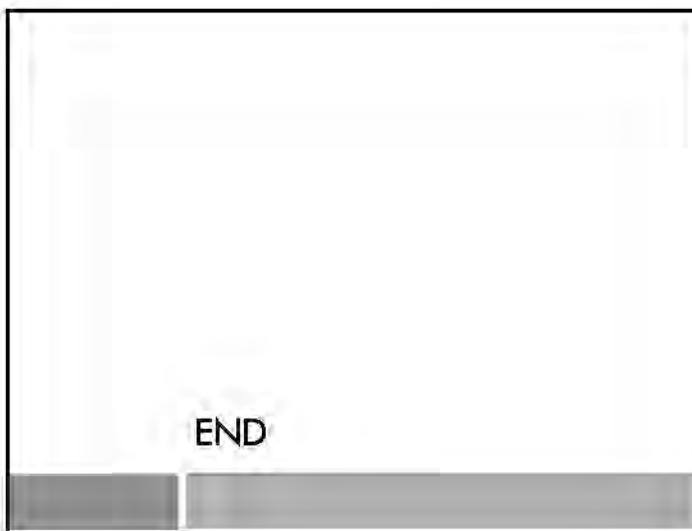
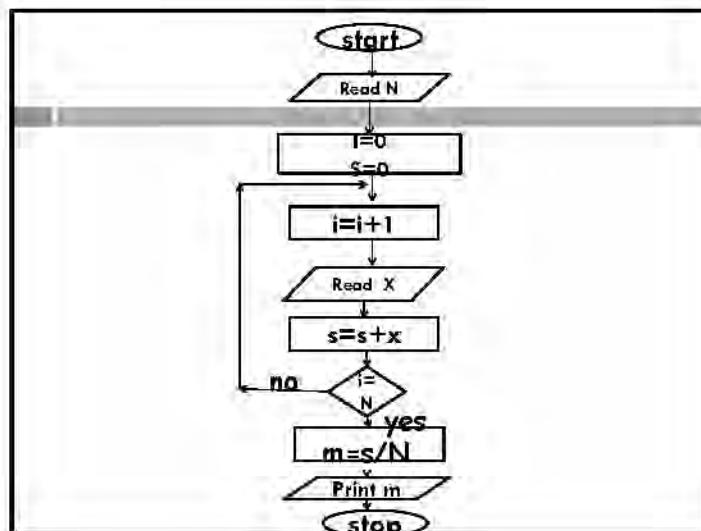
Example : Draw a flowchart to find the mean age of the student :

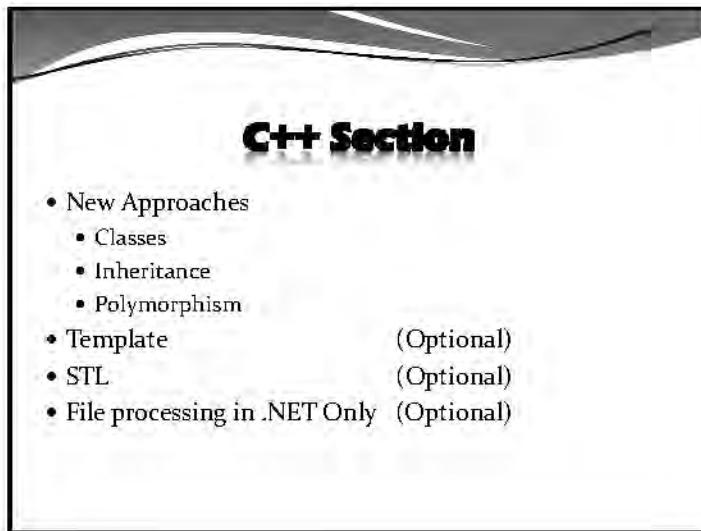
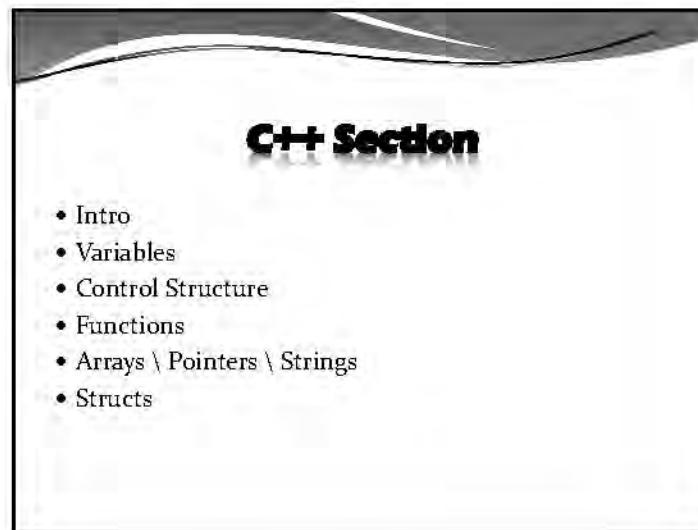
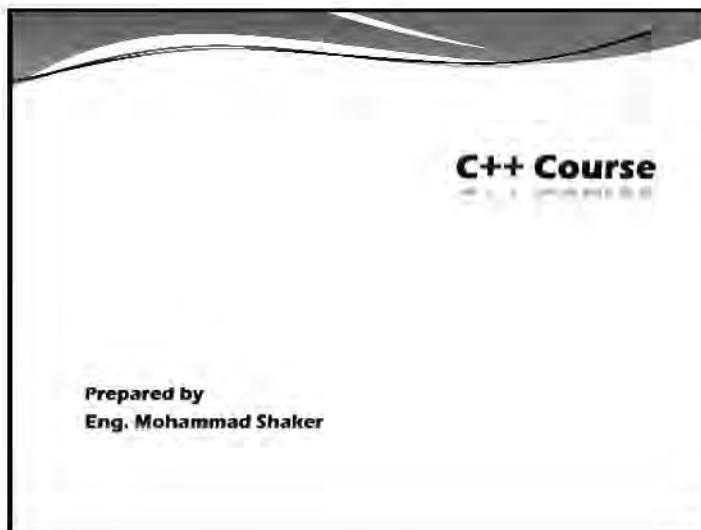
□ Solution :

- الحل: نفترض أن إجمالي عدد الطالب N
- ونستخدم عدداً لرقم كل طالب ونرمز له بالرمز X
- ونرمز لعمر الطالب بـ S
- ونستخدم مجموعاً لأعمار الطلبة ونرمز له بالرمز S
- ونستخدم الرمز m ليدل على معدل أعمار الطلبة.
- Input : N,X
- Output : m

خطوات الحل:

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





Position	Released	Debut in Windows	Programming Languages	Rating	Delta	Score
1	-2	1	Java	8.50%	+0.0%	A
2	1	1	C	7.79%	+0.0%	A
3	1	1	C++	5.83%	+0.0%	A
4	5	11	C#	5.04%	+0.0%	A
5	4	1	PHP	5.00%	+0.0%	A
6	5	1	Visual Basic	4.72%	+0.0%	A
7	10	11	JavaScript	4.27%	+0.0%	A
8	7	1	Python	3.68%	+0.0%	A
9	8	1	Perl	3.12%	+0.0%	A
10	20	11	VB	2.03%	+0.0%	A
11	15	1	JavaScript	1.90%	+0.0%	A
12	11	1	Fortran	1.48%	+0.0%	A
13	4	11	VisualBasic .NET	1.07%	+0.0%	A
14	16	11	LabVIEW	0.82%	+0.0%	A
15	16	1	Excel	0.73%	+0.0%	A
16			Assembly	0.67%		B
17	21	11	Mathematica	0.62%	+0.0%	B



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



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)
- 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 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; // indicate successful compilation
}
```

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 ;
}

Compile 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 !
}

Compile and run

#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

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

Address	Simple	Structured
Pointer	enum	Array
Reference	Integral int, Short, long, bool	Struct
	Floating float, double, long double	Union
		Class

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;

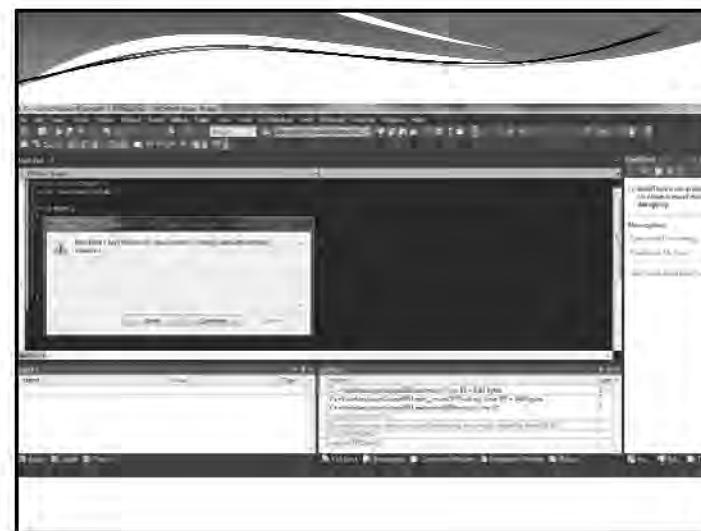
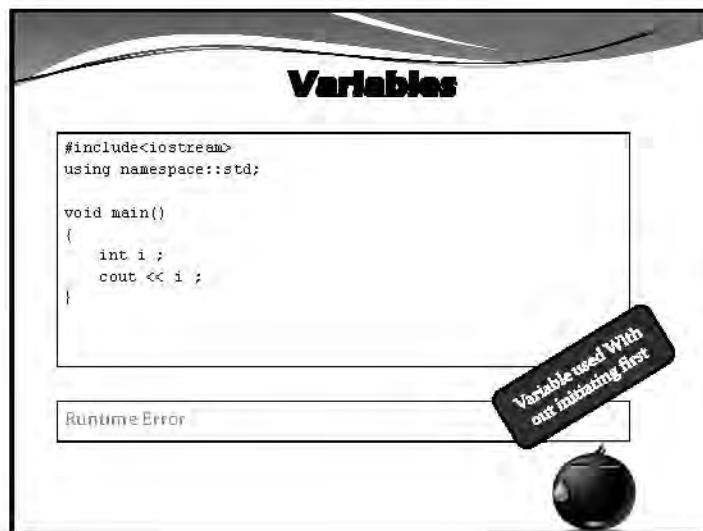
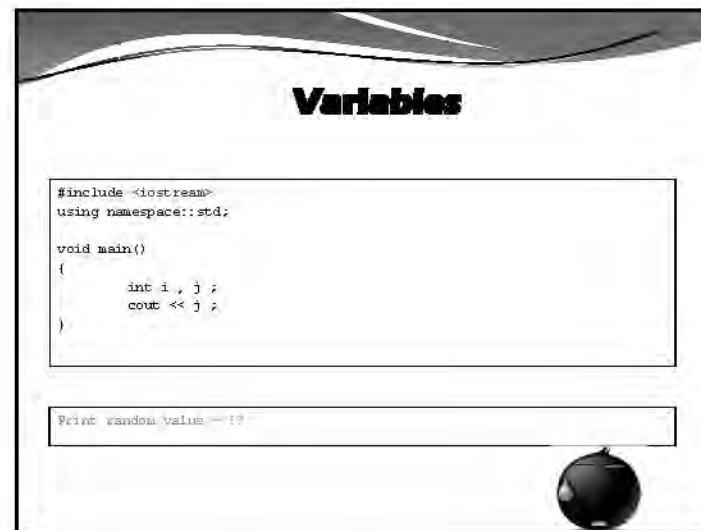
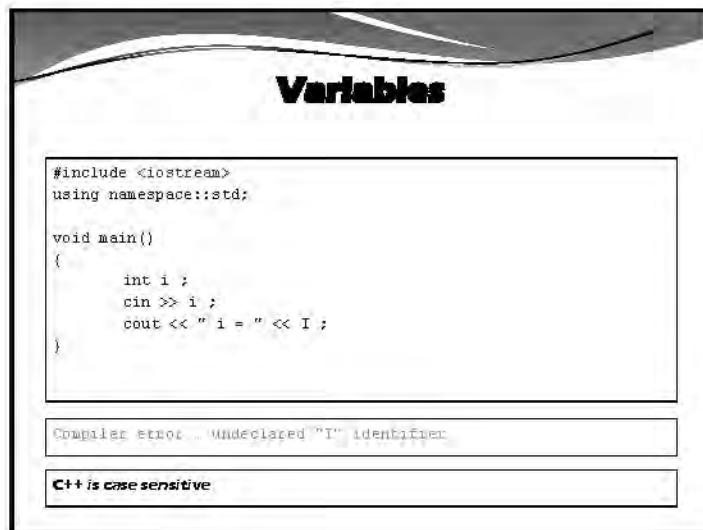
void main()
{
    int i , j ;
    i = 0 ;
    j = 4 ;
}
```

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 , 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 << "j+i = " << j+i << "\n" ;
    cout << "2*i = " << 2*i << "\n" ;
}
```

```
i
j
j+i= 5
2*i= 10
Press any key to continue...
```

Assignment operators

- Assignment between objects of the same type is always supported

Assignment operator	Sample expression	Explanation	Assigns
<i>Assume: int c = 3, d = 5, e = 4, f = 6, g = 12;</i>			
$+=$	$c += 7$	$c = c + 7$	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	Concept	Sample expression	Explanation
$++$	Prefix increment	$++a$	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.
$--$	Prefix decrement	$--b$	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

96

```

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 = 10
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;

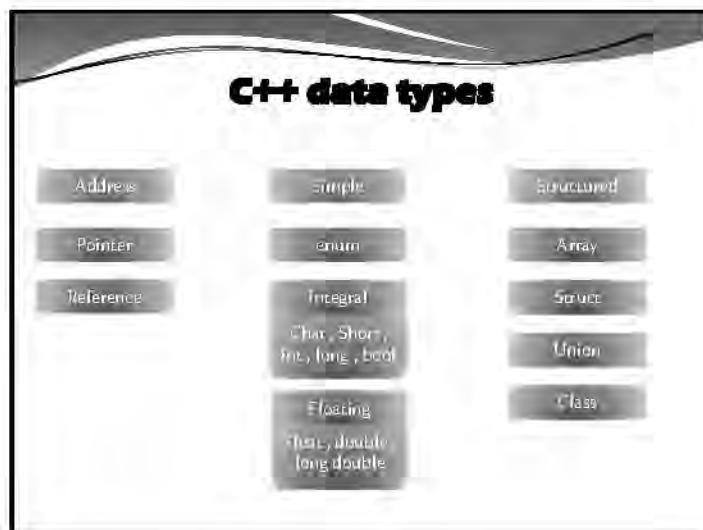
```

output # 2

```

x = 11
y = 11

```



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 ;
}
```

>>> Please type the code and run it

Note that:

- 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 ;
}
```

>>>

Note that:

- A arithmetic 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 << endl ;
}
```

Evaluate Compiler
I.E (3 <= 2)

Parsing Steps

- if (3 <= 2)
- b1 = false ;
- b2 = true ;
- else
- b1 = 52 ;

>>>

Note that:

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

so pls

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
- 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

- Jump up to else

↓ ← ↓

Note that:

- Arithmetic 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

↓ ← ↓

Note that:

- Arithmetic 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

↓ ← ↓

Note that:

- Arithmetic 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 <<

↓ ← ↓

Note that:

- Arithmetic 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 <<

Note that:

- Arithmetic 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

cout << b1 << " " << b2 <<

Note that:

- Arithmetic 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 <<
}
```

**Printing the integral type
in the form of number**

Evaluate Compiler
if (3 <= 2)

Parsing Steps

cout << b1 << " " << b2 <<

Note that:

- Arithmetic 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

cout << b1 << " " << b2 <<

Note that:

- Arithmetic 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)**

Printing Steps

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 = 0.8;
    cout << d << endl ;
    system("pause");
}
```

0

No need to print !!

double data type

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

void main()
{
    double d = .8;
    cout << d << endl;
    system("pause");
}
```

8



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 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 with it later ;)

*Remember that
char, bool are integral types!*

Data types: simple types size, range and precision

Type	Size	Range	Precision
short (also called short int)	2 bytes	-32,768 to 32,767	Not applicable
int	4 bytes	-2,147,483,648 to 2,147,483,647	Not applicable
long (also called long int)	4 bytes	-2,147,483,648 to 2,147,483,647	Not applicable
float	4 bytes	approximately -3.4e+38 to 3.4e+38	7 digits
double	8 bytes	approximately -1.7e+308 to 1.7e+308	15 digits

Data types: simple types size, range and precision

long double	variables	approximately 10 ⁻¹⁰⁰⁰ to 10 ¹⁰⁰⁰	19 digits
char	variable	All ASCII characters (can also be used as an integer type, although we do not recommend doing so.)	Not applicable
bool	variable	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 ;
}

// small example & 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 value

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 ;
}
```

Compiled error



Code Cracking

```
#include <iostream>

int main()
{
    int i,j ;
    return -0001223333 ;
}
```

Hello tman :D



Code Cracking

```
#include <iostream>

Int main()
{
    int i,j ;
    return -12323333 ;
}
```

Compiled error "Int" I is capital letter



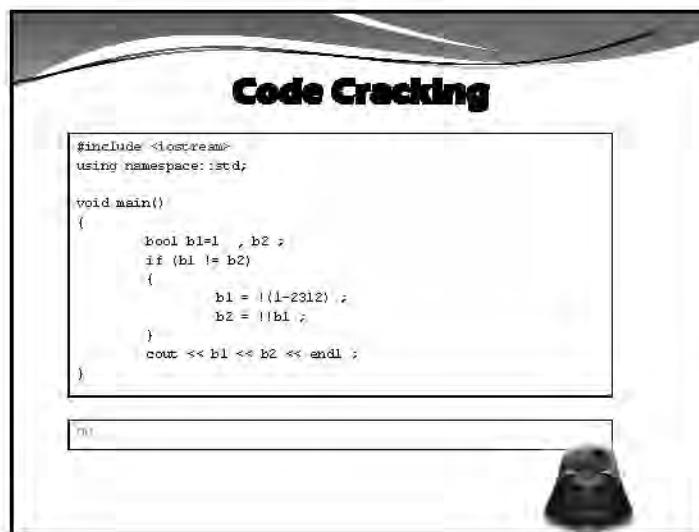
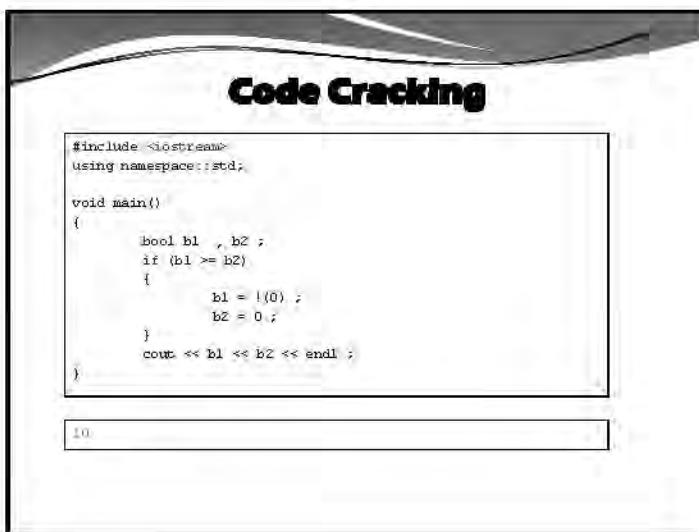
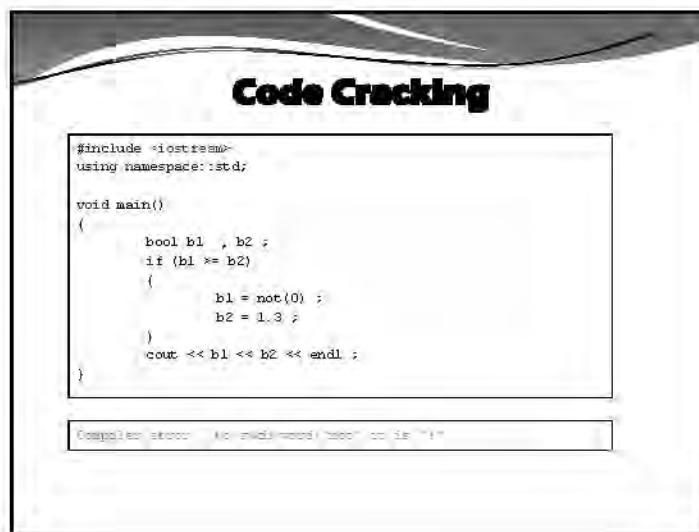
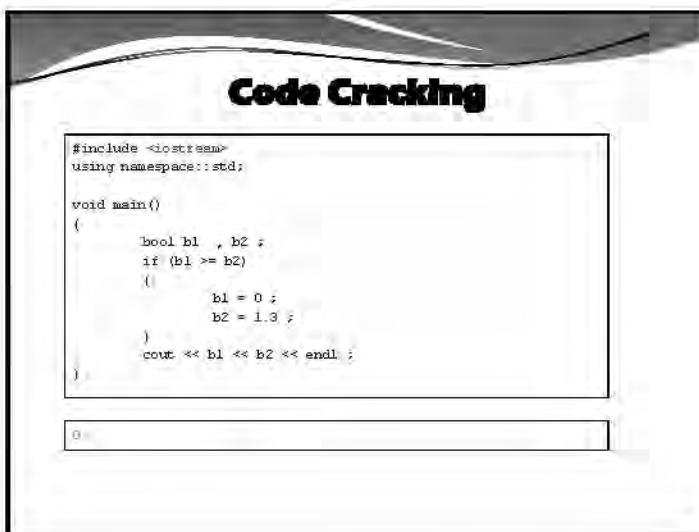
Code Cracking

```
#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 >= !(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 ?
02
```

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 ;
}
```

01

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



Operators

- Assign "="

```
* b = 5 ;           // a = 5
* a = 3 + (b = 6) ; // a = 8
* a = b = 5 ;       // legal a = b = 5
```

Arithmetic Operators

- Arithmetic Operation

C operators	C arithmetic operators	Algorithmic expression	C expression
Addition	+	$x + y$	$x + y$
Subtraction	-	$x - y$	$x - y$
Multiplication	*	$x * y$	$x * y$
Division	/	x / y	x / y
Modulus	%	$x \% y$	$x \% y$

- 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 ;
```

0

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	Evaluated second. If there are several, they are evaluated left to right.
/	Division	
%	Modulus	
+	Addition	Evaluated last. If there are several, they are evaluated left to right.
-	Subtraction	

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++ ?

$$z = p \text{ } r \% q + w / (x - y);$$

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:

Step 1: $y = 2 * 5 + 5 + 3 * 2 + 7; \quad \text{Multiplication}$
 $2 * 5 \text{ is } 10$

Step 2: $y = 10 + 5 + 1 + 5 + 7; \quad \text{Multiplication}$
 $10 + 5 \text{ is } 15$

Step 3: $y = 15 + 3 + 5 + 7; \quad \text{Finalization before addition}$
 $3 + 5 \text{ is } 13$

Step 4: $y = 15 + 13 + 7; \quad \text{Multiplication}$
 $15 + 13 \text{ is } 28$

Step 5: $y = 28 + 7; \quad \text{Last addition}$
 $28 + 7 \text{ is } 35$

Step 6: $y = 35$ (Assignment - value to y)

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:

- 10
- 4
- 324

Type Conversion

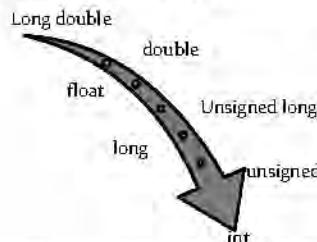
- Used when different data types are used together
 - Expression
 - Arithmetic
 - Relational
 - Assignment operations
 - Parameter passage (through functions)
 - Return type (From Value-Returning functions)

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 ; // x is promoted to float

3.4 + 'b' // 'b' (char) is promoted to float

3.43265L + 1b' // 1b' (long) 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 ;           // float(7.4) 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 explicit 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 ;        // explicit 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 cccc = 'c' ;
    int i ;
    i = int(cccc) ;           // explicit casting
    cout << i << endl ;
}
```

80

Type Conversion – Explicit Casting

- Or like that

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

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

80

Type Conversion – Explicit Casting

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

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

81

Type Conversion – Explicit Casting

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

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

81

Type Conversion – Explicit Casting

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

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

40

Type Conversion – Explicit Casting

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

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

41

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>(i) ;
    f = static_cast<float>(i) ;
}
```

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

No 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 warning encountered

Warnings should be as expected with the following commands

```
i1 = i2 ;
i1 = i3 ;
i2 = i3 ;
```

(Right)

Explicit Casting – static_cast

But in fact the warnings are like just :

```
i1 = i3 ;
i2 = i3 ;
```

Explicit Casting – static_cast

But in fact the warnings are for these:

```
i1 = i3 ;
i2 = i3 ;
```

Why?

C++ data types

Address	Simple	Structural
Pointer	enum	Array
Reference	Integral char, short, int, long, bool	Struct
	Floating float, double, long double	Union
		Class

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 is declared 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 IS 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 = 0

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 ;
}
```

0

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 ;
}
```

Compiler:

enum

- Can't assign int type to another enum ... fatte73yonak 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...
COMPILEERRRR!

Compiler:

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:

enum

- Can assign int type to another enum one but with using "type cast" ... fatte73yonak 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**

Compiler:

enum

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

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

Compiler error anyway cause of -- Mine++
```

Remember how compiler manipulate code...

1st: Mine++

2nd : Cars(Mine++)

3rd : Assignment ++

Check

enum

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

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

Compiler error anyway cause of -- mine++
```

Remember how compiler manipulate code...

