

# Introduction to C and C++

HIGH THOUGHTS MUST HAVE  
HIGH LANGUAGE

*-Aristophanes*

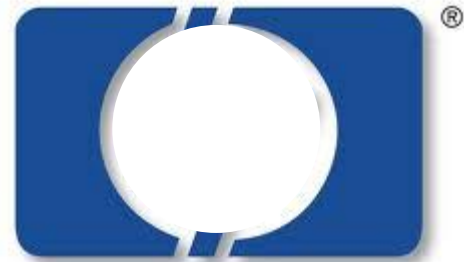


# History of C

- The initial development occurred at AT&T Bell Labs → between 1969 and 1973
- First implemented in 1972
- By Dennis Ritchie, Bell laboratories, USA
- to design the UNIX operating system
- Modified from B (developed by Ken Thompson in 1970)
- B modified from BPCL (Basic Combined Programming Language by Martin Richards of the University of Cambridge in 1966).
- ANSI C, (1983): American National Standards Institute
- ISO in 1990



*Your potential. Our passion.™*



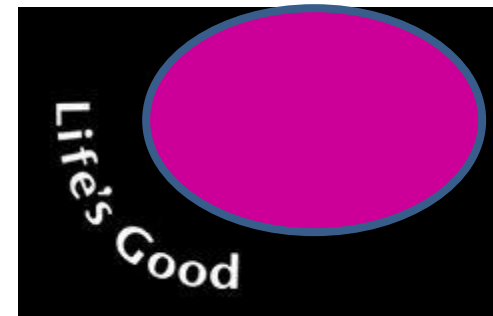
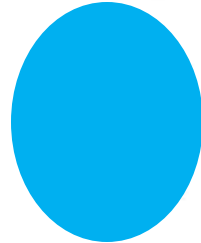
i n v e n t

Connecting People

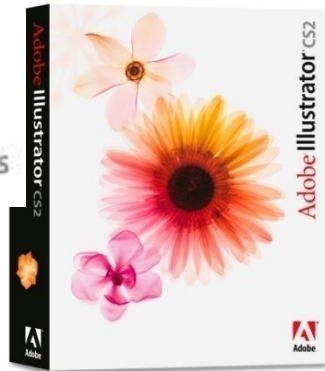
imagination at work



Let's make things better



# Applications of C & C++



Google Chrome