1st: Mine++

2nd : Cars(Mine++)

3rd : Assignment ++

Check

enum

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

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

Compiler error anyway cause of -- Mine++
```

Remember how compiler manipulate code...

1st: Mine++

2nd : Cars(Mine++)

3rd : Assignment ++

Check

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 ;
}

Compiler error
```

Remember how compiler manipulate code ...
1st : Int(Mine)
Check
2nd : Assignment
Check

enum

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

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

Compiler error
```

Remember how compiler manipulate code ...
1st : Int(Mine)
Check
2nd : Assignment
Check

enum

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

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

Compiler error
```

Remember how compiler manipulate code ...
1st : Int(Mine)
Check
2nd : Assignment
Check

enum

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

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

Compiler error
```

Remember how compiler manipulate code ...
1st : Mine + BMW
Check
2nd : Assignment
Check

enum

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

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

Compiler error
```

Remember how compiler manipulate code ..

1st: Mine + BMW
2nd: Assignment "="
Check

enum

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

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

Compiler error
```

Remember how compiler manipulate code ..

1st: Mine + BMW
2nd: Assignment "="
Check

enum

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

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

Compiler error
```

Remember how compiler manipulate code ..

1st: Mine + BMW
2nd: Assignment "="
Check

enum

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

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

Compiler error
```

Remember how compiler manipulate code ..

1st: Mine + BMW
2nd: Assignment "="
Check

enum

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

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

Compiler error:
```

Remember how compiler manipulate code ...

 i.e: Mine + BMW

 i.e : Assignment "+"

 Mine + BMW

Can't Convert "Mine+BMW"
 Which is Integer to
 "Yours" which is an
 string

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

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;
#define MyTab '\t'
#define PonPon ":"D"
```

```
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 radius

2.5

The perimeter of the Circle = 15.708 (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
3.2
the perimeter of the Circle = 20.064
```

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	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 ;
}
```

-30

No parentheses needed for:

Operator manipulating

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

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

1

Not 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 indentation/compilation

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 ;
}
```

0

Operator manipulating

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

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

1

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++ ;           // a = 1 , c = 2
    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
<code>==</code>	equal to
<code>!=</code>	not equal to
<code><</code>	less than
<code><=</code>	less than or equal to
<code>></code>	greater than
<code>>=</code>	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");
}
```

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");
}
```

>

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");
}
```

1

2

Relational Operator

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

void main()
{
    int c = 1 , a = 8 ;
    a = ++c ;
    cout << a << endl << c << endl;
    system("pause");
}
```

8
2

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 = 8 ;
    ++a = 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 = 8 ;
    a = ++a ;
    cout << a << endl << c << endl;
    system("pause");
}
```

Variable can be assigned! 8 8

Relational Operator

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

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

2

Relational Operator

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

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

2

Relational Operator

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

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

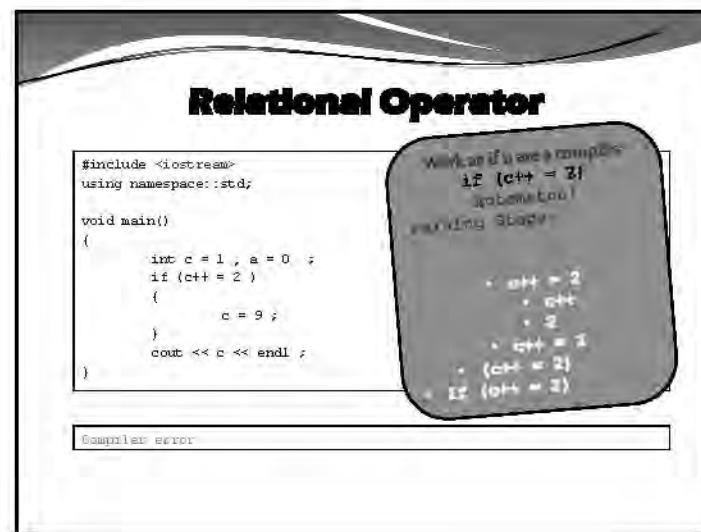
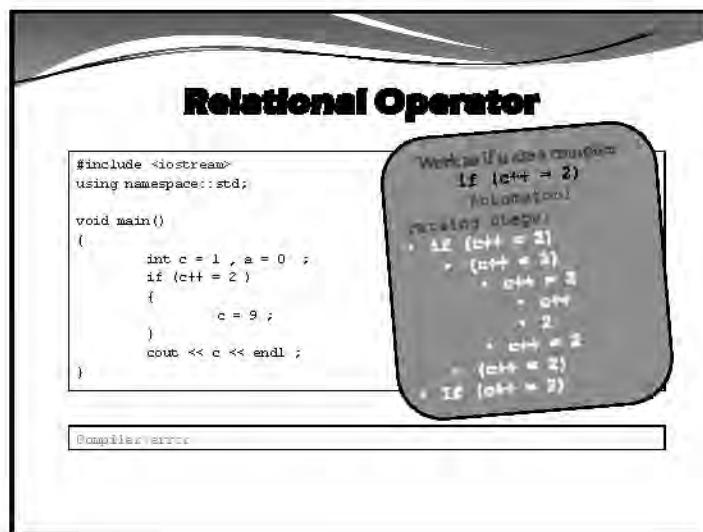
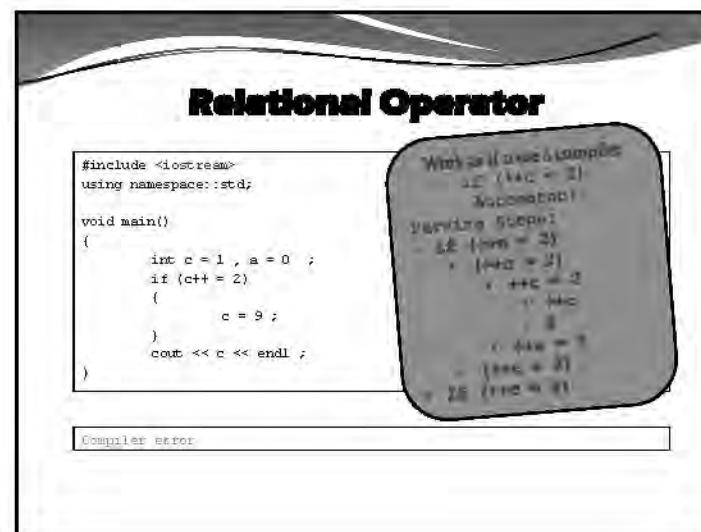
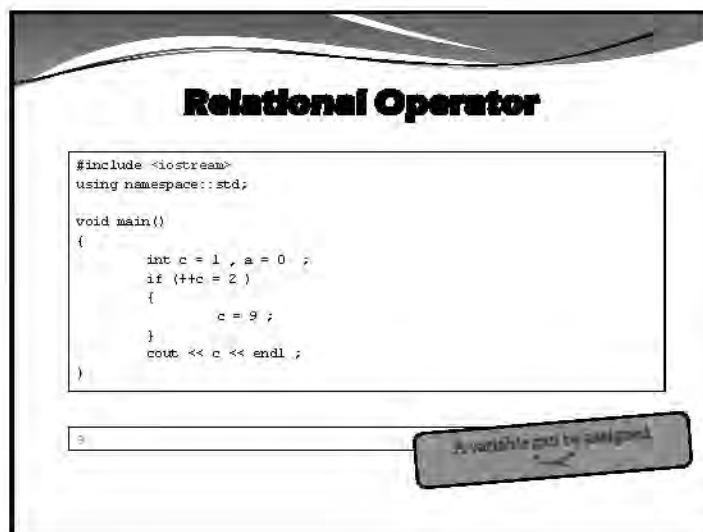
2

Relational Operator

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

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

2



Relational Operator

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

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

Works if we use a compiler
if (c++ = 2)
Automatically
preceding step:
 • 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)
    {
        c = 9 ;
    }
    cout << c << endl ;
}
```

Works if we use a compiler
if (c++ = 2)
Automatically
preceding step:
 • 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)
    {
        c = 9 ;
    }
    cout << c << endl ;
}
```

Works if we use a compiler
if (c++ = 2)
Automatically
preceding step:
 • c++
 • c++ = 2
 • (c++ = 2)
 • If (c++ = 2)

Compiler error

Relational Operator

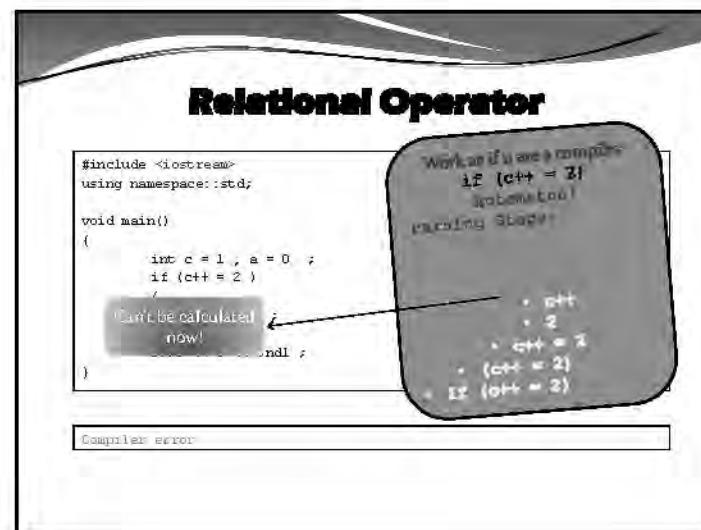
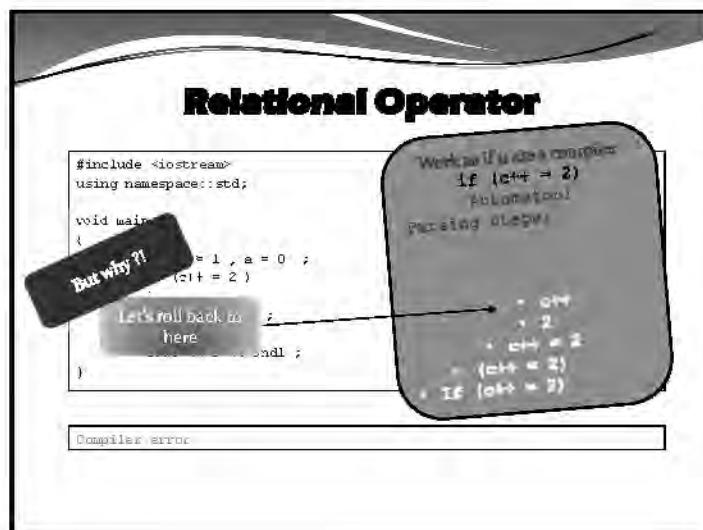
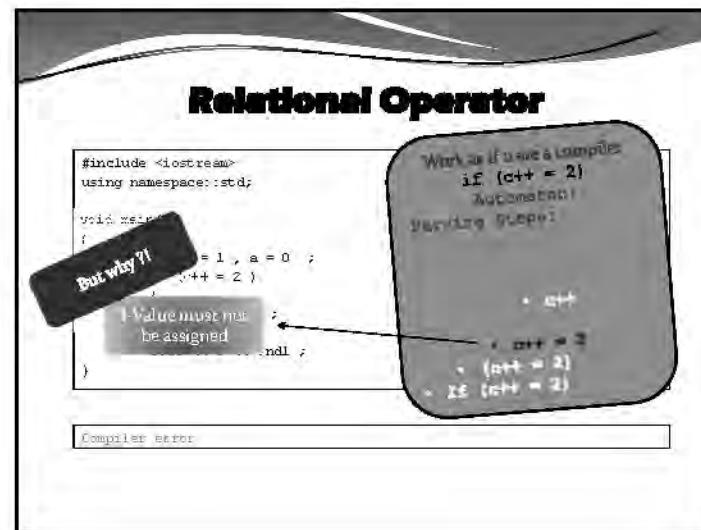
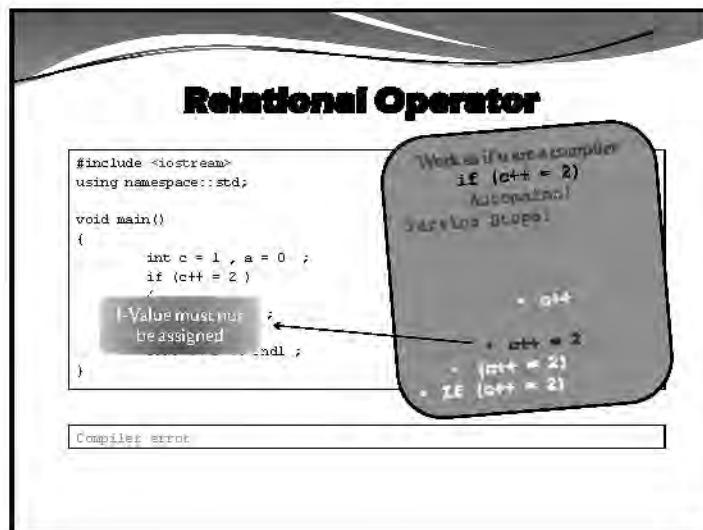
```
#include <iostream>
using namespace::std;

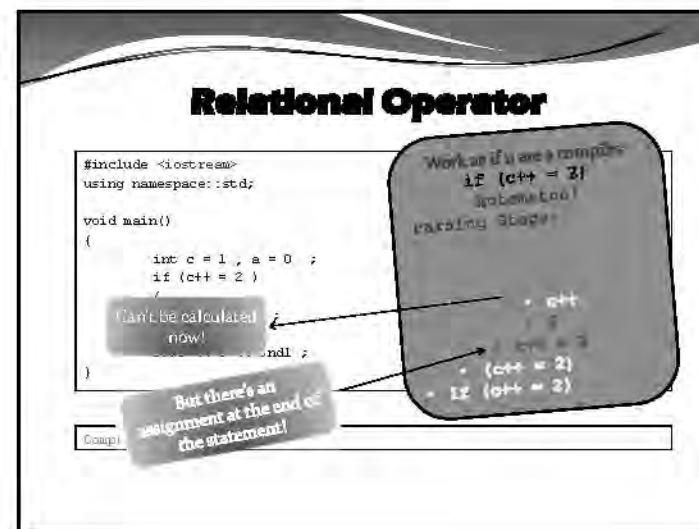
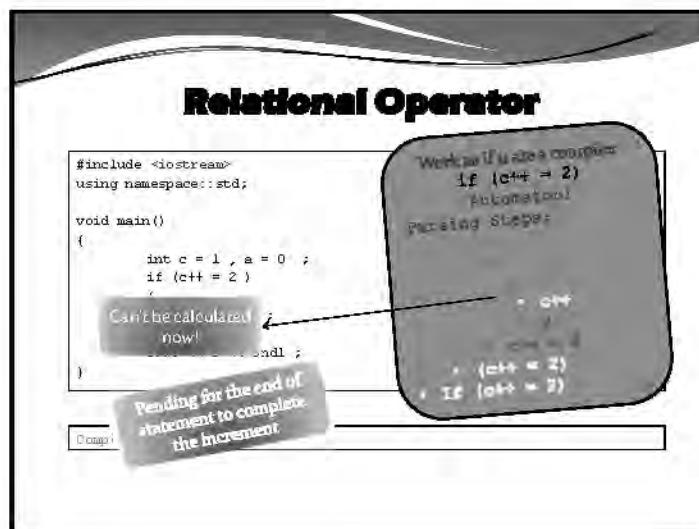
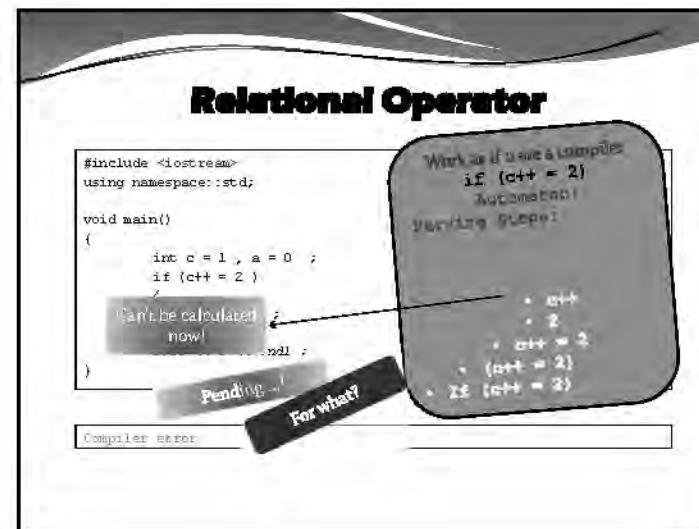
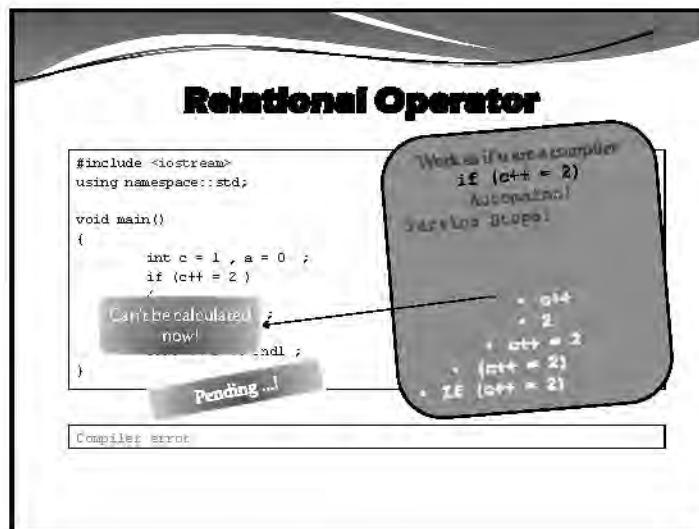
void main()
{
    int c = 1 , a = 0 ;
    if (c++ = 2)
    {
        c = 9 ;
    }
    cout << c << endl ;
}
```

Works if we use a compiler
if (c++ = 2)
Automatically
preceding step:
 • c++
 • c++ = 2
 • (c++ = 2)
 • If (c++ = 2)

Value must not
be assigned :
endl ;

Compiler error





Relational Operator

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

void main()
{
    int c = 1, a = 0;
    if (c++ == 2)
    {
        Can't be calculated now!
        endl;
    }
}
```

Working with complex
if (c++ == 2)
Automatic
variables
stop!

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

We can't assign then
increment

Relational Operator

Expression	Value of Expression	Explanation
'1' < '9'	true	The ASCII value of '1' is 49, and the ASCII value of '9' is 57. Because 49 < 57 is true, it follows that '1' < '9' is true.
'B' > 'T'	false	The ASCII value of 'B' is 66, and the ASCII value of 'T' is 84. Because 66 > 84 is false, it follows that 'B' > 'T' is false.
'4' < '2'	false	The ASCII value of '4' is 52, and the ASCII value of '2' is 48. Because 52 < 48 is false, it follows that '4' < '2' is false.
'6' <= '5'	false	The ASCII value of '6' is 54, and the ASCII value of '5' is 52. Because 54 <= 52 is false, it follows that '6' <= '5' is false.

Relational Operator with strings

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

str1	= "Hello"
str2	= "Air"
str3	= "Bill"
Expression	Value
str1 < str2	true
str1 > "Bar"	false
str3 < "Am"	true
str1 == "hello "	false
str3 != str1	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 ;
}
```

2

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 ;
}
```

1

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 expected

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

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

Evaluate Compiler
 $1 \leq (0 < c \leq 12)$
 Parsing Steps:
 • $1E (0 < c \leq 12)$
 • $(0 < c \leq 12)$
 • $0 < c$

?

Logical Operator

- Not always behave as expected

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

void main()
{
    int c = -1 , a ;
    if ( 0 < c <= 12 ) // Always true
    {
        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 expected

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

void main()
{
    int c = -1 , a ;
    if ( 0 < c <= 12 ) // Always true
    {
        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 expected

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

void main()
{
    int c = -1 , a ;
    if ( 0 < c <= 12 ) // Always true
    {
        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 expected

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

void main()
{
    int c = -1 , a ;
    if ( 0 < c <= 12 ) // Always true
    {
        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 expected

```
#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)

Now Let's parse!

Logical Operator

- Not always behave as expected

```
#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 expected

```
#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:

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

Logical Operator

- Not always behave as expected

```
#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:

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

Logical Operator

- Not always behave as expected

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

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

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)

9

Logical Operator

- Not always behave as expected

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

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

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)

9

Logical Operator

- Not always behave as expected

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

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

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)

9

Logical Operator

- Not always behave as expected

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

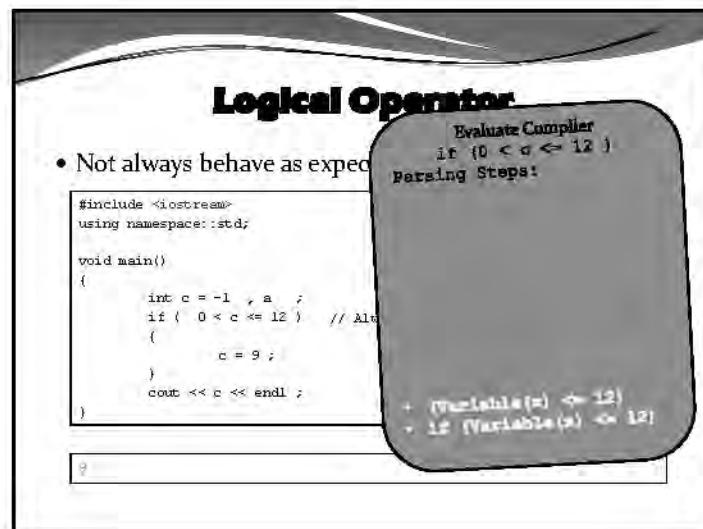
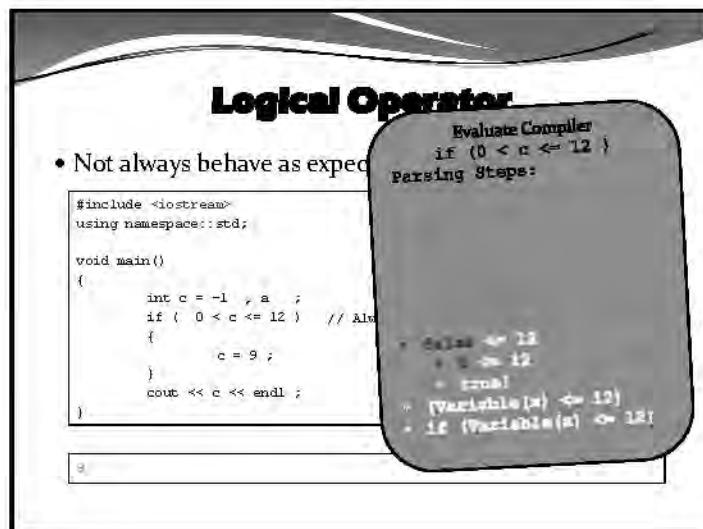
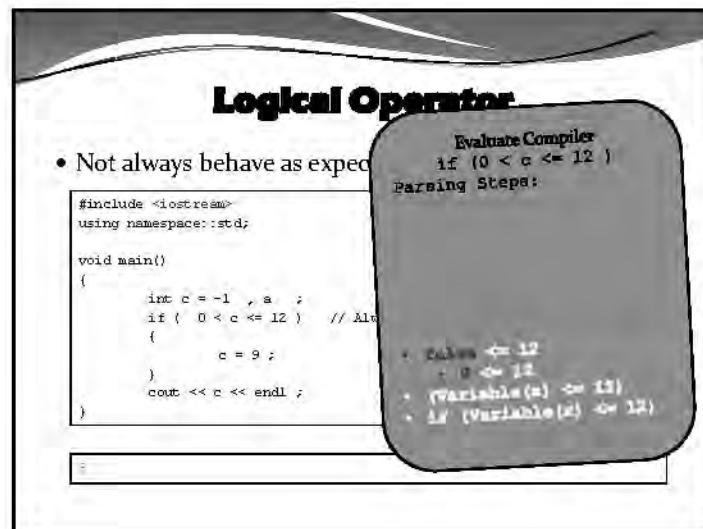
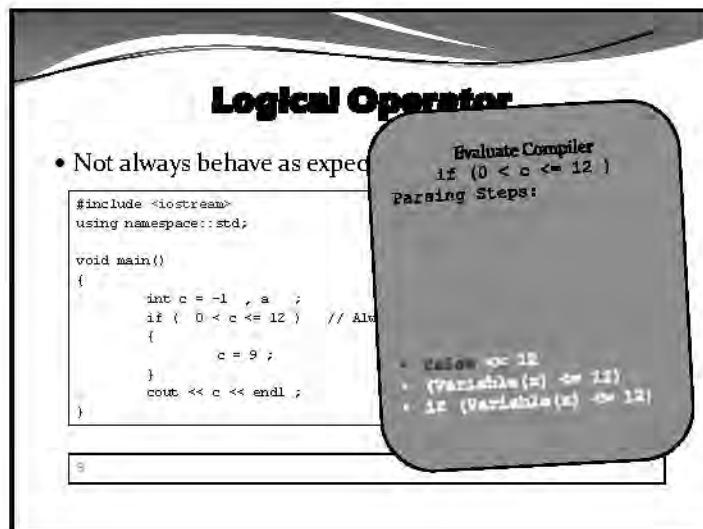
void main()
{
    int c = -1 , a ;
    if ( 0 < c <= 12 ) // Always
    {
        c = 9 ;
    }
    cout << c << endl ;
}
```

Evaluate Compiler
if (0 < c <= 12)

Parsing Steps:

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

9



Logical Operator

- Not always behave as expected

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

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

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

- + (==)
- + if (variable(c) <= 12)

9

Logical Operator

- Not always behave as expected

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

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

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

- + (==)
- + if (variable(c) <= 12)

9

Logical Operator

- Not always behave as expected

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

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

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

- + if (true)

9

Logical Operator

- Not always behave as expected

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

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

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

- + if (true)

9

Logical Operator

- Not always behave as expected

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

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

0

Conditional Operator ?

- Code :

Condition ? Result1 : Result2

- Condition

- If True
 - Result1
- Else // condition is false
 - Result2

Logical Operator

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

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

1

Logical Operator

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

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

1

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

Logical Operator

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

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

1
Also 1 result
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 ;
}
```

1



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;
1
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 use variable "b" before "d"?

Dot... Mr. Khens... (x)... 100 pts.



Code Cracking

X

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

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

5

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



Code Cracking

X

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

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

1

Reference Point!

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



Code Cracking

X

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

void main()
{
    int c=1 , b=2 , d=3;
    cout << c+(l+++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 : **lvalue**

Reference Point!

IT's like the slides before when using `cout << value - 1;` and put another `value` "by 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 ?

Small 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: enum++

Code Cracking

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

void main()
{
    enum Cars { Nissan = 2,4 ,BMW ,Mercedes ,Ferrari ,Renault
}MyCar = Nissan ;
    MyCar = Renault ;
    int il = MyCar ;
}
```

Compiler error: Nissan + 3,4 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 ;
}

Need to declare == operator
```

Code Cracking

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

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

3
```

Code Cracking

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

void main()
{}
```

2005 Compiler : Compiler error no decimal int supported
2005 Compiler : Compiler error 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 ;

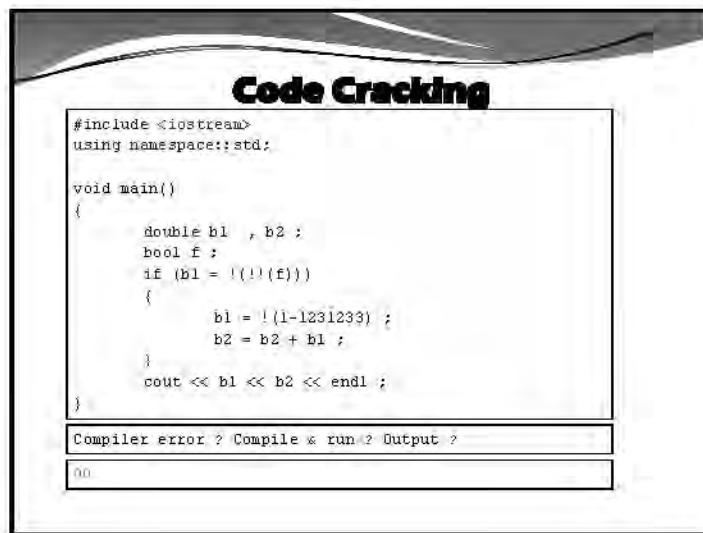
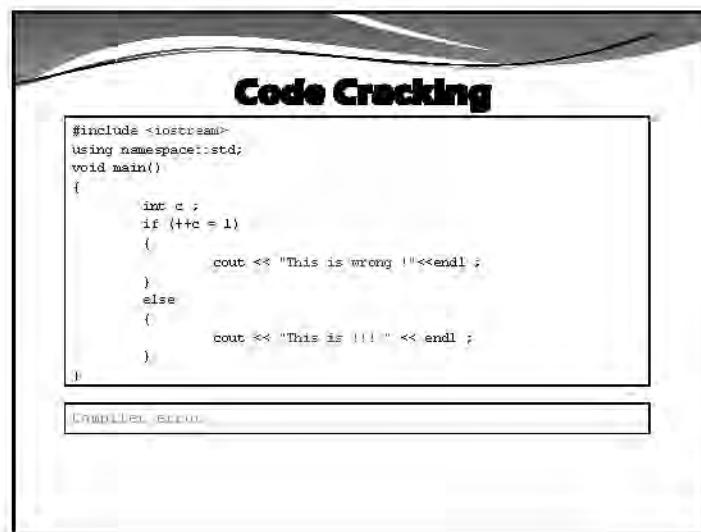
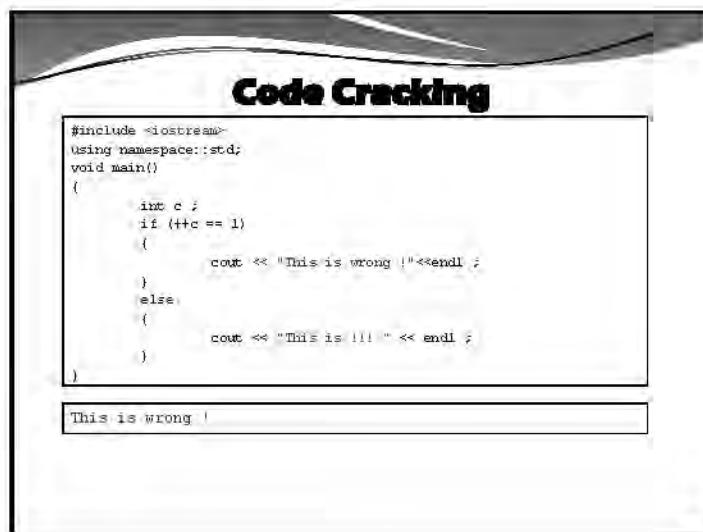
void main()
{
    cout << x << MyStringTab ;}
```

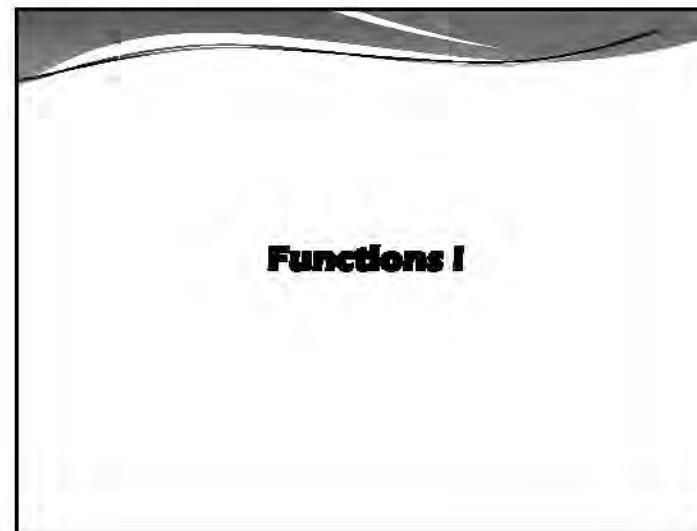
2008 No , it's a compiler error

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 !!!





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 a either 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 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 ;
}
```

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 ;
}
```

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 ;
}
```

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
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 << endl ;
    return 0 ;
}
```

6
6

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
6

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 ;
}
```

6
6

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 message:

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 message: error C2045: '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 message: warning C4244: conversion from 'double' to 'int', possible loss of information

Functions

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

void Mult(void);

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

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

Compiler message: warning C4244: conversion from 'double' to 'int', possible loss of information

Functions

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

void Mult(void);

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

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

Compiling... (use 'make' from bottom toolbar)

Functions

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

void Mult(void);

void main(void)
{
    Mult() ;
}

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

All done - everything is working :-)

Functions

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

void Mult(void);

void main()
{
    Mult() ;
}

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

All done - everything is working :-)

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 ;
}
```

Compiling... (use 'make' from bottom toolbar)

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 ;
}
```

Usage of local variable 'x' outside its scope

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 - undeclared function "Mult"

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 ;
}

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 ;
}

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 ;
}

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

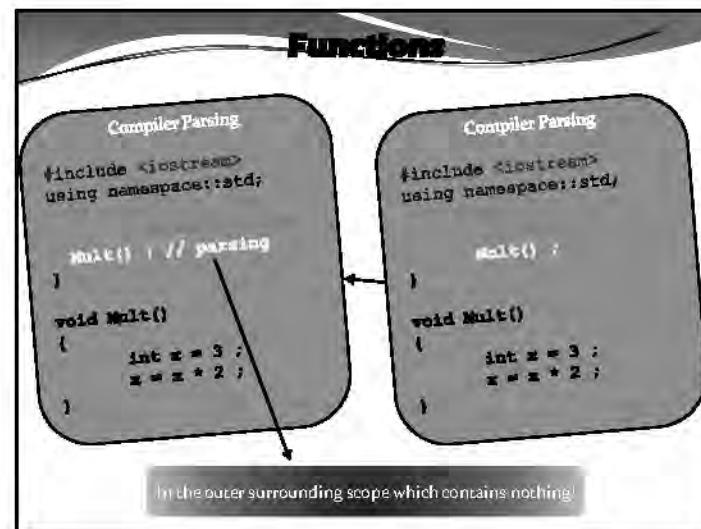
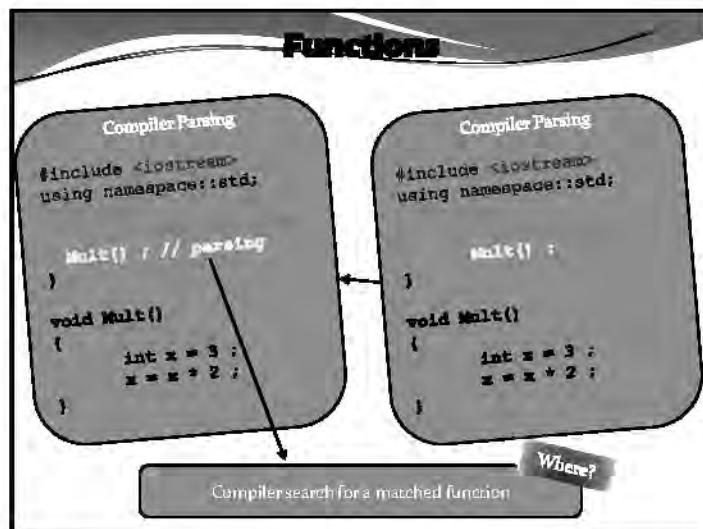
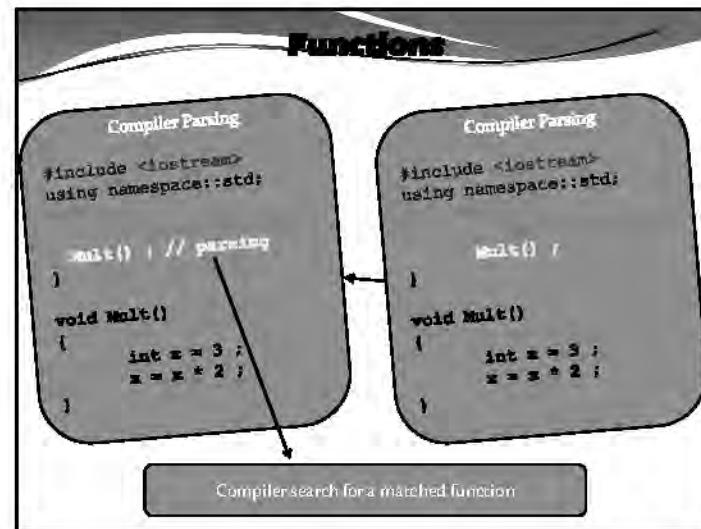
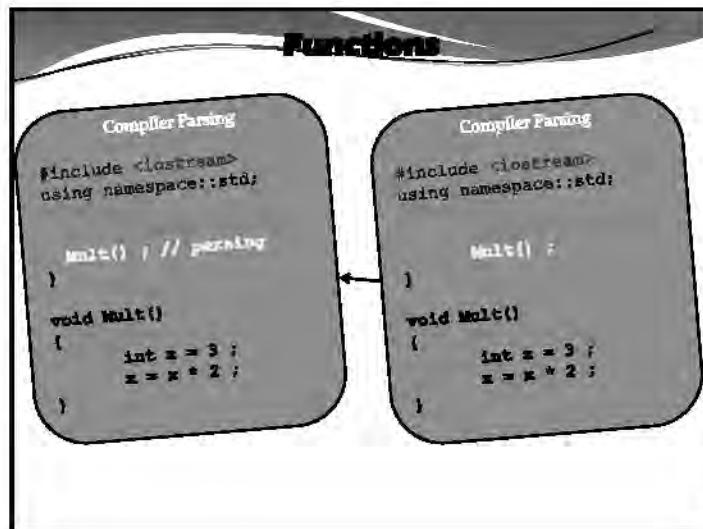
Compiler Parsing

```
#include <iostream>
using namespace::std;
{
    Mult() ;
}

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

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

Compiler error ... missing function's prototype



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 << endl ;
}
```

6

Output will be mixed with character stream.

Functions

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

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

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

6

Output will be mixed with character stream.

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 ;
}
```

6

Compiler error - undeclared 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 ;
    }
}
```

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 , int y)
{
    if (x == 2 )
    {
        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 ;
}

Compile: a.out
Run:
0
4
a.out: warning:
```

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 ;
}

0-0-0-0-10-10-0-0-16-00-hehe
9 More may say 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 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;
}
```

10

Compiler error: identifier was not defined

Headers – Math Library

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

void main(void)
{
    cout << sqrt(100) << endl;
}
```

Compile and run the program "sqrt1.cpp".

Headers – Math Library

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

void main(void)
{
    cout << sqrt(sqrt(100.0)) << endl;
}
```

Compile and run the program "sqrt2.cpp".

Headers – Math Library

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

void main(void)
{
    int x ;
    cout << sqrt(6*x) << endl;
}
```

Compile and run the program "sqrt3.cpp".

Headers – Math Library

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

void main(void)
{
    double x = 3 ;
    cout << sqrt(3*x) << endl;
}
```

Compile and run the program "sqrt4.cpp".

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 ;
}

5
5
5
Now, when compiling many times ...
Every time we have the same value !
```

5
5
5
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 ;
}

5
5
5
Nope without <cstdlib>
```

5
5
5
Nope without <cstdlib>

rand()

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

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

150
151
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 ;
}
```

152
153

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 ;
}
```

Random value -> Value from 0 to 3

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
789
123

*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
49
45

Not an appropriate srand() parameter leads to insufficient usage 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
}

5
(the user entered command line)
```

rand()

```
#include <iostream>
#include <ctime>
using namespace::std;
void main(void)
{
    srand(clock());
    // clock is from <ctime>
    // so needs no value for clock()
    for (int i = 0 ; i < 10 ; i++)
        cout << rand() << endl ;
}
```

```
704
9415
16377
2907
14053
5420
20050
16183
20979
1145
(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 = 3
In function x = 6
8
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: main.c: In function 'main':

2 main's in the same object

Playing with scopes I

- 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 error:

This is inner block

Playing with scopes I

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

void main(void)
{
    int x = 9 , y = 3 ;
    {
        cout << x << endl ;
    }
}
```

Playing with scopes !

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

void main(void)
{
    int x = 9 , y = 3 ;
    {
        cout << x << endl ;
    }
    cout << x << endl ;
}
```

Playing with scopes !

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

void main(void)
{
    int x = 9 , y = 3 ;
    {
        int x = 4 ;
        cout << x << endl ;
    }
}
```

Playing with scopes !

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

void main(void)
{
    int x = 9 , y = 3 ;
    {
        int x ;
        x = 4 ;
        cout << x << endl ;
    }
    cout << x << endl ;
}
```

Playing with scopes !

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

void main(void)
{
    int x = 9 , y = 3 ;
    {
        int x = 4 ;
        cout << x << endl ;
    }
    cout << y << endl ;
}
```

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 ;
}
```

Compile error : 7 undeclared identifiers



Playing with scopes !

```
#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 ;
    cout << ::x << endl ;
}

void main(void)
{
    int x = 9 , y = 3 ;
    {
        int x = 6 ; // Local Variable
    }
}
```

Compile error : local declaration in outer scope
Nothing to do here !

Playing with scopes !

```
#include <iostream>

using namespace::std;
int x = 34 ; // Global Variable

void SoSo ()
{
    int x = 3 ; cout << x << endl ;
} // local Variable

void main(void)
{
    int x = 9 , y = 3 ;
    {
        int x = 6 ; // local Variable
    }
    SoSo();
}

```

34
3
6
3

Playing with scopes !

```
#include <iostream>

using namespace::std;
int x = 34 ; // Global Variable

void SoSo ()
{
    int x = 3 ; cout << ++x << endl ;
} // local Variable

void main(void)
{
    int x = 9 , y = 3 ;
    {
        int x = 6 ; // local Variable
    }
    SoSo();
}

```

35
3
6
3

Playing with scopes !

```
#include <iostream>

using namespace::std;
int x = 34 ; // Global Variable

void SoSo ()
{
    int x = 3 ; cout << ::x++ << endl ;
} // local Variable

void main(void)
{
    int x = 9 , y = 3 ;
    {
        int x = 6 ; // local Variable
    }
    SoSo();
    cout << ::x << endl ;
}

```

34
3
6
3

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;
}
```

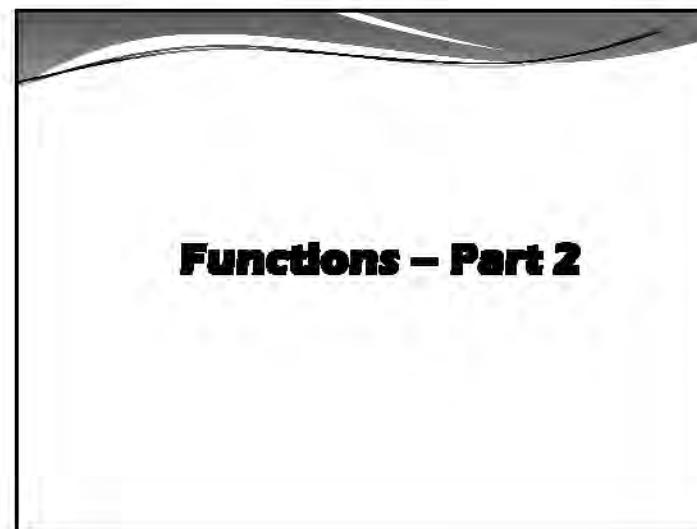
Enter the Input Value: 2

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(2.2) is 3.0 ceil(-1.1) is -1.0
cos(x)	trigonometric cosine of x (x in radians)	cos(0) = 1.0 (1.0)
exp(x)	exponential function ex	exp(1.0) = 1.0 (2.71828) exp(-1.0) = 0.73576
fabs(x)	absolute value of x	fabs(-3.0) is 3.0 fabs(0.0) is 0.0 fabs(1.0) is 1.0
floor(x)	rounds x to the largest integer not greater than x	floor(1.4) is 1.0 (1.4) floor(-1.4) is -2.0 (-1.4)
fmod(x, y)	remainder of x/y as a floating-point number	fmod(1.0, 0.5) is 0.5 (0.5)
log(x)	natural logarithm of x (base e)	log(1.0) is 0.0 (0.0) log(2.0) is 0.693147 (0.693147)
log10(x)	logarithm of x (base 10)	log10(1.0) is 0.0 (0.0) log10(10.0) is 1.0 (1.0)
pow(x, y)	calculated power of x	pow(2.0, 3.0) is 8.0 (8.0) pow(2.0, -1.0) is 0.5 (0.5)
sinc(x)	hyperbolic sine of x (x in radians)	sinc(0.0) is 1.0 (1.0) sinc(1.0) is 0.841471 (0.841471)
sqr(x)	square root of x	sqr(1.0) is 1.0 (1.0) sqr(-1.0) is -1.0 (-1.0)
tan(x)	trigonometric tangent of x (x in radians)	tan(0.0) is 0.0 (0.0)



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);
}
```

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);
}
```

Recursion I

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

int factorial(int n)
{
    if (n == 0)
        return 1 ;
    else
        return n*factorial(n-1) ;
}

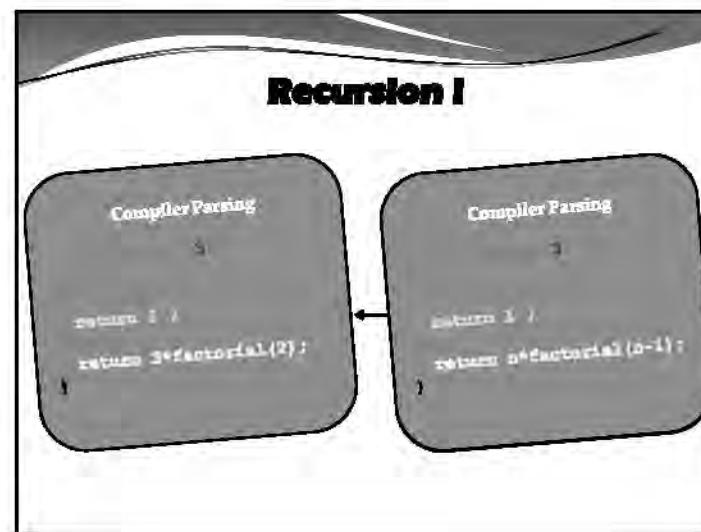
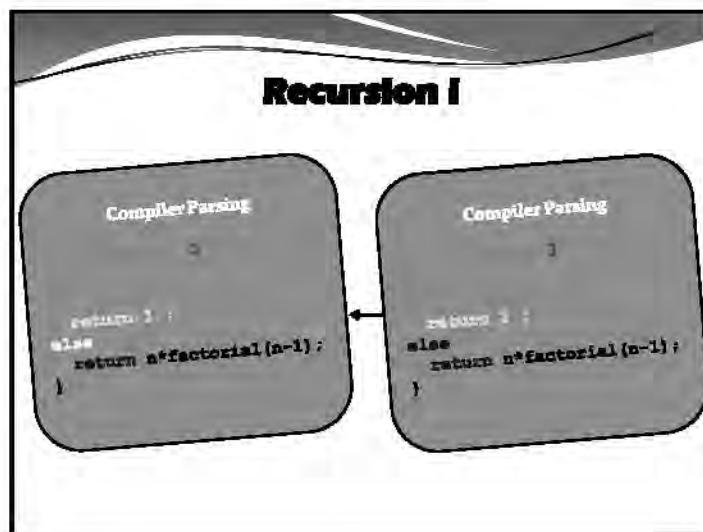
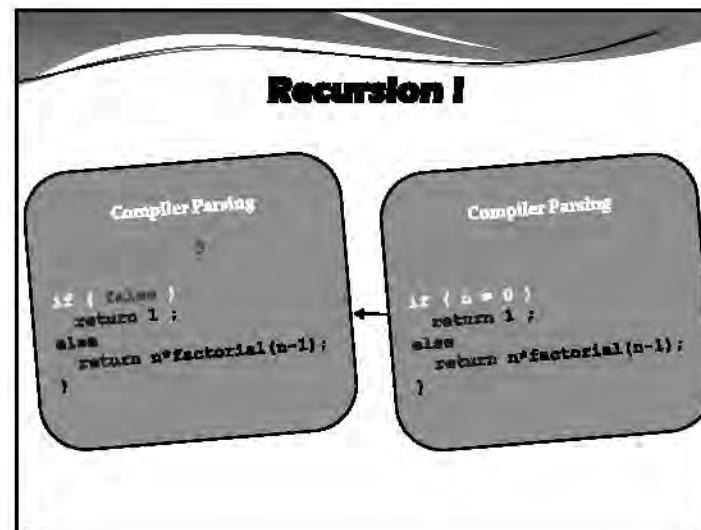
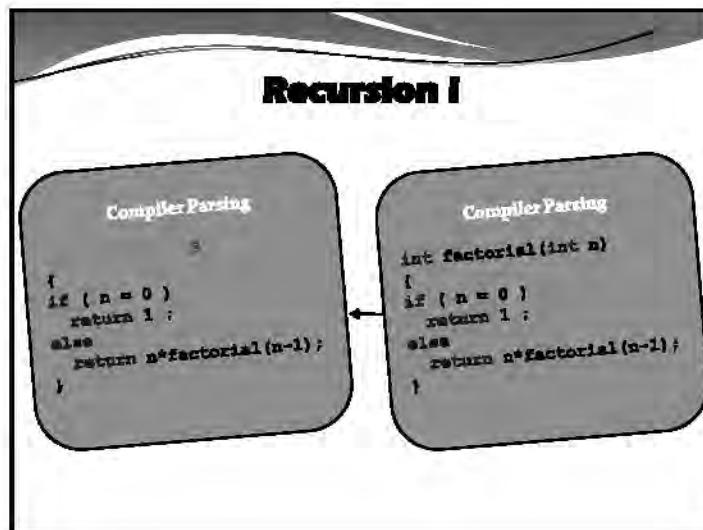
void main(void)
{
    cout << factorial(3);
}
```

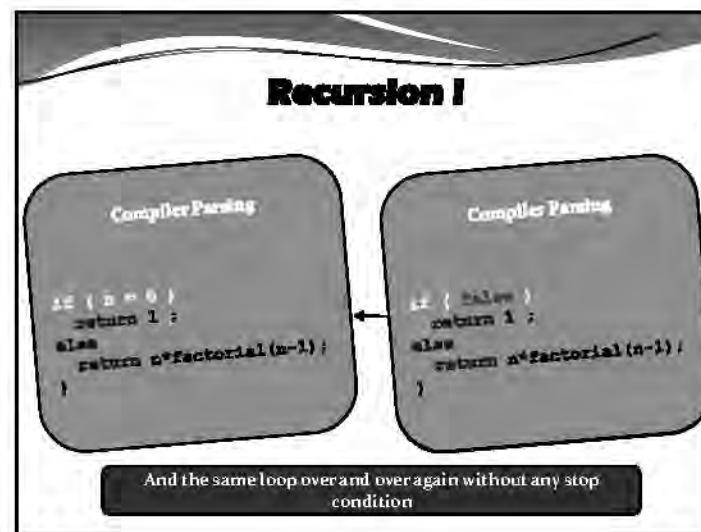
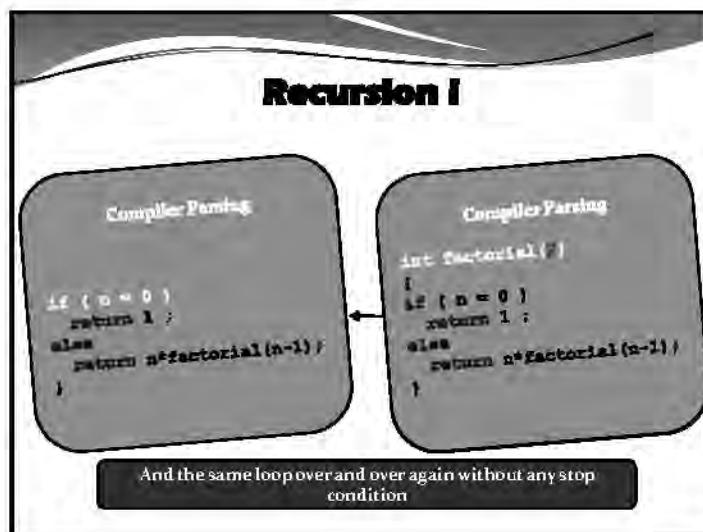
Compile the programs above | stack overflow | what would happen??

So the output should be gigantic!!

Bottom line: try to be the same as calculated and don't do loops under the same

150 pts





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 ;
}
```

9

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

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);
}
```

This is a Recursive call
this is a Recursive call
this is a Recursive call
this is a Recursive call . Number 1
this is a Recursive call . Number 2
this is a Recursive call . Number 3
this is a Recursive call . Number 4

**Note the space after first line
as 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 3
this is a Recursive call . Number 2
this is a Recursive call . Number 1
Press any key to continue

Recursion !

- Recursive function sum of numbers from 1 to n
- Discussion ...
 - How to do it ?
 - Finding the idea !
 - Rule to Rule !
 - We have ...
 - $1 + 2 + 3 + \dots + n$
 - The Rule is :
 - $n + \text{sum of the numbers from } 1 \text{ to } n-1 (\dots \sum_{i=1}^{n-1})$
 - $n + \text{summation}(n-1)$
 - 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;
}
```

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;
}
```

10

Recursion I

```
#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 Decided (0.000000) , checking = 10

Recursion I

```
#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");
}
```

10 - Microsoft Visual Studio [10]
Should return 10 in this case

Recursion I

```
#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 ;
}
```

Normal EC - 100% - sum as small function // Solution one

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
 - $2^1 = 2 \cdot 2^0$ $2^2 = 2 \cdot 2^1$; ...
 - The Rule is:
 - $x \cdot x^{(n-1)}$
 - 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 (n == 0 )
        return 1 ;
    else
    {
        return (x*power(x,n-1));
    }
}

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

Press any key to continue . . .

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(2,2) << endl ;
}
```

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 << endl ;
    if (times > 0 )
    {
        cout << "This is recursive func -\n" ;
        CrazyMessages(times-1);
    }
    cout << "Msg Returning with " << times << endl ;
}

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 << endl ;
    if (times >= 0 )
    {
        cout << "This is recursive func -\n" ;
        cout << "Times are : " << times << endl ;
        CrazyMessages(times-1);
    }
    cout << "Msg Returning with " << times << endl ;
}

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 Returning with 3 !!!
Msg Returning with 2 !!!
Msg Returning with 1 !!!
Msg Returning with 0 !!!
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(2,2) << endl ;
}
```

Press any key to continue

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);
}
```

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);
}
```

Example: trying to assign default values for parameters

When defaulting a parameter in a function, all other parameters -if found- ~~should be~~ should be defaulted too

Default parameters

```
#include <iostream>
using namespace std;
int x = 0;

void print(int x, int n=0)
{
    cout << x << endl;
}

void main(void)
{
    print(2,3);
}
```

?

Default parameters

```
#include <iostream>
using namespace std;
int x = 0;

void print(int x, int n, int y=0)
{
    cout << x << endl;
}

void main(void)
{
    print(2,3,4);
}
```

?

Y=1 is defined

Default parameters

```
#include <iostream>
using namespace std;
int x = 0;

void print(int x, int n, int y=0)
{
    cout << x << endl;
}

void main(void)
{
    print(2,3,4);
}
```

?

Y=1 is defined

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
3
0

Default parameters can be used in the function parameter.

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: can we call it argument?

Non - call it 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 ... because didn't compile the :-)

Non - call it 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 ... cos we didn't default the y ...
Now... did u know why? :-)
```

Compiler Parsing

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 ... cos we didn't default the y ...
Now... did u know why? :-)
```

Compiler Parsing

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 ... cos we didn't default the y ...
Now... did u know why? :-)
```

Compiler Parsing

Default parameters

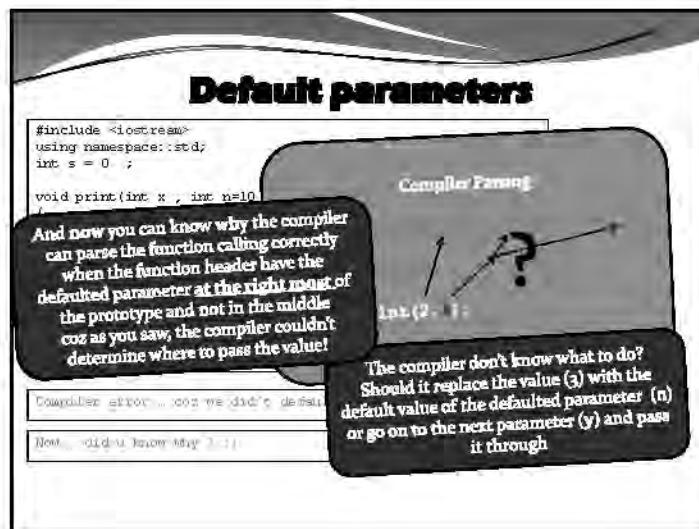
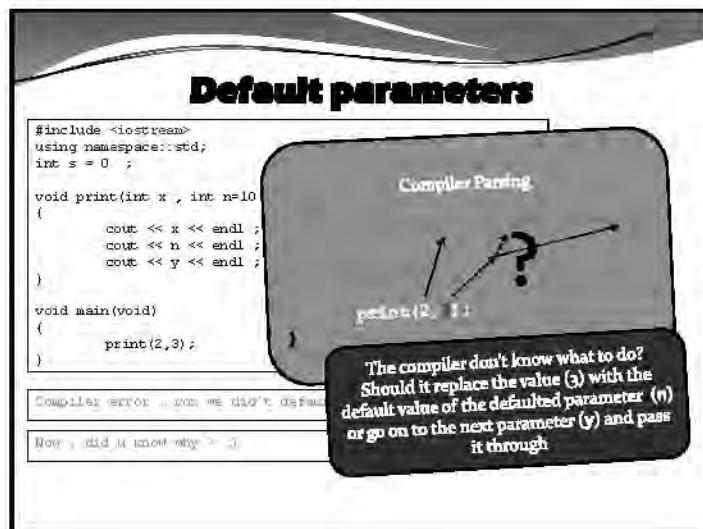
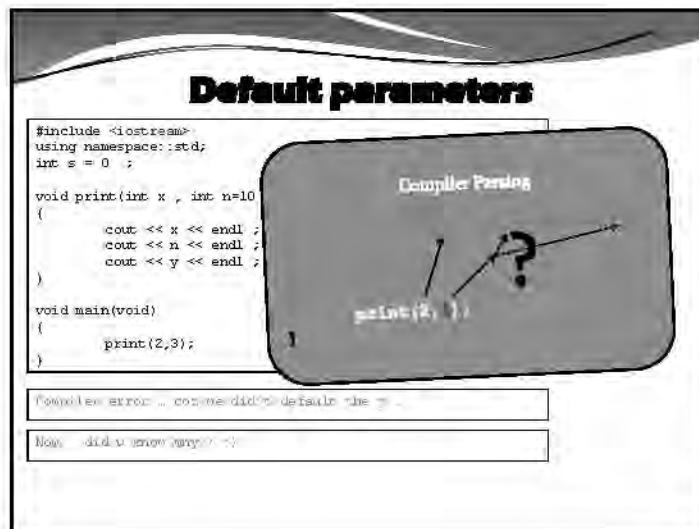
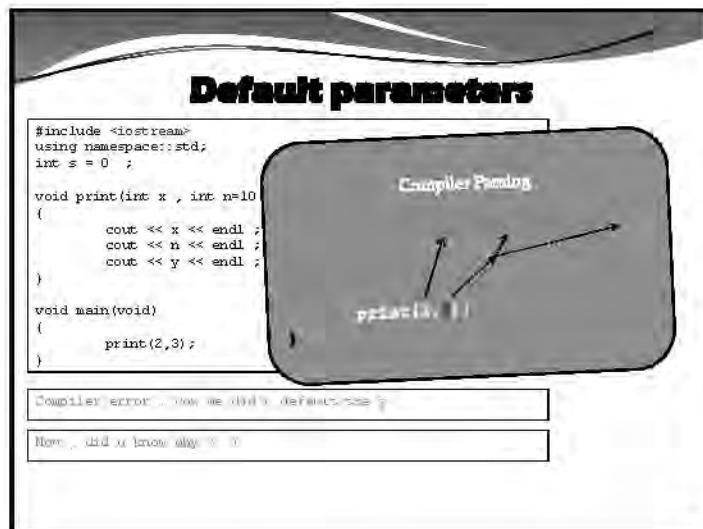
```
#include <iostream>
using namespace::std;
int s = 0 ;

void print(int x , int n=10)
{
    cout << x << endl ;
    cout << n << endl ;
    cout << y << endl ;
}

void main(void)
{
    print(2,3);
}

Compiler error ... cos we didn't default the y ...
Now... did u know why? :-)
```

Compiler Parsing



Default parameters

```
#include <iostream>
using namespace::std;
int a = 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 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 = 0
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 : 2
 Please 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 ... S 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 ... S before 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 ;
}
```

```
1.main() : Before calling FuncByValue()
1.0: function before multiplication, x = 2
1.main() : After multiplication, x = 8
2.
2.main() : Before calling FuncByValue(), x = 2
2.main() : After calling FuncByValue(), x = 8
2.main() : After calling FuncByValue(), x = 8
```

Calling by value Vs calling by reference

```
#include <iostream>
using namespace::std;
int s = 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 ;
}
```

Output :
In main , before calling the function , x = 2
In Function after multiplication , x = 8
In main , after calling the function , x = 2
Press [Enter] to continue...

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 x = 0 ;

int f ( int xx )
{
    x = x * 2 ;
    return x ;
}

int f ( float xx )
{
    x = x * 2 ;
    return x ;
}

void main(void)
{
    int y1 = f(2) ;
    float y2 = f(2) ;

    cout << f(y1) << endl ;
    cout << f(y2) << endl ;
}
```

Output :
f
f
Overloading Demands More Functionality Comparison

Function Overloading

```
#include <iostream>
using namespace::std;
int s = 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 ;
}
```

Output: 4
6

0x1111111111111111 → 1111111111111111

Function Overloading

```
#include <iostream>
using namespace::std;
int s = 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 ;
}
```

Output:
4
6

Function Overloading

```
#include <iostream>
using namespace::std;
int s = 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 ;
}
```

Output:
4
6

0x1111111111111111 → 1111111111111111

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);
}
```

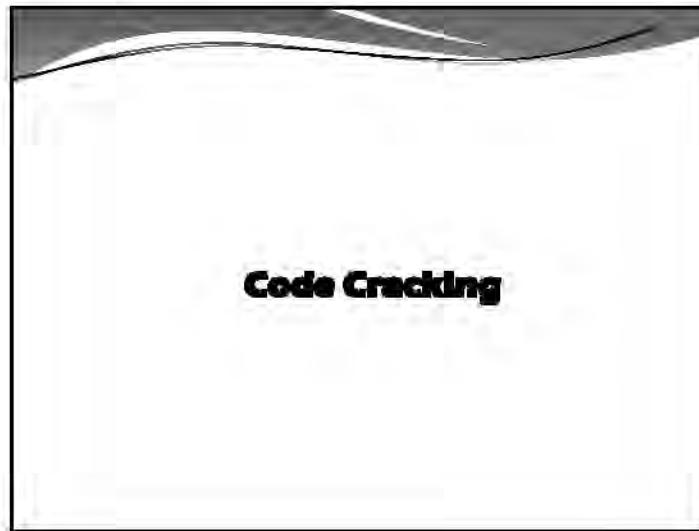
Output:
4
6

0x1111111111111111 → 1111111111111111

```
#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);
}
```

```
#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 x , float 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);
}
```

```
#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 x , float z)
{
    x = x * 2 ;
    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

```
#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 define



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--
this is a Recursive call , Number--
this is a Recursive call , Number--
Press any key to continue . . .

Recommend ID: 23

```
#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);
}
```

MEV	$\approx 2.5 \times 10^{-11}$	Stand. dev. = 1.1%
OB	$\approx 0.3 \times 10^{-11}$	Stand. dev. = 0.0%
QEV	$\approx 0.3 \times 10^{-11}$	Stand. dev. = 0.0%

Recursion I - 44

```
#include <iostream>
using namespace std;
int s = 0 ;

void CrazyMessages (int times )
{
    cout << "Msg called with " << times << endl ;
    if (times >= 0 )
    {
        cout << "This is recursive func.\n" ;
        cout << "Times are : " << times-- << endl ;
        CrazyMessages(times-1);
    }
    cout << "Msg Returning with "<<times<< endl;
}

void main(void)
{
    CrazyMessages (3) ;
}
```

```
Big called with 3  
This is recursive func  
Times are 3  
Big called with 2  
This is recursive func  
Times are 2  
Big called with 1  
Big Recurusing with 0 ()  
Big Recurusing with 1 ()  
Times are recursive count 1
```

Recommendation Letter Out

```
#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

startle | -shərəl

- Let's see this

```
#include <iostream>

using namespace std;
int x = 34 ;

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 ! - #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 !

```
9
5
9
first x in file = 9
second x in file = 6
push(x) in file = 6
second x in file = 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
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 ;
    }
}
```

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 ;
    cout << "GoGo !" << endl ;
}
```

True
False
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");
}
```

s = 0
Means False!

Remember that every other number other than 0 means true

I know

Tweaked Control Structure I

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

void main(void)
{
    int x = 5 ;
    while (x != 10)
    {
        x++;
        cout << x << endl ;
    }
}
```

5
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 ;
    }
}
```

```
25  
36  
49  
64  
81  
100
```

Tweaked Control Structure I

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

void main(void)
{
    int x = 5 ;
    while (x*x < 10)
    {
        cout << x*x << endl ;
    }
}
```

Nothing happens

Tweaked Control Structure I

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

void main(void)
{
    int x = 5 ;
    while (cin)
    {
        cin >> x ;
        cout << x*x << endl ;
    }
}
```

Nothing happens

Tweaked Control Structure I

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

void main(void)
{
    int x = 5 ;
    do
    {
        cout << x << endl ;
        x++;
    } while ( x < 10 ) ;
}
```

Nothing happens

Tweaked Control Structure I

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

void main(void)
{
    int x = 5 ;
    do
    {
        cout << x << endl ;
        x++;
    } while ( x < 10 ) ;
}
```

Computer says nothing when one runs it

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 atleast one , even though 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 :
2
You have entered , x = 2
Enter the x :
0
You have entered , x = 0
Please 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
Please 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 = 1
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
beep ! , number = 5
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 ; i++)
    {
        cout << "beep ! , number = " << i << endl ;
        i++;
    }
    cout << "finished" << endl ;
}

beep .. number = 0
finished
Press any key to continue
Why is it ok?
Watch out for the semi colon ';' after the for statement.. Cos it close it
```

```
#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?
See below is still working in an infinite loop !
```

```
#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 = 4
beep ! , number = 5
beep ! , number = 6
beep ! , number = 7
beep ! , number = 8
beep ! , number = 9
finished
Press any key to continue
No-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 < c ; 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
beep ! , number = 5
beep ! , number = 6
beep ! , number = 7
beep ! , number = 8
beep ! , number = 9
beep ! , number = 10
beep ! , number = 11
beep ! , number = 12
beep ! , number = 13
beep ! , number = 14
beep ! , number = 15
beep ! , number = 16
beep ! , number = 17
beep ! , number = 18
beep ! , number = 19
beep ! , number = 20
beep ! , number = 21
beep ! , number = 22
beep ! , number = 23
beep ! , number = 24
beep ! , number = 25
beep ! , number = 26
beep ! , number = 27
beep ! , number = 28
beep ! , number = 29
beep ! , number = 30
beep ! , number = 31
beep ! , number = 32
beep ! , number = 33
beep ! , number = 34
beep ! , number = 35
beep ! , number = 36
beep ! , number = 37
beep ! , number = 38
beep ! , number = 39
beep ! , number = 40
beep ! , number = 41
beep ! , number = 42
beep ! , number = 43
beep ! , number = 44
beep ! , number = 45
beep ! , number = 46
beep ! , number = 47
beep ! , number = 48
beep ! , number = 49
beep ! , number = 50
beep ! , number = 51
beep ! , number = 52
beep ! , number = 53
beep ! , number = 54
beep ! , number = 55
beep ! , number = 56
beep ! , number = 57
beep ! , number = 58
beep ! , number = 59
beep ! , number = 60
beep ! , number = 61
beep ! , number = 62
beep ! , number = 63
beep ! , number = 64
beep ! , number = 65
beep ! , number = 66
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 < c ; i++)
    {
        cout << "beep ! , number = " << i << endl ;
    }
    cout << "finished" << endl ;
}

Compile: ex01

```

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 ;
}
```

Compile error - undeclared identifier

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 ;
}
```

Compile error - 3 undeclared identifiers

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
0
-1
0

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 == 3)
        {
            break ;
        }
    }
}
```

0

1

2

Tweaked Control Structure I

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

void main(void)
{
    for (int i = 0 ; i<5 ; i++)
    {
        cout << i << endl ;
        if ( i == 3)
        {
            break ;
        }
    }
}
```

0

1

2

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
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
3
4
```

Output: as you ... = = cout <<

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 << "WHEEEEEE ! ! ! ";
            break ;
        }
    }
}

0
1
2
3
4
```

Output: WHEEEEEE ! ! ! This may say 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
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)
        {
            cout << "WeeWeee" << endl ;
            continue ;
        }
    }
}

0
1
2
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 << "WeeWeee" << endl ;
        }
    }
}

0
1
2
3
4
WeeWeee
5
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 = 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 ;
    }
}

Enter x:
You have 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:
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 ;
    }
}

Enter x:
3
You have entered , x = 3
WOW , i can't believe it , you entered x = 3
WOW , i can't believe it , you entered x = 3
Not a 1 or 2
Press any key to continue . . .

```

Coz of forgetting break

Tweaked Control Structure I

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

void main(void)
{
    int x = 2;
    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;
        case 2 :
            cout << "Wow , i can't believe it , you entered x = 2" << endl;
        default :
            cout << "Not a 1 or 2 " << endl;
    }
}

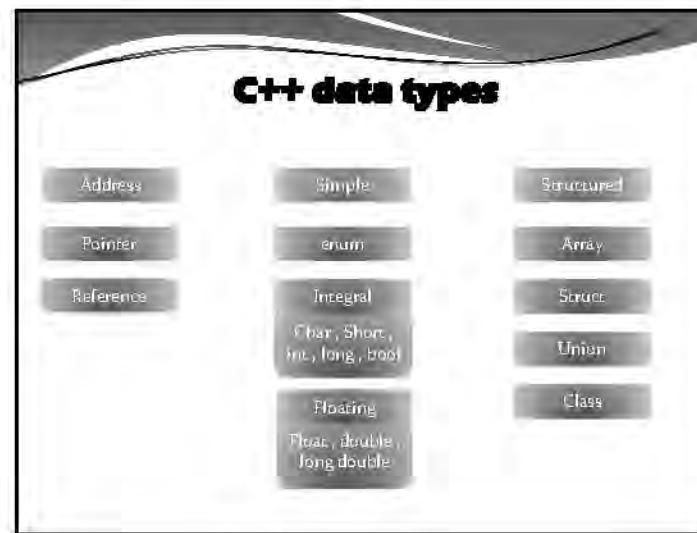
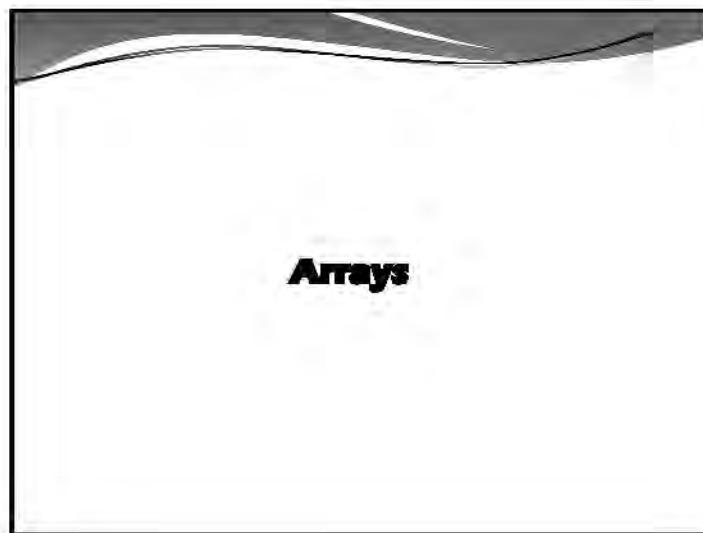
Enter x
1
You have entered / x = 1
Wow , i can't believe it , you entered x = 1
1
Wow , i can't believe it , you entered x = 2
1
Not a 1 or 2
Press any key to continue...
```

Tweaked Control Structure I

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

void main(void)
{
    int x = 2;
    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
You have entered / x = 2
Wow , i can't believe it , you entered x = 2
1
Press any key to continue...
```



Arrays

The diagram illustrates an array A with 8 elements, indexed from 0 to 7. Below the array, its elements are labeled as $A[0], A[1], A[2], A[3], A[4], A[5], A[6], A[7]$. The indices 0 through 7 are also shown below the array.

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
}
```

The diagram illustrates an array A with 1000 elements, indexed from 0 to 999. Below the array, its elements are labeled as $A[0], A[1], A[2], A[3], A[4], \dots, A[996], A[999]$.

Arrays

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

void main(void)
{
    int intArr [10] = {4,5,6,6} ;
    cout << intArr << endl ;
    system("pause");
}
```

When printing the array it prints 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 ;)

0045F008
Press any key to continue

The array "variable" is the location of its first element in memory

Arrays

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

void main(void)
{
    int intArr [10] = {4,5,6,6} ;
    cout << intArr[0] << endl ;
    system("pause");
}
```

When printing the array it prints it's location on memory by default.

Just the $(char arr [1])$
Don't do that!
It prints the char contained in it!
We'll see it in minutes ;)

* 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<=4 ; i++)
    {
        cout << "A[" << i << "] = " << A[i] << endl ;
    }
}

A[0] = -1
A[1] = 2092806
A[2] = 1040544
A[3] = 0
A[4] = 1523066256
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]
A[1]
A[2]
A[3]
A[4]
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] = 0
A[2] = 0
A[3] = 0
A[4] = 0
A[5] = 0
A[6] = 0
A[7] = 0
A[8] = 0
A[9] = 0
A[10] = 0
Press any key to continue . . .
```

Arrays

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

void main(void)
{
    int A [5];
    A[0] = 0;
    A[0] = 34;
    A[2] = 2;
    A[2] = 1;
    A[2+1] = 5;
    A[1] = 7;
    A[4] = 6;
    for (int i = 0 ; i<=4 ; i++)
    {
        cout << "A[" << i << "] = " << A[i] << endl;
    }
}

A[0] = 0
A[1] = 7
A[2] = 1
A[3] = 5
A[4] = 6
Press any key to continue...
```

Arrays

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

void main(void)
{
    int A [5];
    A[0] = 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+1] << endl;
    }
}

A[0] = 5
A[1] = 7
A[2] = 0
A[3] = 6
A[4] = 6
Press any key to continue...
```

Arrays

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

void main(void)
{
    int A [5];
    A[0] = 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] = 1337023180
A[0] = 0
A[1] = 7
A[2] = 0
A[3] = 5
A[4] = 6
Press any key to continue...
```

Arrays

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

void main(void)
{
    int A [5];
    A[0] = 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] = 1337023180
A[0] = 0
A[1] = 1
A[2] = 1
A[3] = 5
A[4] = 6
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;
    }
}

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] = 133799999
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 << "The outputted array is : " << endl ;
    cout << "The outputted array is : " << endl ;
    for (int i=0 ; i<5 ; i++)
    {
        cout << a[i] << endl ;
    }
}

The outputted array is :
0
2
4
6
8
The outputted array is :
```

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 - local variable 'a' must have static storage duration
```

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 conversion from double to int

Arrays

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

void main(void)
{
    int x = 44. ;
}
```

Console: >>>x

But with warning for converting from double to int ...
Possible loss of data



<< Note [You should know why] >>

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
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;
}
```

Compiled error... Inconsistent type 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
31
35
39
43
47
51
55
59
63
67
71
75
79
83
87
91
95
99
103
107
111
115
119
123
127
131
135
139
143
147
151
155
159
163
167
171
175
179
183
187
191
195
199
203
207
211
215
219
223
227
231
235
239
243
247
251
255
259
263
267
271
275
279
283
287
291
295
299
303
307
311
315
319
323
327
331
335
339
343
347
351
355
359
363
367
371
375
379
383
387
391
395
399
403
407
411
415
419
423
427
431
435
439
443
447
451
455
459
463
467
471
475
479
483
487
491
495
499
503
507
511
515
519
523
527
531
535
539
543
547
551
555
559
563
567
571
575
579
583
587
591
595
599
603
607
611
615
619
623
627
631
635
639
643
647
651
655
659
663
667
671
675
679
683
687
691
695
699
703
707
711
715
719
723
727
731
735
739
743
747
751
755
759
763
767
771
775
779
783
787
791
795
799
803
807
811
815
819
823
827
831
835
839
843
847
851
855
859
863
867
871
875
879
883
887
891
895
899
903
907
911
915
919
923
927
931
935
939
943
947
951
955
959
963
967
971
975
979
983
987
991
995
1003
1007
1011
1015
1019
1023
1027
1031
1035
1039
1043
1047
1051
1055
1059
1063
1067
1071
1075
1079
1083
1087
1091
1095
1103
1107
1111
1115
1119
1123
1127
1131
1135
1139
1143
1147
1151
1155
1159
1163
1167
1171
1175
1179
1183
1187
1191
1195
1203
1207
1211
1215
1219
1223
1227
1231
1235
1239
1243
1247
1251
1255
1259
1263
1267
1271
1275
1279
1283
1287
1291
1295
1299
1303
1307
1311
1315
1319
1323
1327
1331
1335
1339
1343
1347
1351
1355
1359
1363
1367
1371
1375
1379
1383
1387
1391
1395
1399
1403
1407
1411
1415
1419
1423
1427
1431
1435
1439
1443
1447
1451
1455
1459
1463
1467
1471
1475
1479
1483
1487
1491
1495
1499
1503
1507
1511
1515
1519
1523
1527
1531
1535
1539
1543
1547
1551
1555
1559
1563
1567
1571
1575
1579
1583
1587
1591
1595
1599
1603
1607
1611
1615
1619
1623
1627
1631
1635
1639
1643
1647
1651
1655
1659
1663
1667
1671
1675
1679
1683
1687
1691
1695
1703
1707
1711
1715
1719
1723
1727
1731
1735
1739
1743
1747
1751
1755
1759
1763
1767
1771
1775
1779
1783
1787
1791
1795
1799
1803
1807
1811
1815
1819
1823
1827
1831
1835
1839
1843
1847
1851
1855
1859
1863
1867
1871
1875
1879
1883
1887
1891
1895
1899
1903
1907
1911
1915
1919
1923
1927
1931
1935
1939
1943
1947
1951
1955
1959
1963
1967
1971
1975
1979
1983
1987
1991
1995
2003
2007
2011
2015
2019
2023
2027
2031
2035
2039
2043
2047
2051
2055
2059
2063
2067
2071
2075
2079
2083
2087
2091
2095
2099
2103
2107
2111
2115
2119
2123
2127
2131
2135
2139
2143
2147
2151
2155
2159
2163
2167
2171
2175
2179
2183
2187
2191
2195
2199
2203
2207
2211
2215
2219
2223
2227
2231
2235
2239
2243
2247
2251
2255
2259
2263
2267
2271
2275
2279
2283
2287
2291
2295
2299
2303
2307
2311
2315
2319
2323
2327
2331
2335
2339
2343
2347
2351
2355
2359
2363
2367
2371
2375
2379
2383
2387
2391
2395
2399
2403
2407
2411
2415
2419
2423
2427
2431
2435
2439
2443
2447
2451
2455
2459
2463
2467
2471
2475
2479
2483
2487
2491
2495
2499
2503
2507
2511
2515
2519
2523
2527
2531
2535
2539
2543
2547
2551
2555
2559
2563
2567
2571
2575
2579
2583
2587
2591
2595
2599
2603
2607
2611
2615
2619
2623
2627
2631
2635
2639
2643
2647
2651
2655
2659
2663
2667
2671
2675
2679
2683
2687
2691
2695
2699
2703
2707
2711
2715
2719
2723
2727
2731
2735
2739
2743
2747
2751
2755
2759
2763
2767
2771
2775
2779
2783
2787
2791
2795
2799
2803
2807
2811
2815
2819
2823
2827
2831
2835
2839
2843
2847
2851
2855
2859
2863
2867
2871
2875
2879
2883
2887
2891
2895
2899
2903
2907
2911
2915
2919
2923
2927
2931
2935
2939
2943
2947
2951
2955
2959
2963
2967
2971
2975
2979
2983
2987
2991
2995
3003
3007
3011
3015
3019
3023
3027
3031
3035
3039
3043
3047
3051
3055
3059
3063
3067
3071
3075
3079
3083
3087
3091
3095
3099
3103
3107
3111
3115
3119
3123
3127
3131
3135
3139
3143
3147
3151
3155
3159
3163
3167
3171
3175
3179
3183
3187
3191
3195
3199
3203
3207
3211
3215
3219
3223
3227
3231
3235
3239
3243
3247
3251
3255
3259
3263
3267
3271
3275
3279
3283
3287
3291
3295
3299
3303
3307
3311
3315
3319
3323
3327
3331
3335
3339
3343
3347
3351
3355
3359
3363
3367
3371
3375
3379
3383
3387
3391
3395
3399
3403
3407
3411
3415
3419
3423
3427
3431
3435
3439
3443
3447
3451
3455
3459
3463
3467
3471
3475
3479
3483
3487
3491
3495
3499
3503
3507
3511
3515
3519
3523
3527
3531
3535
3539
3543
3547
3551
3555
3559
3563
3567
3571
3575
3579
3583
3587
3591
3595
3599
3603
3607
3611
3615
3619
3623
3627
3631
3635
3639
3643
3647
3651
3655
3659
3663
3667
3671
3675
3679
3683
3687
3691
3695
3699
3703
3707
3711
3715
3719
3723
3727
3731
3735
3739
3743
3747
3751
3755
3759
3763
3767
3771
3775
3779
3783
3787
3791
3795
3799
3803
3807
3811
3815
3819
3823
3827
3831
3835
3839
3843
3847
3851
3855
3859
3863
3867
3871
3875
3879
3883
3887
3891
3895
3899
3903
3907
3911
3915
3919
3923
3927
3931
3935
3939
3943
3947
3951
3955
3959
3963
3967
3971
3975
3979
3983
3987
3991
3995
4003
4007
4011
4015
4019
4023
4027
4031
4035
4039
4043
4047
4051
4055
4059
4063
4067
4071
4075
4079
4083
4087
4091
4095
4099
4103
4107
4111
4115
4119
4123
4127
4131
4135
4139
4143
4147
4151
4155
4159
4163
4167
4171
4175
4179
4183
4187
4191
4195
4199
4203
4207
4211
4215
4219
4223
4227
4231
4235
4239
4243
4247
4251
4255
4259
4263
4267
4271
4275
4279
4283
4287
4291
4295
4299
4303
4307
4311
4315
4319
4323
4327
4331
4335
4339
4343
4347
4351
4355
4359
4363
4367
4371
4375
4379
4383
4387
4391
4395
4399
4403
4407
4411
4415
4419
4423
4427
4431
4435
4439
4443
4447
4451
4455
4459
4463
4467
4471
4475
4479
4483
4487
4491
4495
4499
4503
4507
4511
4515
4519
4523
4527
4531
4535
4539
4543
4547
4551
4555
4559
4563
4567
4571
4575
4579
4583
4587
4591
4595
4599
4603
4607
4611
4615
4619
4623
4627
4631
4635
4639
4643
4647
4651
4655
4659
4663
4667
4671
4675
4679
4683
4687
4691
4695
4699
4703
4707
4711
4715
4719
4723
4727
4731
4735
4739
4743
4747
4751
4755
4759
4763
4767
4771
4775
4779
4783
4787
4791
4795
4799
4803
4807
4811
4815
4819
4823
4827
4831
4835
4839
4843
4847
4851
4855
4859
4863
4867
4871
4875
4879
4883
4887
4891
4895
4899
4903
4907
4911
4915
4919
4923
4927
4931
4935
4939
4943
4947
4951
4955
4959
4963
4967
4971
4975
4979
4983
4987
4991
4995
5003
5007
5011
5015
5019
5023
5027
5031
5035
5039
5043
5047
5051
5055
5059
5063
5067
5071
5075
5079
5083
5087
5091
5095
5099
5103
5107
5111
5115
5119
5123
5127
5131
5135
5139
5143
5147
5151
5155
5159
5163
5167
5171
5175
5179
5183
5187
5191
5195
5199
5203
5207
5211
5215
5219
5223
5227
5231
5235
5239
5243
5247
5251
5255
5259
5263
5267
5271
5275
5279
5283
5287
5291
5295
5299
5303
5307
5311
5315
5319
5323
5327
5331
5335
5339
5343
5347
5351
5355
5359
5363
5367
5371
5375
5379
5383
5387
5391
5395
5399
5403
5407
5411
5415
5419
5423
5427
5431
5435
5439
5443
5447
5451
5455
5459
5463
5467
5471
5475
5479
5483
5487
5491
5495
5499
5503
5507
5511
5515
5519
5523
5527
5531
5535
5539
5543
5547
5551
5555
5559
5563
5567
5571
5575
5579
5583
5587
5591
5595
5599
5603
5607
5611
5615
5619
5623
5627
5631
5635
5639
5643
5647
5651
5655
5659
5663
5667
5671
5675
5679
5683
5687
5691
5695
5699
5703
5707
5711
5715
5719
5723
5727
5731
5735
5739
5743
5747
5751
5755
5759
5763
5767
5771
5775
5779
5783
5787
5791
5795
5799
5803
5807
5811
5815
5819
5823
5827
5831
5835
5839
5843
5847
5851
5855
5859
5863
5867
5871
5875
5879
5883
5887
5891
5895
5899
5903
5907
5911
5915
5919
5923
5927
5931
5935
5939
5943
5947
5951
5955
5959
5963
5967
5971
5975
5979
5983
5987
5991
5995
6003
6007
6011
6015
6019
6023
6027
6031
6035
6039
6043
6047
6051
6055
6059
6063
6067
6071
6075
6079
6083
6087
6091
6095
6099
6103
6107
6111
6115
6119
6123
6127
6131
6135
6139
6143
6147
6151
6155
6159
6163
6167
6171
6175
6179
6183
6187
6191
6195
6199
6203
6207
6211
6215
6219
6223
6227
6231
6235
6239
6243
6247
6251
6255
6259
6263
6267
6271
6275
6279
6283
6287
6291
6295
6299
6303
6307
6311
6315
6319
6323
6327
6331
6335
6339
6343
6347
6351
6355
6359
6363
6367
6371
6375
6379
6383
6387
6391
6395
6399
6403
6407
6411
6415
6419
6423
6427
6431
6435
6439
6443
6447
6451
6455
6459
6463
6467
6471
6475
6479
6483
6487
6491
6495
6499
6503
6507
6511
6515
6519
6523
6527
6531
6535
6539
6543
6547
6551
6555
6559
6563
6567
6571
6575
6579
6583
6587
6591
6595
6599
6603
6607
6611
6615
6619
6623
6627
6631
6635
6639
6643
6647
6651
6655
6659
6663
6667
6671
6675
6679
6683
6687
6691
6695
6699
6703
6707
6711
6715
6719
6723
6727
6731
6735
6739
6743
6747
6751
6755
6759
6763
6767
6771
6775
6779
6783
6787
6791
6795
6799
6803
6807
6811
6815
6819
6823
6827
6831
6835
6839
6843
6847
6851
6855
6859
6863
6867
6871
6875
6879
6883
6887
6891
6895
6899
6903
6907
6911
6915
6919
6923
6927
6931
6935
6939
6943
6947
6951
6955
6959
6963
6967
6971
6975
6979
6983
6987
6991
6995
7003
7007
7011
7015
7019
7023
7027
7031
7035
7039
7043
7047
7051
7055
7059
7063
7067
7071
7075
7079
7083
7087
7091
7095
7099
7103
7107
7111
7115
7119
7123
7127
7131
7135
7139
7143
7147
7151
7155
7159
7163
7167
7171
7175
7179
7183
7187
7191
7195
7199
7203
7207
7211
7215
7219
7223
7227
7231
7235
7239
7243
7247
7251
7255
7259
7263
7267
7271
7275
7279
7283
7287
7291
7295
7299
7303
7307
7311
7315
7319
7323
7327
7331
7335
7339
7343
7347
7351
7355
7359
7363
7367
7371
7375
7379
7383
7387
7391
7395
7399
7403
7407
7411
7415
7419
7423
7427
7431
7435
7439
7443
7447
7451
7455
7459
7463
7467
7471
7475
7479
7483
7487
7491
7495
7499
7503
7507
7511
7515
7519
7523
7527
7531
7535
7539
7543
7547
7551
7555
7559
7563
7567
7571
7575
7579
7583
7587
7591
7595
7599
7603
7607
7611
7615
7619
7623
7627
7631
7635
7639
7643
7647
7651
7655
7659
7663
7667
7671
7675
7679
7683
7687
7691
7695
7699
7703
7707
7711
7715
7719
7723
7727
7731
7735
7739
7743
7747
7751
7755
7759
7763
7767
7771
7775
7779
7783
7787
7791
7795
7799
7803
7807
7811
7815
7819
7823
7827
7831
7835
7839
7843
7847
7851
7855
7859
7863
7867
7871
7875
7879
7883
7887
7891
7895
7899
7903
7907
7911
7915
7919
7923
7927
7931
7935
7939
7943
7947
7951
7955
7959
7963
7967
7971
7975
7979
7983
7987
7991
7995
7999
8003
8007
8011
8015
8019
8023
8027
8031
8035
8039
8043
8047
8051
8055
8059
8063
8067
8071
8075
8079
8083
8087
8091
8095
8099
8103
8107
8111
8115
8119
8123
8127
8131
8135
8139
8143
8147
8151
8155
8159
8163
8167
8171
8175
8179
8183
8187
8191
8195
8199
8203
8207
8211
8215
8219
8223
8227
8231
8235
8239
8243
8247
8251
8255
8259
8263
8267
8271
8275
8279
8283
8287
8291
8295
8299
8303
8307
8311
8315
8319
8323
8327
8331
8335
8339
8343
8347
8351
8355
8359
8363
8367
8371
8375
8379
8383
8387
8391
8395
8399
8403
8407
8411
8415
8419
8423
8427
8431
8435
8439
8443
8447
8451
8455
8459
8463
8467
8471
8475
8479
8483
8487
8491
8495
8499
8503
8507
8511
8515
8519
8523
8527
8531
8535
8539
8543
8547
8551
8555
8559
8563
8567
8571
8575
8579
8583
8587
8591
8595
8599
8603
8607
8611
8615
8619
8623
8627
8631
8635
8639
8643
8647
8651
8655
8659
8663
8667
8671
8675
8679
8683
8687
8691
8695
8699
8703
8707
8711
8715
8719
8723
8727
8731
8735
8739
8743
8747
8751
8755
8759
8763
8767
8771
8775
8779
8783
8787
8791
8795
8799
8803
8807
8811
8815
8819
8823
8827
8831
8835
8839
8843
8847
8851
8855
8859
8863
8867
8871
8875
8879
8883
8887
8891
8895
8899
8903
8907
8911
8915
8919
8923
8927
8931
8935
8939
8943
8947
8951
8955
8959
8963
8967
8971
8975
8979
8983
8987
8991
8995
9003
9007
9011
9015
9019
9023
9027
9031
9035
9039
9043
9047
9051
9055
9059
9063
9067
```

Code Checking - 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 ;
}
```

0
4
6
8
10
12
14
16
18
20

Code Checking - 01

```
#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 ;
}
```

0
10

String!**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;
    }
}
```

100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333

Strings using Arrays

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

void main(void)
{
    char str1 [] = "hello";
    char str2 [] = {'h','e','l','l','o'};
    (str1==str2) ? cout << "Yuppy !" : cout << "Bo Bo :(" ;
}
```

No No :(

Strings using Arrays

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

void main(void)
{
    char str1 [] = "hello";
    char str2 [] = {'h','e','l','l','o'};
    (str1==str2) ? cout << "Yuppy !" : cout << "Bo Bo :(" ;
}
```

Bo Bo :(

Why is it so?

Strings using Arrays

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

void main(void)
{
    char str1 [] = "hello";
    char str2 [] = {'h','e','l','l','o'};
    (str1==str2) ? cout << "Yuppy !" : cout << "Bo Bo :(" ;
}
```

No No :(

Why is it so?

One 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'};
    (str1==str2) ? cout << "Yuppy !" : cout << "Bo Bo :(" ;
    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;
    }
}
```

6	3	0
10	4	10
10	1	10
20	0	20
10	8	10
11	1	11
0		0

Strings using Arrays

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

void main(void)
{
    char str1 [] = "Hello";
    char str2 [] = {'H','e','l','l','o'};
    if(str1==str2) cout << "Pappy !";
    cout << "Bo Bo :(";
    cout << endl;
    for (int i = 0 ; i < 6 ; i++)
    {
        cout << str1[i];
    }
    cout << endl;
    for (int i = 0 ; i < 6 ; i++)
    {
        cout << str2[i];
    }
    cout << endl;
}
```

Bo Bo :(
Hello
Hello
Hello
Hello
Hello
Hello

Strings using Arrays

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

void main(void)
{
    char str1 [10];
    cin >> str1;
    cout << "The imputed string as a whole : , is " << str1 << endl;
}


```

Bo Bo
The imputed string as a whole : , is Hello
Press any key to continue

Strings using Arrays

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

void main(void)
{
    char str1 [10];
    cin >> str1;
    cout << "The imputed string as a whole : , is " << str1 << endl;
}


```

ooooooooooooooooooo
The imputed string as a whole : , is oooooooooooo
Press any key to continue

Compiler and print bar width:Run-time error '0'! - Overflow!

Strings using Arrays

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

void main(void)
{
    char str1 [10];
    cin >> str1;
    cout << "The imputed string as a whole : , is " << str1 << endl;
}


```

Bo Bo
The imputed string as a whole : , is Java
Press any key to continue

Steps at first whitespace character

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 ;
}

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 prints
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 0043F950
Press any key to continue
```

When printing the array it prints
it's location on memory by
default

Just the char array
(char charArr [1])
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 prints
it's location on memory by
default

Just the char array
(char charArr [1])
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!
You 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 0010111234
Press any key (F10) 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();
}
```

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();
}

Press any key (F10) to continue . . .
```

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;
    }
    cout << "Auto Arr after +2 to all elements : "
        << endl;
    for (int i=0 ; i<3 ; i++)
    {
        cout << "Arr1[" << i << "]=" <<
            (Arr1[i]+2) << endl;
    }
}

void main(void)
{
    AutoArr();
}

```

Press any key to continue .

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;
    }
    cout << "Auto Arr after +2 to all elements : "
        << endl;
    for (int i=0 ; i<3 ; i++)
    {
        cout << "Arr1[" << i << "]=" <<
            (Arr1[i]+2) << endl;
    }
}

void main(void)
{
    AutoArr();
}

```

Press any key to continue .

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;
    }
    cout << "Auto Arr after +2 to all elements : "
        << endl;
    for (int i=0 ; i<3 ; i++)
    {
        cout << "Arr1[" << i << "]=" <<
            (Arr1[i]+2) << endl;
    }
}

void main(void)
{
    AutoArr();
    AutoArr();
}

```

Press any key to continue .

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;
    }
    cout << "Auto Arr after +2 to all elements : "
        << endl;
    for (int i=0 ; i<3 ; i++)
    {
        cout << "Arr1[" << i << "]=" <<
            (Arr1[i]+2) << endl;
    }
}

void main(void)
{
    AutoArr();
}

```

Press any key to continue .

Strings using Arrays – static

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

void StaticArr()
{
    static int Arr1 [3] = {1,2,3};
    cout << "Arr : " << endl;
    for (int i=0 ; i<3 ; i++)
    {
        cout << "Arr1[" << i << "]=" << Arr1[i] << endl;
    }
    cout << "Arr after +2 to all elements :" << endl;
    for (int i=0 ; i<3 ; i++)
    {
        cout << "Arr1[" << i << "]=" << (Arr1[i]+2) << endl;
    }
}

void main(void)
{
    StaticArr();
    StaticArr();
}
```

Strings using Arrays – static

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

void StaticArr()
{
    static int Arr1 [3] = {1,2,3};
    cout << "Arr : " << endl;
    for (int i=0 ; i<3 ; i++)
    {
        cout << "Arr1[" << i << "]=" << Arr1[i] << endl;
    }
    cout << "Arr after +2 to all elements :" << endl;
    for (int i=0 ; i<3 ; i++)
    {
        cout << "Arr1[" << i << "]=" << (Arr1[i]+2) << endl;
    }
}

void main(void)
{
    StaticArr();
    StaticArr();
}
```

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);
}
```

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 were 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 PassingArrRef(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 , can't be referenced call by
// its 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);
}

// Passing arrays by value is passing it with reference
// not by value ...

```

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 pass arrays with values , can't be referenced call by
// its self

```

Multiple subscripted Arrays

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

void main(void)
{
    int Arr[3][4] ; // 3 rows, 4 columns
}
```

	Column 0	Column 1	Column 2	Column 3
Row 0	at 0 1 1 0 1	at 0 1 1 1 1	at 0 1 1 2 1	at 0 1 1 3 1
Row 1	at 1 1 1 0 1	at 1 1 1 1 1	at 1 1 1 2 1	at 1 1 1 3 1
Row 2	at 2 1 1 0 1	at 2 1 1 1 1	at 2 1 1 2 1	at 2 1 1 3 1

Diagram illustrating multiple subscripting:

- Row index: Row 0, Row 1, Row 2
- Column index: Column 0, Column 1, Column 2, Column 3
- Subscripting: arr[Row][Column]

Multiple subscripted Arrays

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

void main(void)
{
    int Arr[2][3] ;
}
```

Compiler error: ... Should be [1][1]

Multiple subscripted Arrays

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

void main(void)
{
    int Arr [][] ;
}
```

Compiler error: missing subscripted size

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

void main(void)
{
    int Arr [][3] ;
}
```

Compiler error: missing subscripted size

Multiple subscripted Arrays

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

void main(void)
{
    int Arr[2][3] = { {2,3,5} , {4,1,3} } ;
}
```

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} } ;
}
```

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} , {4,0} } ;
    cout << Arr[2][3] << endl ;
}
```

2 3 5
4 0

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 ;
}
```

2

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

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]);
}
```

2

Multiple subscripted Arrays passing to function

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

void ArrFun(int A [1][3])
{
    cout << A [1][1] ;
}

void main(void)
{
    int Arr[2][3] = { {2,3,5}, {4,3} } ;
    ArrFun(Arr);
}
```

2

No compiler error for missing the first subscript in the function's prototype ... like single subscripted arrays

Multiple subscripted Arrays passing to function

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

void ArrFun(int A [][1])
{
    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] ;
}
```

The diagram illustrates the memory layout of a 2x3 integer array. It shows two horizontal rows of memory cells. The first row, labeled 'Row #0', has three cells and is pointed to by the address 5000. The second row, labeled 'Row #1', has two cells and is pointed to by the address 5005. Arrows from the pointer labels indicate the starting address of each row.

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 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()
()
```

Unresolved external - 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 ;
                           // 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 0 ;
}
```

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
6

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
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 << endl ;
    return 0 ;
}
```

6
6

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
6

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 ;
}
```

Compilation 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 ;
}
```

Execution 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 ;
}
```

Compilation error - x is undefined

Functions

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

void Mult(void);

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

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

Execution error - overflow (16-bit word) detected

Functions

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

void Mult(void);

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

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

Compilation error - will return "void"!

Functions

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

void Mult(void);

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

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

Program error - can't understand function.

Functions

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

void Mult(void);

void main(void)
{
    Mult() ;
}

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

No less... everything's working!

Functions

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

void Mult(void);

void main()
{
    Mult() ;
}

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

No less... everything's working!

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 ;
}

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

Compiler Parsing
Compiler error ... missing function's prototype

Functions

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

void main()
{
    Mult() ;
}

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

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

Compiler Parsing
Compiler error ... missing function's prototype

Functions

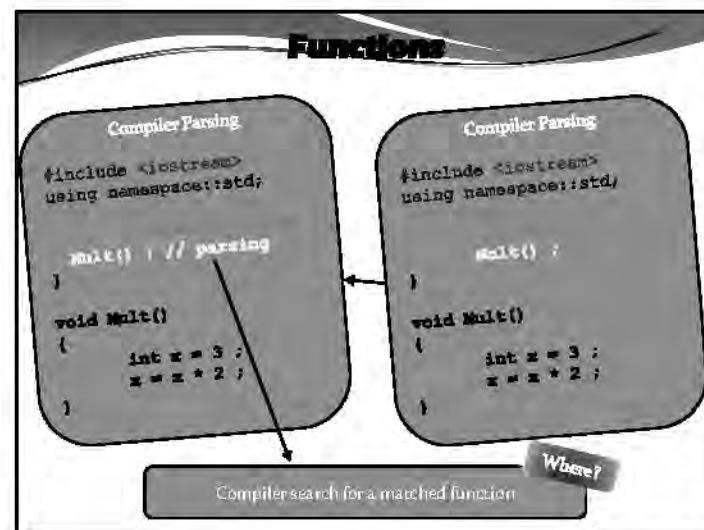
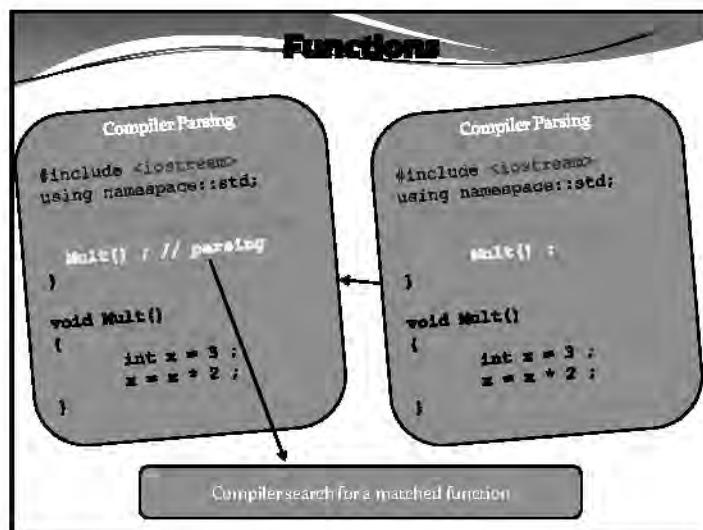
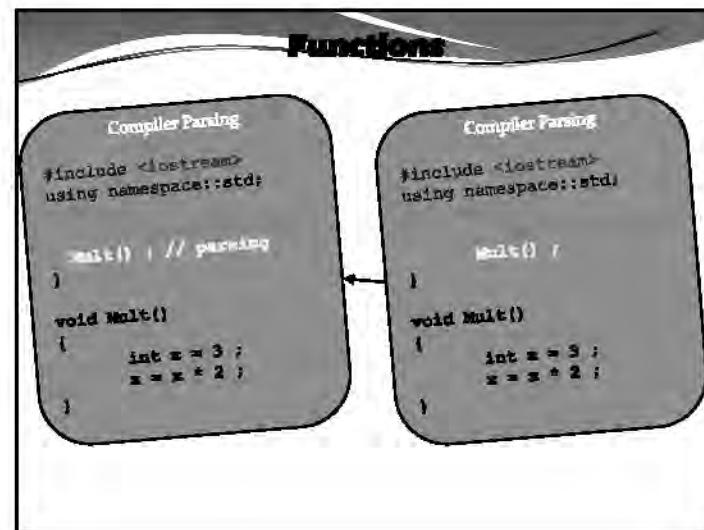
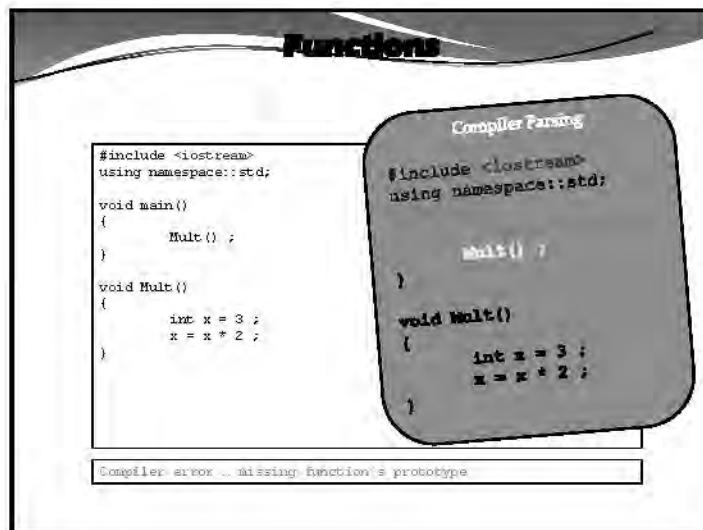
```
#include <iostream>
using namespace::std;

void main()
{
    Mult() ;
}

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

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

Compiler Parsing
Compiler error ... missing function's prototype



Functions

```

Compiler Parsing
#include <iostream>
using namespace::std;

    Mult() ; // PENDING
}

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 ... 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 << endl ;
}
```

6.

(Missing = 11 because of other code problems)

Functions

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

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

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

Compiling ... unresolvable nested declarations

6.

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: undefined reference to 'Mult'.

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 ;
}
```

D

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 'int'.

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 ;
    }
}
```

D

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;
    }
}
```

Unresolved external... can't resolve all function calls in this scope

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:

No

o

Like command

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
Please enter user command:

Functions

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

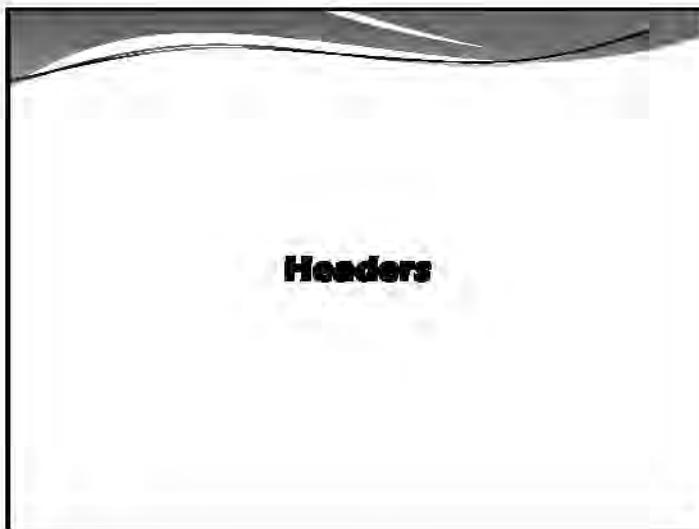
void Crazy1();
void Crazy2(int);

void main(void)
{
    Crazy1();
}

void Crazy1()
{
    int x = 3 ;
    Crazy2(x);
    cout << x ;
}

void Crazy2(int x )
{
    x+=6 ;
    cout << x << endl ;
}
```

100 pts



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

- The Concept of headers

Headers – Math Library

- To use it
 - Include <cmath>

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

void main(void)
{
    cout << sqrt(100.0) << endl;
}
```

1.0

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... can't use integer "no overloaded function"

Headers – Math Library

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

void main(void)
{
    cout << sqrt(sqrt(100.0)) << endl;
}
```

3.1618

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;
}
```

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 ;
}
```

Output:

Random Number:

Now, when compiling many times ...
Every time we have the same value !

0.1
0.1
0.1

rand()

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

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

1SE
1SE

Module Name: _____

rand()

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

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

1SE
1SE

Version 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 ;
}
```

1SE
1SE

- ## 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 ;
}
```

rand()

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

void main(void)
{
    int x = 3 ;
    srand (time());
    cout << x << endl ;
}
```

Compil et exécute plusieurs fois.

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
123

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
48
48

Not an appropriate srand() parameter leads to insufficient 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
```

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
16331
2607
1405
5450
20050
1816
20477
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 "return type"

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

Function

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

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

void main()
()
```

Compile 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 ; // permitted 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 ;
}
```

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 ;
}
```

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 ;
}
```

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
6

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 ;
}
```

Ungültige Syntax

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 ;
}
```

Closed in: 00000

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 ;
}
```

Closed in: 00000 - no associated

Functions

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

void Mult(void);

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

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

Compilieren: g++ -o funktion Funktion.cpp und dann doppelklicken

Functions

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

void Mult(void);

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

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

Ergebnis: 6
Ergebnis: 6 (durch das fehlende "return 0";)

Functions

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

void Mult(void);

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

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

Ergebnis: 6
Ergebnis: 6 (durch das fehlende "return 0";)

Functions

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

void Mult(void);

void main(void)
{
    Mult() ;
}

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

Ergebnis: 6
Ergebnis: 6 (durch das fehlende "return 0";)

Functions

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

void Mult(void);

void main()
{
    Mult();
}

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

At least one everything's missing :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 ... no matching return type

Functions

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

Mult(void);

void main()
{
    Mult();
}

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

Compiler error ... mismatched argument types

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 ;
}

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

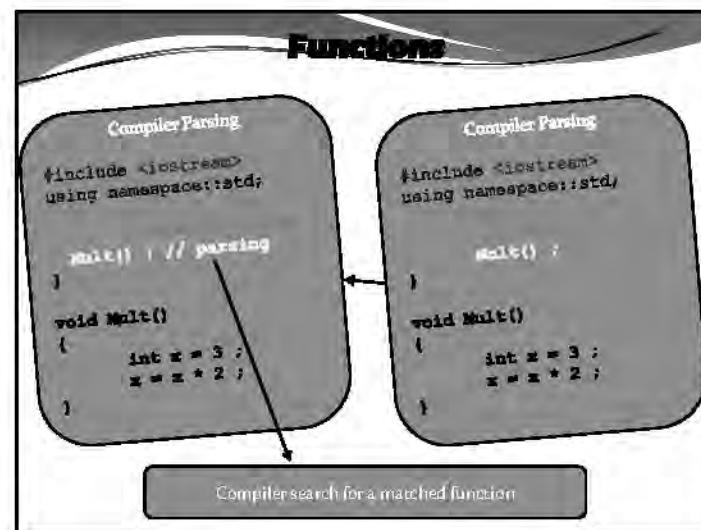
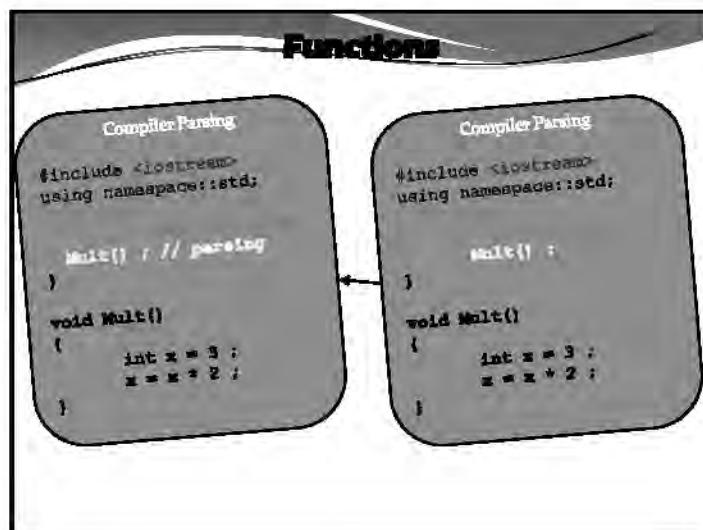
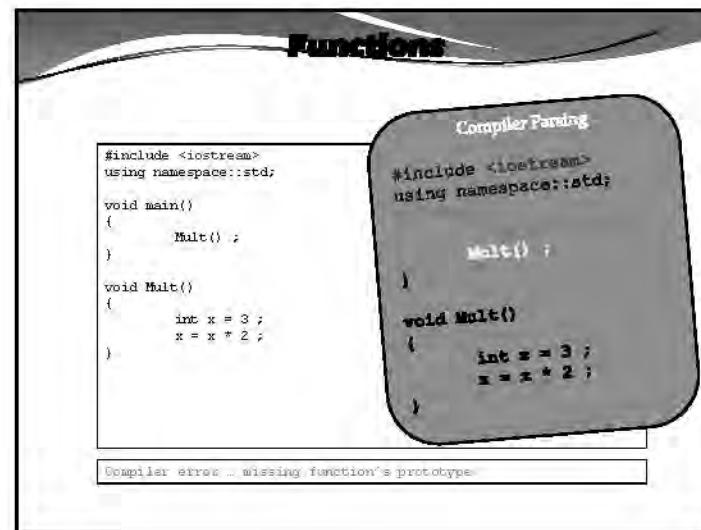
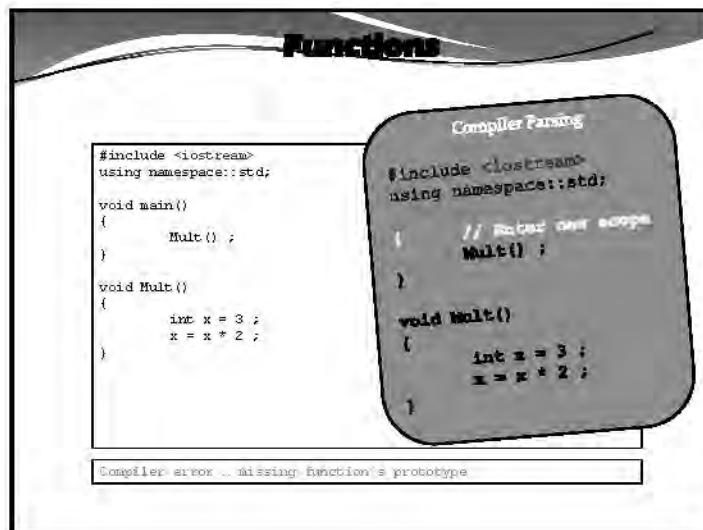
Compiler Parsing

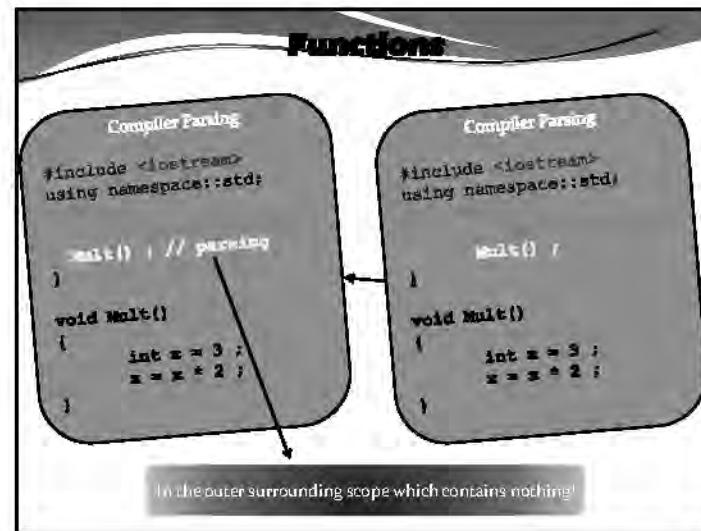
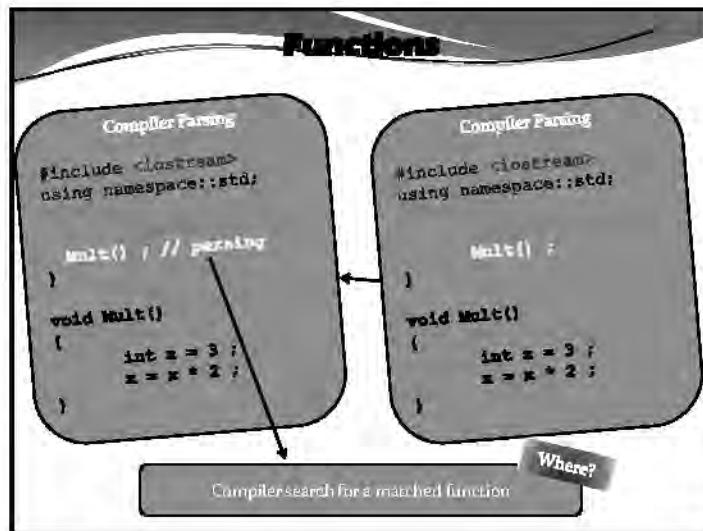
```
#include <iostream>
using namespace::std;
void main()
{
    mult() ;
}

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

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

Compiler error ... missing function's prototype





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 ;
}
```

Functions

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

int Mult(void);

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

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

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 << endl ;
}

cout
```

Output will be printed after the final statement.

Functions

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

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

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

Output will be printed by the 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 ;
}
```

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 ;
}
```

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 - mismatch 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 HeHe (int z)
    {
        return 0 ;
    }
}
```

D:\C\100\100\main.cpp(10): error C2059: syntax error: identifier 'HeHe'

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:

No -

E

Line 10

Functions

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

int Mult(int );
void main(void)
{
    for (int i = 0 ; i<=10 ; i++)
    {
        cout << Mult(i) << endl ;
    }
    cout << "hehe" << endl ;
    cout << endl ;
}

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

D:\C\100\100\main.cpp(10): error C2059: syntax error: identifier 'Mult'

D:\C\100\100\main.cpp(10): error C2059: syntax error: identifier 'Mult'

Functions

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

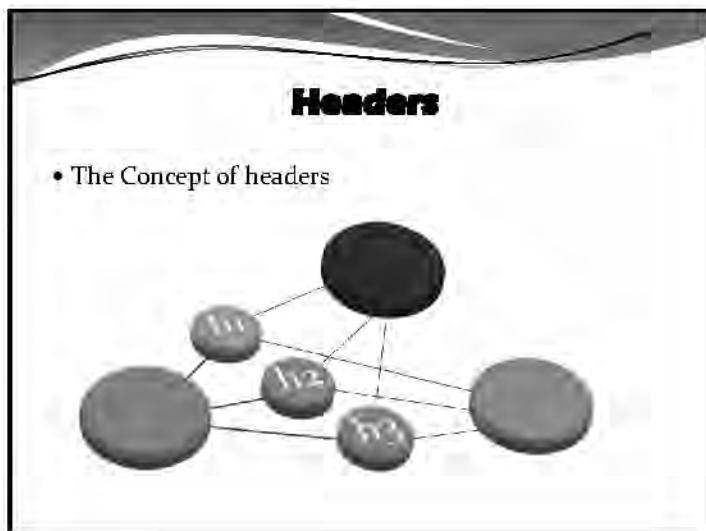
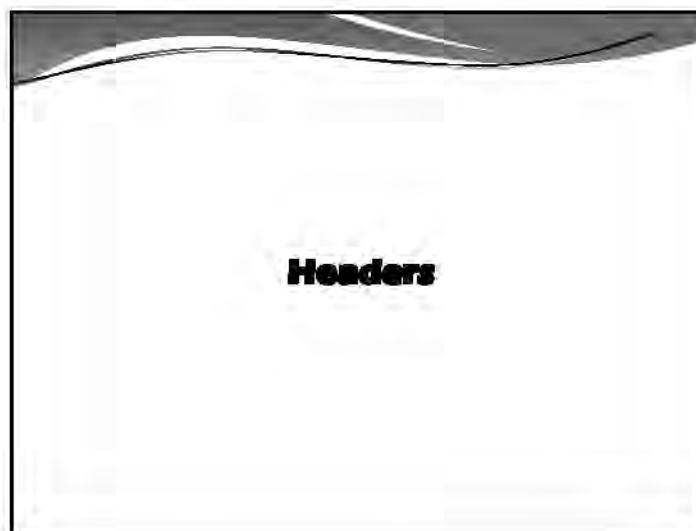
void Crazy1();
void Crazy2(int);

void main(void)
{
    Crazy1();
}

void Crazy1()
{
    int x = 3 ;
    Crazy2(x) ;
    cout << x ;
}

void Crazy2(int x)
{
    x+=6 ;
    cout << x << endl ;
}
```

100 ms.



- ## 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' not found

Headers – Math Library

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

void main(void)
{
    cout << sqrt(100) << endl;
}
```

Compiler error: undefined reference to function 'sqrt'

Headers – Math Library

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

void main(void)
{
    cout << sqrt(sqrt(100.0)) << endl;
}
```

1.16128

Headers – Math Library

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

void main(void)
{
    int x ;
    cout << sqrt(6*x) << endl;
}
```

Compiler error – result type integer too 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 ;
}

#1
Random Number:
Now, when compiling many times ...
Every time we have the same value !

#1.
#1.
#1.
```

rand()

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

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

#1
Random Number:
Now, when compiling many times ...

#1.
```

rand()

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

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

#1
#2
#3
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 ;
}

#1
#2
#3
```

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 ;
}
```

cout << "enter value for x" ;

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

189

*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() parameter leads to insufficient 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
}


```

for 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
16857
1507
14053
5950
30050
16164
20477
5146

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
Please input 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 ?
C:\Users\ASUS\OneDrive\Documents\CHAD\CHAD0100

Playing with scopes I

- 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 says:

This is inner block

Playing with scopes I

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

void main(void)
{
    int x = 9 , y = 3 ;
    {
        cout << x << endl ;
    }
}
```

Compiler says:

Playing with scopes I

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

void main(void)
{
    int x = 9 , y = 3 ;
    (
        cout << x << endl ;
    )
    cout << x << endl;
}
```

Compiler says:

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 ;
    )
}
```

Compiler says:

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 ;
}

3
3
```

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 ;
}

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 ;
}

Using other scope - undeclared identifier
```

?10?

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 ;
}

3
```

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
    }
}
```

Compiler is here -> nothing in workspace
Waiting to compile... yes !

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();
}
```

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;
    }
}

```

72
15

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(3.14) = 4.00000
cos(x)	trigonometric cosine of x (x in radians)	cos(0) = 1.00000
exp(x)	exponential function e ^x	exp(1) = 2.71828
fabs(x)	absolute value of x	fabs(-1.1e-5) = 1.1e-5
floor(x)	rounds x to the largest integer not greater than x	floor(1.1e-5) = 1.1e-5
fmod(x, y)	remainder of x / a (floating point number)	fmod(1.1e-5, 1.1e-5) = 0.00000
log(x)	natural logarithm of x (base e)	log(2.71828) = 1.14770
log10(x)	logarithm of x (base 10)	log10(10) = 1.00000
pow(x, y)	calculated to power y (x) ^y	pow(2, -1) = 0.500
sin(x)	trigonometric sine of x (x in radians)	sin(1) = 0.84147
sqr(x)	square root of x	sqr(3.14) = 1.77828
tan(x)	trigonometric tangent of x (x in radians)	tan(1) = 1.55740

Functions – Part 2

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);
}
```

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);
}
```

Recursion I

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 overflow! "==" and not "=="!!!

So the output should 1 right!!!

But!!!! if(n==0) is the same as if(false) and thus it never enter the scope

250 pts

Recursion I

Compiler Parsing

```
int factorial(int n)
{
    if (n == 0)
        return 1;
    else
        return n*factorial(n-1);
}
```

Compiler Parsing

```
int factorial(int n)
{
    if (n == 0)
        return 1;
    else
        return n*factorial(n-1);
}
```

Recursion I

Compiler Parsing

```
if (n == 0)
    return 1;
else
    return n*factorial(n-1);
}
```

Compiler Parsing

```
if (n == 0)
    return 1;
else
    return n*factorial(n-1);
}
```

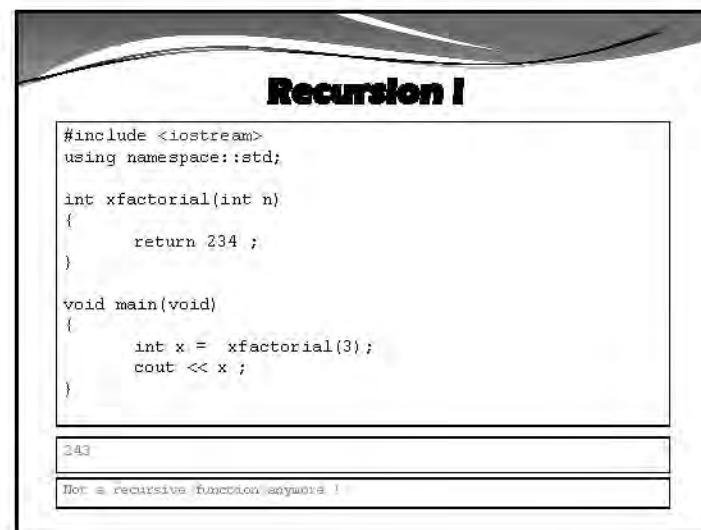
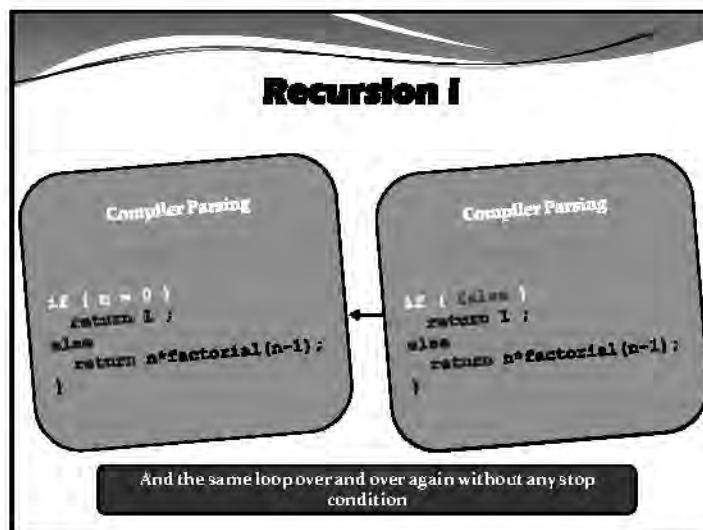
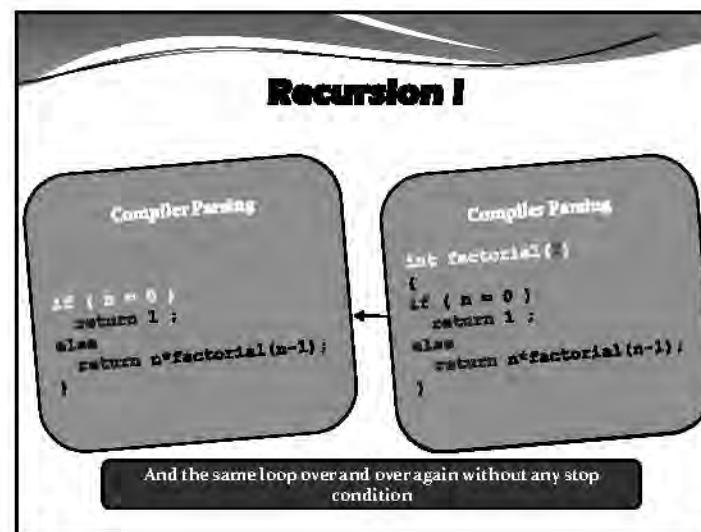
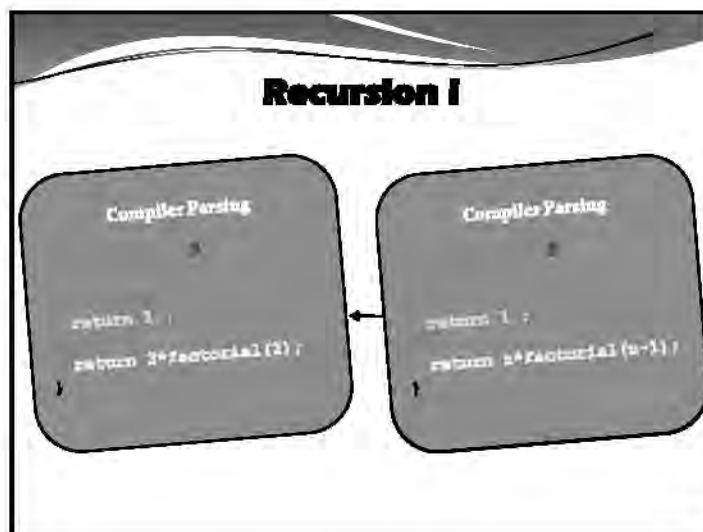
Recursion I

Compiler Parsing

```
return 1;
else
    return n*factorial(n-1);
}
```

Compiler Parsing

```
return 1;
else
    return n*factorial(n-1);
}
```



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 reduce 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
 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);
}
```

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);
}
```

Please 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 !

```
#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 3
this is a Recursive call . Number 2
this is a Recursive call . Number 1
this is a Recursive call . Number 0
Press any key to continue...
```

Recursion !

- Recursive function sum of numbers from i to n
- Discussion ...
 - How to do it ?
 - Finding the idea !
 - Rule to Rule !
 - We have ...
 - $1 + 2 + 3 + \dots + n$
 - The Rule is :
 - $n + \text{sum of the numbers from } i \text{ to } n-1 (\rightarrow \sum_{i=1}^{n-1})$
 - $n + \text{summation}(n-1)$
 - 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 ;
}

7
```

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 ;
}
```

12
Here a recursion will occur, infinitely.

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");
}
```

10 - Visual Studio 2010
Should Recurse 1 in first test!

```
#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 ;
}
```

Completed Step 1 - sum of small letters (0-9) tally one.)

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
 - $2^1 = 2 \cdot 2^0$ $2^2 = 2 \cdot 2^1$; ...
 - The Rule is:
 - $x^n = x^{n-1} \cdot x$
 - 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 ;
}
```

Output: 8
Time complexity: O(n)

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 ;
}
```

Output: 8
Time complexity: O(n)

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 returning with 0 !!!
Msg returning with 1 !!!
Msg returning with 2 !!!
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 returning with 0 !!!
Msg returning with 1 !!!
Msg returning with 2 !!!
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 0
This is recursive func .
Times are : 0
Msg called with 1
This is recursive func .
Times are : 1
Msg called with 2
This is recursive func .
Times are : 2
Msg called with 3
This is recursive func .
Times are : 3
Msg returning with 3 !!!
Msg returning with 2 !!!
Msg returning with 1 !!!
Msg returning with 0 !!!
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 ;
}
```

Output: Error ... missing one 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);
}
```

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);
}
```

Compiles successfully @ default value for parameter 2

When defaulting a parameter in a function , all other parameters - if found - in the function 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);
}
```

Default parameters

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

void print(int x , int n , int y )
{
    cout << x << endl ;
}

void main(void)
{
    print(2,2,4);
}
```

??

?? (3.0 min)

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
3
4

0x11 8 main

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
3

0x11 8 main

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);
}
```

Compiler error - no value given to variable y.

Host - did you mean 'y'?

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);
}

Compiler error ... cos we didn't default the y ...
Now... did u know why? :-)
```

Why the compiler couldn't do it?

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 error ... cos we didn't default the y ...
Now... did u know why? :-)
```

Compiler Parsing

```
print(2,3);
```

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, ?)

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, ?)

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 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)
```

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!

Compiler error - can't default define
Now... and it's known why...

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 **&** after 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 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 ;
}
```

*(Output : Before calling the function
In function before multiplication - 2
In function after multiplication - 8

In main , before calling the function - 2
8
In main , after calling the function - 2)*

Calling by value Vs calling by reference

```
#include <iostream>
using namespace::std;
int s = 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 ;
}
```

*(Output : Before calling the function - 2
In function before multiplication - 2
In function after multiplication - 8

In main , before calling the function - 2
8
In main , after calling the function - 2)*

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

Function Overloading

Function Overloading

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);
}
```

```
#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);
}
```

```
#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 x , float z)
{
    x = x * 2 ;
    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);
}
```

```
#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 x , float z)
{
    x = x * 2 ;
    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 << "DoGo !" << endl ;
}
```

True
False
DoGo!

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 ;
    }
}
```

5
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
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
10

Tweaked Control Structure I

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

void main(void)
{
    int x = 5 ;
    while (x != 10)
    {
        ++x;
        cout << x << endl ;
    }
}
```

5
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 ;
    }
}

25
25
25
25
25

The determinately
```

Tweaked Control Structure I

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

void main(void)
{
    int x = 5 ;
    while (x*x < 10)
    {
        cout << x*x << endl ;
    }
}

Missing do part
```

Tweaked Control Structure I

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

void main(void)
{
    int x = 5 ;
    while (cin)
    {
        cin >> x ;
        cout << x*x << endl ;
    }
}

1
2
3
4
5
6
7
8
9
10

Program exit
```

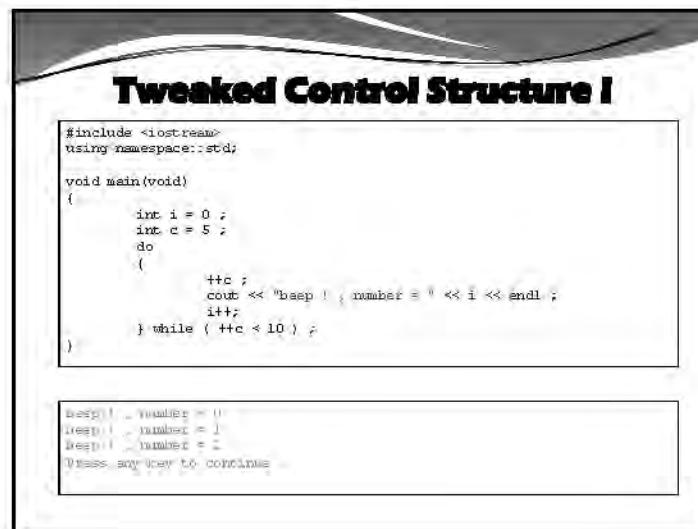
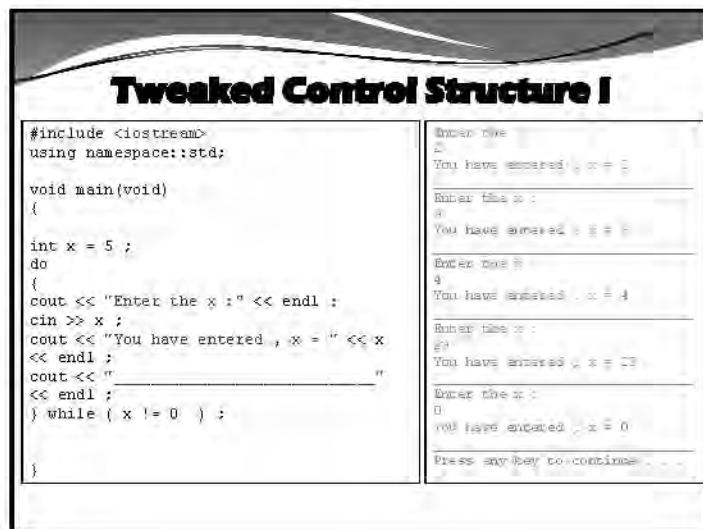
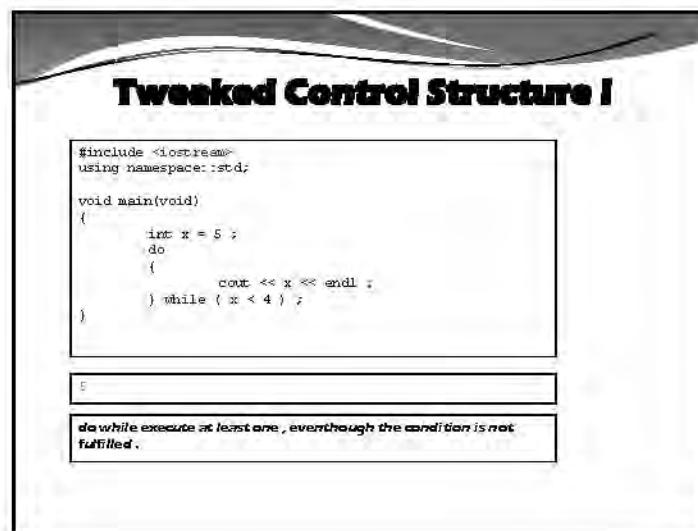
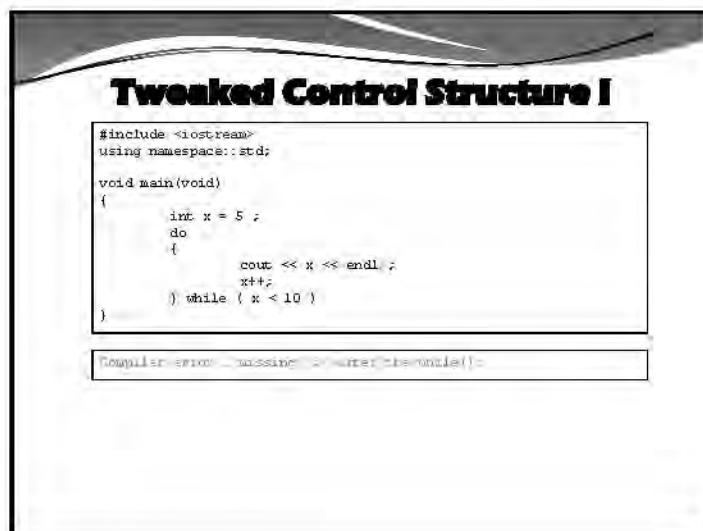
Tweaked Control Structure I

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

void main(void)
{
    int x = 5 ;
    do
    {
        cout << x << endl ;
        x++ ;
    } while ( x < 10 ) ;
}

1
2
3
4
5
6
7
8
9
10

Program exit
```



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
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

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
beep | number = 5
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 = 0
finished
Press any key to continue . . .

```

What is that?

Watch out for the semi colon ";" after the for statement . Can 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 ?

See we are still working in an 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 = 1
beep ! , number = 2
beep ! , number = 3
beep ! , number = 4
beep ! , number = 456890

Un-stoppable loop !
```

Tweaked Control Structure I

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

void main(void)
{
    int i = 66 ;
    int c = 5 ;
    for (int i = 0 ; i < ; 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
beep ! , number = 5
beep ! , number = 6
beep ! , number = 7
beep ! , number = 8
beep ! , number = 9
beep ! , number = 10
beep ! , number = 115840

Un-unstoppable 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 ;
}

beep ! , number = 0
beep ! , number = 1
beep ! , number = 2
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 , int j = 3 ; i<5 ; i++)
    {
        cout << "beep ! , number = " << j++ << 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
finished

Press any key to continue . . .

```

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 ;
    }
}
```

```
5  
4  
3  
2  
1
```

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  
0  
0  
0  
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  
1  
2
```

Tweaked Control Structure I

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

void main(void)
{
    for (int i = 0 ; i<5 ; i++)
    {
        cout << i << endl ;
        if ( i == 3)
        {
            break ;
        }
    }
}
```

0
1
2
3

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 ... = = woc ==

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 << "WEEEEE !!!!";
            break ;
        }
    }
}

0
1
2
WEEEEE !!!
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 << 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 << "WeeWeee" << endl ;
            continue ;
        }
    }
}

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 << "Meowwww" << endl ;
        }
    }
}

0
1
2
Meowwww
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 = 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 ;
    }
}

Enter x :
4
You have entered , x = 4
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 ;
            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;
    }
}

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 = 1
Not a 1 or 2
Press any key to continue
```

Cuz of forgetting break:

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
1
WOW , i can't believe it , you entered
1
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
WOW , i can't believe it , you entered x = 2
Not a 1 or 2
Press any key to continue
```

Arrays

C++ data types

Address

Simple

STRUCTURE 1

Pointer

enum

Array

References

Integral

Struct

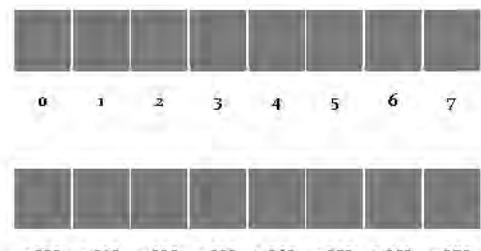
*Chair, Shove
in, long, be*

Union

Floating

Class

Arrays



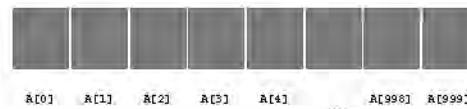
Note : **n**th element in position **n-1**

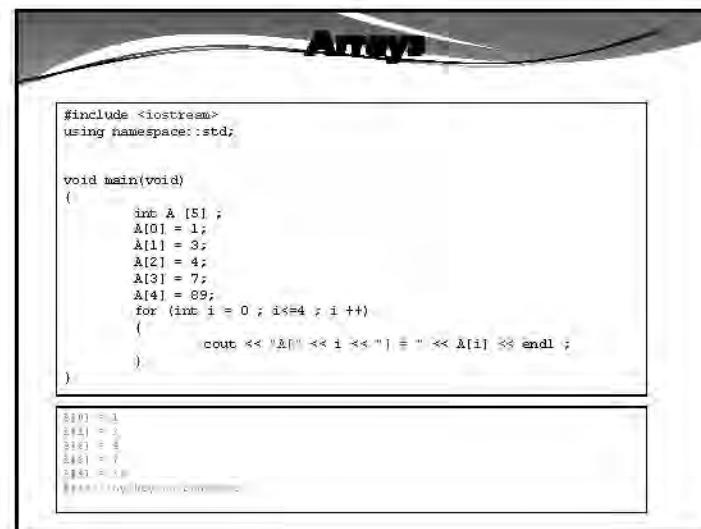
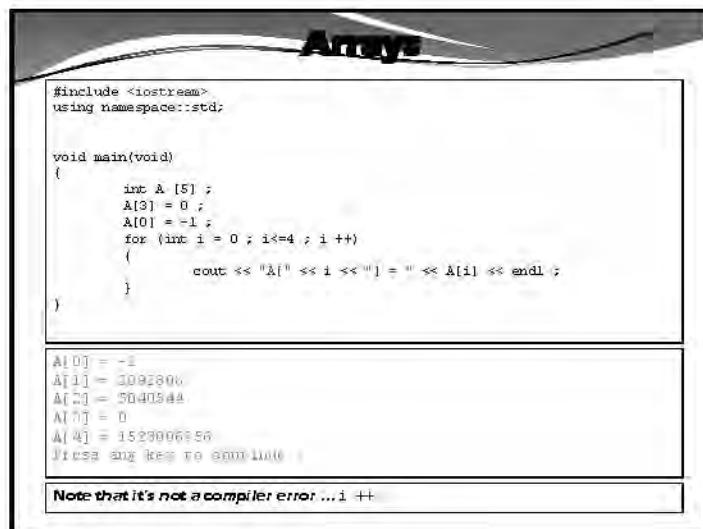
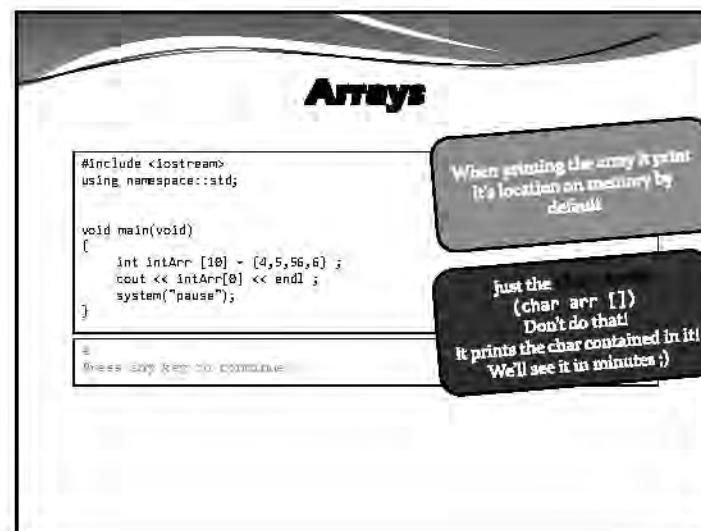
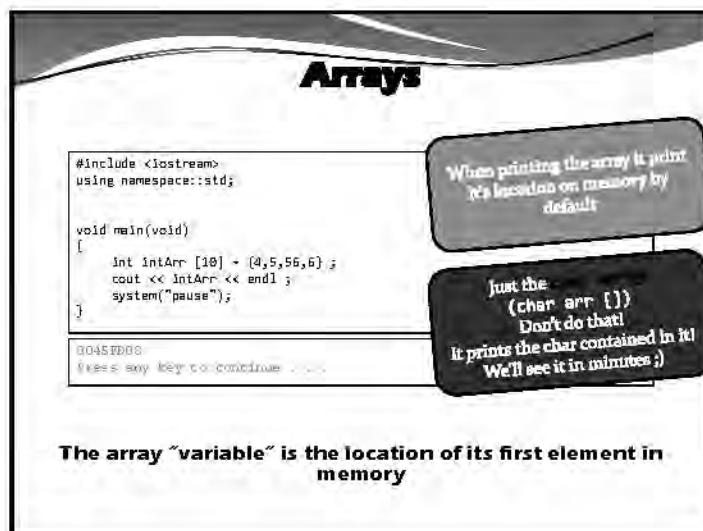
Arrays

- 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 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
A[5] = 34
```

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 ;
    }
}

A[0] = -1
A[1] = 0
A[2] = 0
A[3] = 0
A[4] = 0
A[5] = 0
A[6] = 0
A[7] = 0
A[8] = 0
A[9] = 0
A[10] = 0
```

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] = 6
A[3] = 2
A[4] = 5
```

Please enter the correct code.

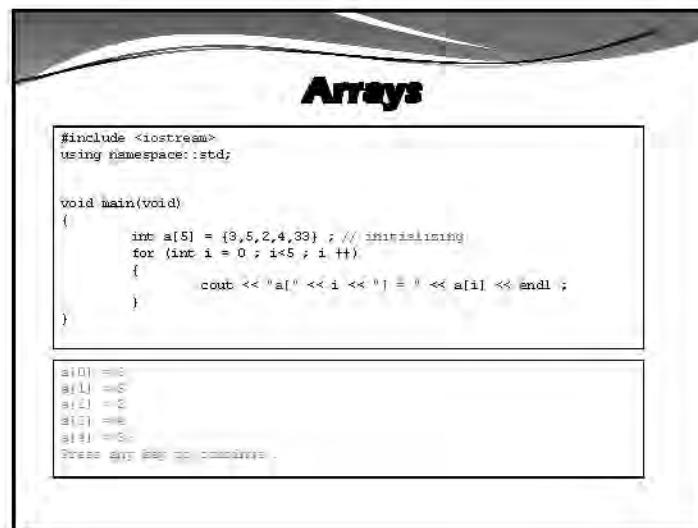
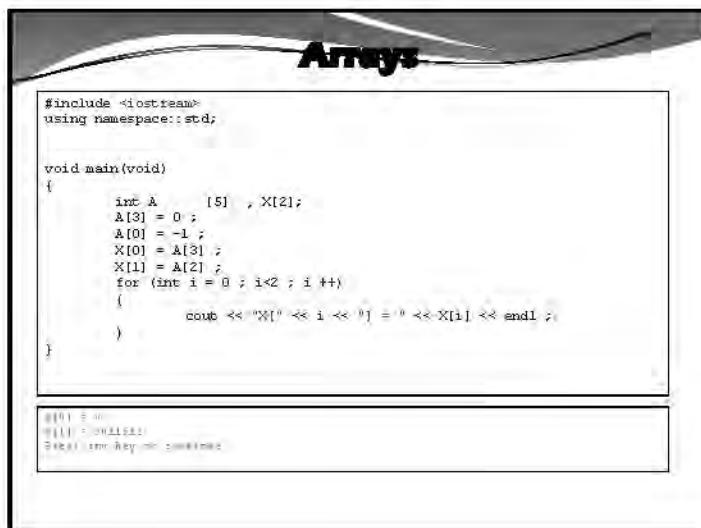
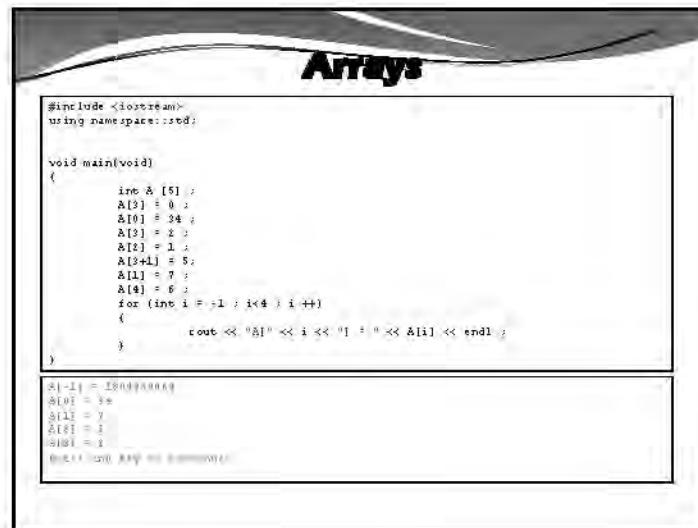
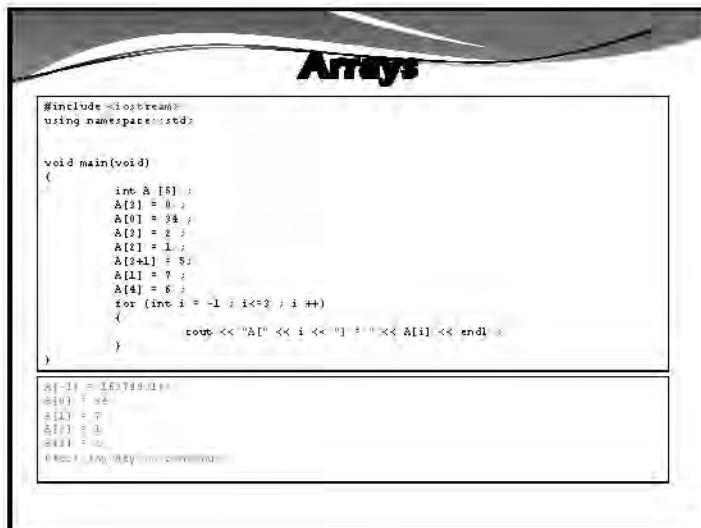
Arrays

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

void main(void)
{
    int & A [5];
    A[3] = 0;
    A[0] = 34;
    A[2] = 2;
    A[0] = 1;
    A[3+1] = 5;
    A[1] = 7;
    A[4] = 6;
    for (int i = -1 ; i<=2 ; i++)
    {
        cout << "A[" << i+1 << "] = " << A[i+1] << endl ;
    }
}

A[0] = 34
A[1] = 1
A[2] = 2
A[3] = 5
A[4] = 6
```

Please enter the correct code.



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 & run:

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 output:

```
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] = 32341223
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 << "The outputted array is : " << endl ;
    cout << "The outputted array is : " << endl ;
    for (int i=0 ; i<5 ; i++)
    {
        cout << a[i] << endl ;
    }
}
```

The computed array is:

```
4
5
6
7
8
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] ;
}
```

16

Arrays

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

void main(void)
{
    float a[4] = {30};
    cout << a[0] << endl;
}
```

10

Arrays

```
#include <iostream>
using namespace::std;
const int GoGo = 0 ;

void main(void)
{
    float a[GoGo] = {30};
    cout << a[0] << endl;
}
```

Compiler error - bad allocation at line 5 of file Gogo

Arrays

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

void main(void)
{
    float a[4] = {30. , 32 , 33.4 , 23.0 };
    cout << a[0] << endl;
}
```

30

Bad with warning of truncation from double to int

Arrays

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

void main(void)
{
    int x = 44. ;
}
```

Compiler error

*But with warning for converting from double to int ...
Possible loss of data*

→ Note! (You should know why)

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;
}
```

```
30
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;
}
```

Compilation error ... Type mismatch between integral types

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");
}
```

Please email to: info@tawtline.com

- 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
```

Stringwiring Atyp

```
#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;
    }
}
```

1.0e
1.01
1.02
1.03
1.44
0

1.0e
1.01
1.02
1.26
1.11
0
PRESERVE SPACES DO NOT REMOVE

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

void main(void)
{
 char str1 [] = "Hello";
 char str2 [] = {'H','e','l','l','o','\0'};
 (str1[0]==str2[0]) ? cout << "Tuppy ! " : cout << "Bo Bo :("
```

```
#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 << "Bo Bo :("
```

Yappy

## Strings using Arrays

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

void main(void)
{
 char str1 [] = "Hello";
 char str2 [] = {'H','e','l','l','o'};
 (str1==str2) ? cout << "Yuppy :)" : cout << "Bo Bo :(" ;
}
```

No No :)

## Strings using Arrays

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

void main(void)
{
 char str1 [] = "Hello";
 char str2 [] = {'H','e','l','l','o'};
 (str1==str2) ? cout << "Yuppy :)" : cout << "Bo Bo :(" ;
}
```

Bo Bo :()

Why is it so?

## Strings using Arrays

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

void main(void)
{
 char str1 [] = "Hello";
 char str2 [] = {'H','e','l','l','o'};
 (str1==str2) ? cout << "Yuppy :)" : cout << "Bo Bo :(" ;
}
```

Why is it so?

One we are comparing  
addresses and not values!

No No :)

## Strings using Arrays

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

void main(void)
{
 char str1 [] = "Hello";
 char str2 [] = {'H','e','l','l','o'};
 (str1==str2) ? cout << "Yuppy :)" : cout << "Bo Bo :(" ;
 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;
 }
}
```

|    |   |    |
|----|---|----|
| 6  | 3 | 0  |
| 10 | 4 | 10 |
| 10 | 1 | 10 |
| 20 | 0 | 20 |
| 10 | 8 | 10 |
| 11 | 1 | 11 |
| 0  |   | 0  |

# Strings using Arrays

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

void main(void)
{
 char str1 [] = "Hello";
 char str2 [] = {'H','e','l','l','o','\0'};
 (strcmp(str1) ? cout << "Furry!" : cout << "No No :\(");
 cout << endl;

 for (int i = 0 ; i < 6 ; i++)
 {
 cout << str1[i] ;
 }

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

 for (int i = 0 ; i < 6 ; i++)
 {
 cout << str2[i] ;
 }
 cout << endl;
}
```

# Strings using Arrays

# Strings using Arrays

# Strings using Arrays

## 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;
}

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 prints  
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 0043F958
Press any key to continue...
```

When printing the array it prints  
it's location on memory by  
default

Just the char array  
(char charArr [1])  
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 prints  
it's location on memory by  
default

Just the char array  
(char charArr [1])  
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!
You 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 0010111234
Press any key (F10) 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();
}
```

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;
 }
 cout << "Auto Arr after +2 to all elements : "
 << endl;
 for (int i=0 ; i<3 ; i++)
 {
 cout << "Arr1[" << i << "]=" <<
 (Arr1[i]+2) << endl;
 }
}

void main(void)
{
 AutoArr();
}

```

Press any key to continue .

### 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;
 }
 cout << "Auto Arr after +2 to all elements : "
 << endl;
 for (int i=0 ; i<3 ; i++)
 {
 cout << "Arr1[" << i << "]=" <<
 (Arr1[i]+2) << endl;
 }
}

void main(void)
{
 AutoArr();
}

```

Press any key to continue .

### 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;
 }
 cout << "Auto Arr after +2 to all elements : "
 << endl;
 for (int i=0 ; i<3 ; i++)
 {
 cout << "Arr1[" << i << "]=" <<
 (Arr1[i]+2) << endl;
 }
}

void main(void)
{
 AutoArr();
 AutoArr();
}

```

Press any key to continue .

### 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;
 }
 cout << "Auto Arr after +2 to all elements : "
 << endl;
 for (int i=0 ; i<3 ; i++)
 {
 cout << "Arr1[" << i << "]=" <<
 (Arr1[i]+2) << endl;
 }
}

void main(void)
{
 AutoArr();
}

```

Press any key to continue .

## Strings using Arrays – static

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

void StaticArr()
{
 static int Arr1 [3] = {1,2,3};
 cout << "Arr : " << endl;
 for (int i=0 ; i<3 ; i++)
 {
 cout << "Arr1[" << i << "]=" << Arr1[i] << endl;
 }
 cout << "Arr after +2 to all elements :" << endl;
 for (int i=0 ; i<3 ; i++)
 {
 cout << "Arr1[" << i << "]=" << (Arr1[i]+2) << endl;
 }
}

void main(void)
{
 StaticArr();
 StaticArr();
}
```

## Strings using Arrays – static

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

void StaticArr()
{
 static int Arr1 [3] = {1,2,3};
 cout << "Arr : " << endl;
 for (int i=0 ; i<3 ; i++)
 {
 cout << "Arr1[" << i << "]=" << Arr1[i] << endl;
 }
 cout << "Arr after +2 to all elements :" << endl;
 for (int i=0 ; i<3 ; i++)
 {
 cout << "Arr1[" << i << "]=" << (Arr1[i]+2) << endl;
 }
}

void main(void)
{
 StaticArr();
 StaticArr();
}
```

## 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);
}
```

## 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 were 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 PassingArrRef(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 , can't be referenced call by
// its 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);
}

// Passing arrays by value is passing it with reference
// not by value ...

```

## 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 pass arrays with values , can't be referenced call by
// its self

```

## Multiple subscripted Arrays

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

void main(void)
{
 int Arr[3][4] ; // 3 rows, 4 columns
}
```

|       | Column 0     | Column 1     | Column 2     | Column 3     |
|-------|--------------|--------------|--------------|--------------|
| Row 0 | at 0 1 1 0 1 | at 0 1 1 1 1 | at 0 1 1 2 1 | at 0 1 1 3 1 |
| Row 1 | at 1 1 1 0 1 | at 1 1 1 1 1 | at 1 1 1 2 1 | at 1 1 1 3 1 |
| Row 2 | at 2 1 1 0 1 | at 2 1 1 1 1 | at 2 1 1 2 1 | at 2 1 1 3 1 |

Diagram illustrating multiple subscripting:

- Row index (Row subscript)
- Column index (Column subscript)
- Element index (Element subscript)

## 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 subscripted size

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

void main(void)
{
 int Arr [] [3] ;
}
```

Compiler error - missing subscripted size

## Multiple subscripted Arrays

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

void main(void)
{
 int Arr[2][3] = { {2,3,5} , {4,1,3} } ;
}
```

|   |   |   |
|---|---|---|
| 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} } ;
}
```

|   |   |   |
|---|---|---|
| 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} , {4,0} } ;
 cout << Arr[2][3] << endl ;
}
```

|       |
|-------|
| 2 3 5 |
| 4 0   |

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 ;
}
```

2

## 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

## 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]);
}
```

2

## Multiple subscripted Arrays passing to function

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

void ArrFun(int A [1][3])
{
 cout << A [1][1] ;
}

void main(void)
{
 int Arr[2][3] = { {2,3,5}, {4,3} } ;
 ArrFun(Arr);
}
```

2

No compiler error for missing the first subscript in the function's prototype ... like single subscripted arrays

### Multiple subscripted Arrays passing to function

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

void ArrFun(int A [][1])
{
 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] ;
}
```

The diagram illustrates the memory layout of a 2x3 matrix. It shows two horizontal arrays of boxes representing rows. The first row, labeled 'Row #0', has three boxes. The second row, labeled 'Row #1', has two boxes. Above the first row is a pointer labeled '5000' pointing to its first element. Above the second row is a pointer labeled '5005' pointing to its first element.

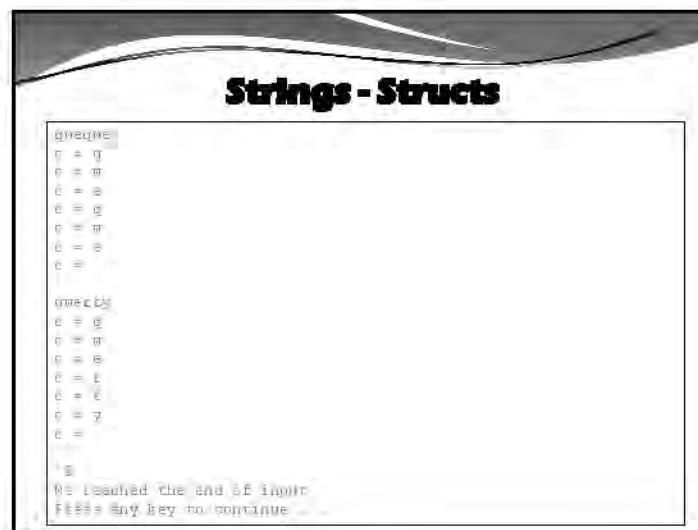
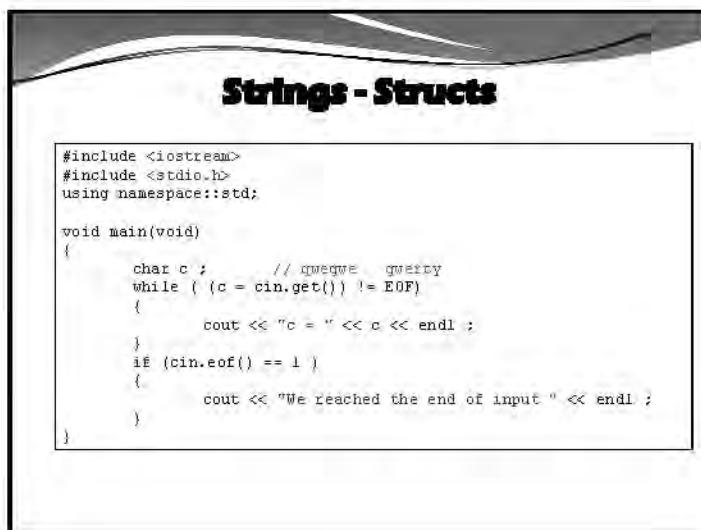
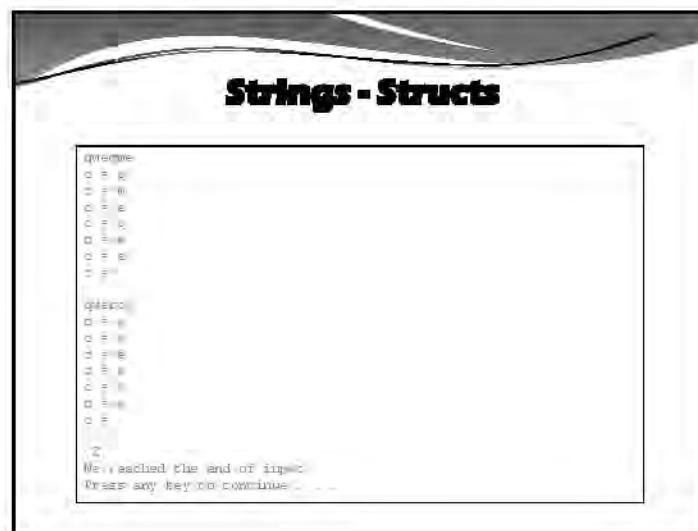
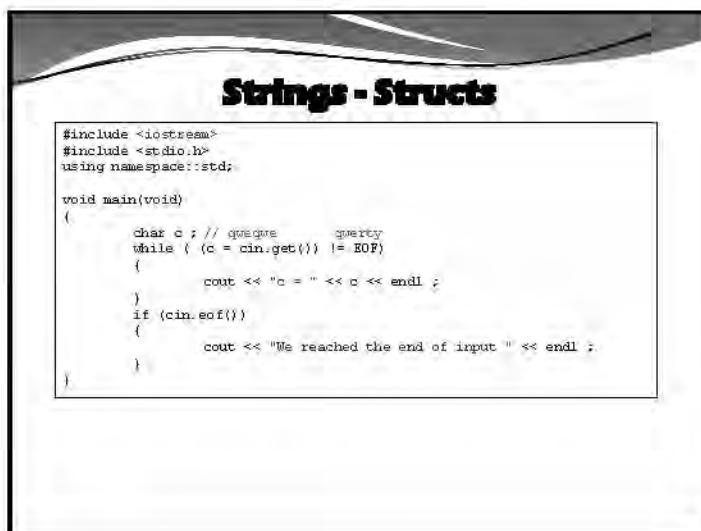
### Strings - Structs

### 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 stream "even white space"
- Returns **EOF** if end-of-file encountered
- **EOF**
  - `ctrl+z` IBM
  - `ctrl+d` Unix-Mac
- `cin.eof`
  - Returns 1 (True) if EOF occurred



## Strings - Structs

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

void main(void)
{
 char c; // queue query
 while ((c = cin.get()) != EOF)
 {
 cout << "c = " << c << endl;
 }
 if (cin.eof() == 1)
 {
 cout << "We reached the end of input." << endl;
 }
}
```

Compile and run

## 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

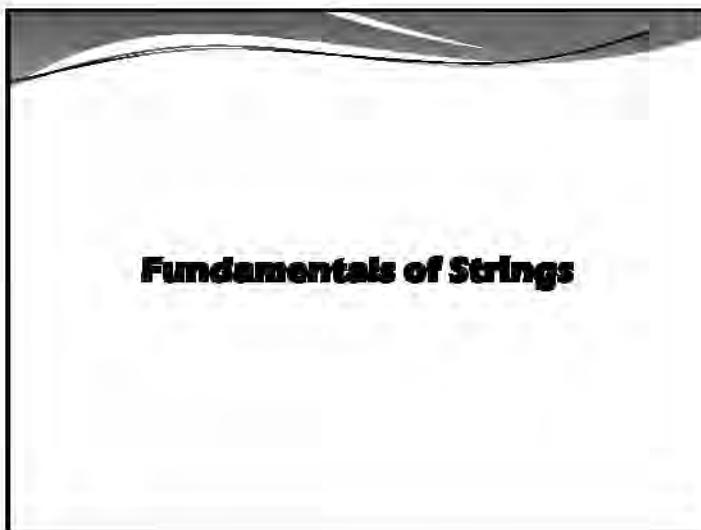
- Character handling library
  - <cctype>
    - Functions to perform tests and manipulations on characters
- As long as `char` doesn't allow negative values
  - EOF usually has value -1

## Strings - Structs

|                       |                                                                                                  |
|-----------------------|--------------------------------------------------------------------------------------------------|
| <code>isalnum</code>  | Check if character is alphanumeric ( <code>isalpha    isdigit</code> )                           |
| <code>isalpha</code>  | Check if character is alphabetic ( <code>isupper    islower</code> )                             |
| <code>isctrl</code>   | Check if character is a control character ( <code>iswcntrl</code> )                              |
| <code>isdigit</code>  | Check if character is a decimal digit ( <code>isdecimal</code> )                                 |
| <code>isgraph</code>  | Check if character has graphical representation using locale ( <code>isalpha    isdigit</code> ) |
| <code>islower</code>  | Check if character is lowercase letter ( <code>isalpha</code> )                                  |
| <code>isprint</code>  | Check if character is printable ( <code>isalpha    isdigit</code> )                              |
| <code>ispunct</code>  | Check if character is a punctuation character ( <code>isalpha    isdigit</code> )                |
| <code>isspace</code>  | Check if character is a white-space character ( <code>isalpha    isdigit</code> )                |
| <code>isupper</code>  | Check if character is uppercase letter ( <code>isalpha</code> )                                  |
| <code>isxdigit</code> | Check if character is hexadecimal digit ( <code>isalpha    isdigit</code> )                      |

And secondly, two functions to convert between letter cases:

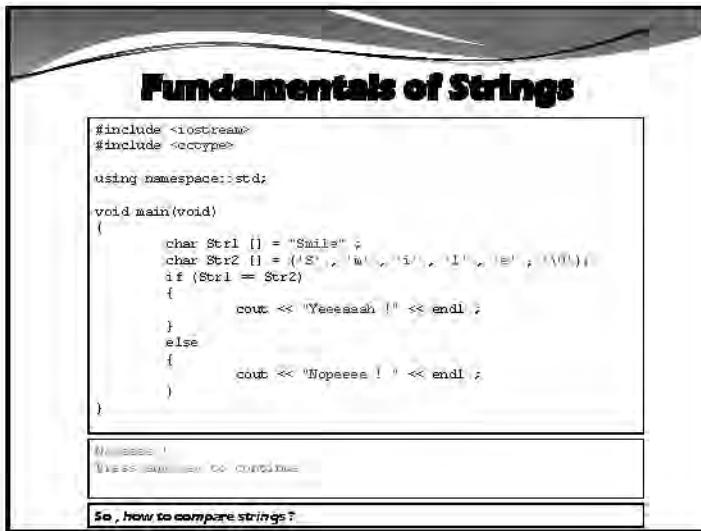
|                      |                                                                |
|----------------------|----------------------------------------------------------------|
| <code>tolower</code> | Convert uppercase letter to lowercase ( <code>isupper</code> ) |
| <code>toupper</code> | Convert lowercase letter to uppercase ( <code>islower</code> ) |



## 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".



```
#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 << "Yessssah !" << endl;
 }
 else
 {
 cout << "Nopeeee !" << endl;
 }
}
```

Output :  
Yessssah !

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;
}
```

## Fundamentals of Strings

```
#include <iostream>
#include <cctype>

using namespace::std;

void main(void)
{
 bool match = true ;
 char Str1 [1 = "Smile" ;
 char Str2 [1 = {'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 [1 = "Smile" ;
 char Str2 [1 = {'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 << endl;
}
```

No.

## Fundamentals of Strings

```
#include <iostream>
#include <cctype>

using namespace::std;

void main(void)
{
 bool match = true ;
 int i = 0 ;
 char Str1 [1 = "Smile" ;
 char Str2 [1 = {'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 << endl;
}
```

No.

## Fundamentals of Strings

```
#include <iostream>
#include <cctype>

using namespace::std;

void main(void)
{
 bool match = true ;
 int i = 0 ;
 char Str1 [1 = "Smile" ;
 char Str2 [1 = {'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 << 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 - any

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[1][5] = {"SoSo", "hee", "GoGo"};
```

Compile and run

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

void main(void)
{
 char TwoDimStr[1][1] = {"SoSo", "hee", "GoGo"};
```

Compile and run

## 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;
 }
}
```

SoSoSoSoSo  
heeheehee  
GoGoGoGoGo

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;
 }
}

Source: 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;
 }
}

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", "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;
 }
}
```

## 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;
 }
}
```

Source: Inserter  
Size: has  
Source: Inserter  
SoS  
bee  
IW

## 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] << endl;
 }
 cout << endl;
}
```

Reference: Threaded C++ 2020  
Date:  
Author: Inserter  
Source: "Same"

## Fundamentals of Strings

- Input Stream ...
  - get (CharArray, Size, Delimiter)
  - *1<sup>st</sup> parameter:*
    - Pointer to a constant array
  - *2<sup>nd</sup> parameter:*
    - Maximum number of characters to read
      - Reads until Size-1 or delimiter encountered (default EOF)
  - *3<sup>rd</sup> 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
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
he wanna 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 wanna go
str2 = he
Enter String
she wanna 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];
 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 wanna go
str2 = he
Enter String
she wanna 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 = he
Please 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
str2 = he
Please 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
str1 =
Please 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
Please 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 ;
}

Enter String
i Wanna go
str1 = i wanna go
Enter String
he wants to go
str2 = go
Please 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');// weve
 cout << "str1 = " << str1 << endl ;
}

Enter String
30
num
str1 =
Press any key to continue . . .

```

**Solutions:**  
Putting a "cin.get(); between string inputs

## 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 bad
 cin.get(str1 , Length , '\n') ;

 cout << "str1 = " << str1 << endl ;
}
```

Enter String  
10 bad  
str1 = 10ad  
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 bad
 cin.get(str1 , Length , '\n') ;

 cout << "str1 = " << str1 << endl ;
}
```

Enter String  
10bad  
str1 = 10bad  
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(c) ;

 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  
m  
str1 = i am happy  
c = m  
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 wanna go and play :D  
Enter a char  
saw = i wanna c  
c =  
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  
saw = i wanna :  
c =  
Press any key to continue...

*So it's impossible to guess how many cin.get() are needed to consume all potential 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();

 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
saw = i wanna :
c =
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
saw = i wanna :
c =
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)
    - 1<sup>st</sup> parameter:
      - Maximum number of characters to ignore
    - 2<sup>nd</sup> 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) ; // means go and see 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 see now !
Entered 5 char
.
.
.
str1 = i wanna go and see
c = .
Please enter 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 see !
 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 see !
Entered 5 char
.
.
.
str1 = i wanna go and see
c = .
Please enter 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;

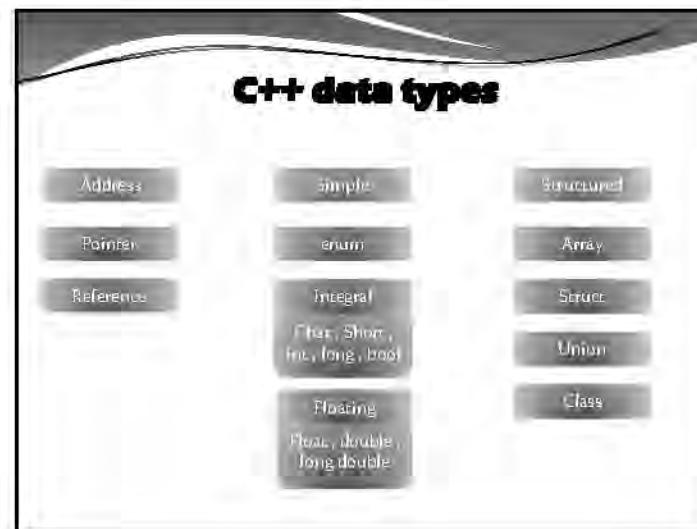
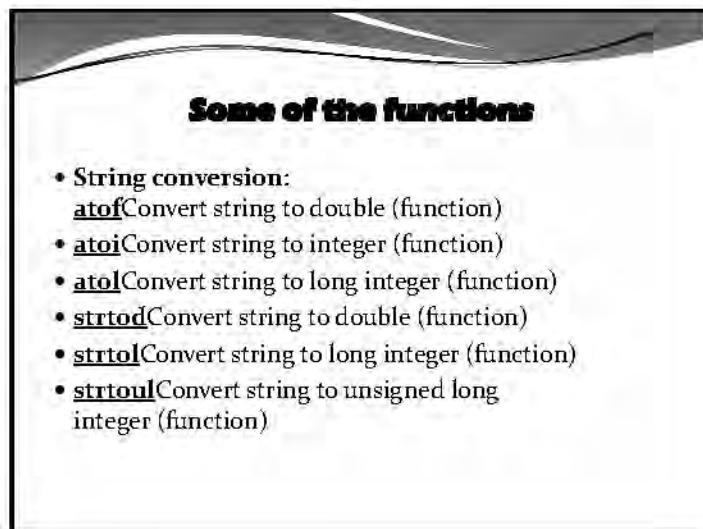
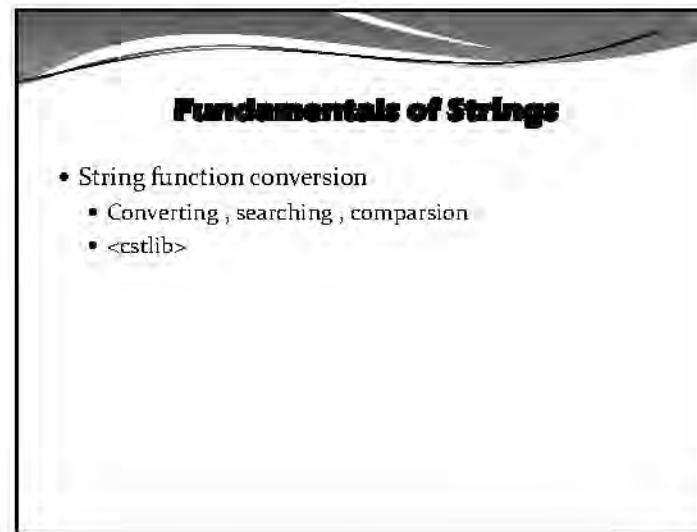
void main(void)
{
 int I Wanna Eat ;
 char str1[Length] ;

 cout << "Enter String : " << endl ;
 cin.getline(str1 , 80 , '\n') ;
 cout << "str1 = " << str1 << endl ;

 cout << "Enter a num : " << endl ;
 cin >> I Wanna Eat ;

 cout << "str1 = " << str1 << endl ;
 cout << "IWannaEat = " << IWannaEat << endl ;
}

Enter String :
I am very
str1 = I am happy
Enter a num :
12
str1 = I am happy
IWannaEat = 12
Press any key to continue
```



## 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];
 }
}

Comment: Notice that there is no semicolon at the end of the struct definition.
```

## 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 ;
}

A
B
C
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 ;
}

A
B
C
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
Please type 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
Please type 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 ;
}

D:\Users\user\Documents\MyFirstS.cpp: In function 'int main()':
D:\Users\user\Documents\MyFirstS.cpp:12:1: error: missing semicolon before 'cout' [-fpermissive]
 cout << TempStruct[i].a << endl ;
```

## 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 ;
}

0
0
0
Please type 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 linking the name of the STRUCT we should declare you must redeclare
```

## 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 ;
}

0
0
8
```

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 ;
}

1
2
8
```

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
8
```

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 ;
}

Output: 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 ;
}

Output: Error
```

## Structs

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

void main(void)
{
 struct student
 {
 char Name[20] ;
 int Num ;
 int id ;
 };
 student st1 = ("Mohammad" , 196 , 7);
}

Output: Error
```

## 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);
}

Output: Error
```

## 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;
 }
}
```

## 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.Name == Tempst1.Name)
 {
 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.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 = {"Mohammed", 196, 7};
 student st2 = {"ZeeZee", 111, 4};
 student Tempst1 = {"Mohammed", 196, 7};

 if (st1.Num == st2.Num)
 {
 cout << "Yeah !" << endl;
 }
 else
 {
 cout << "No !" << endl;
 }
 cout << st1.Num << endl;
 cout << st2.Num << endl;
}
```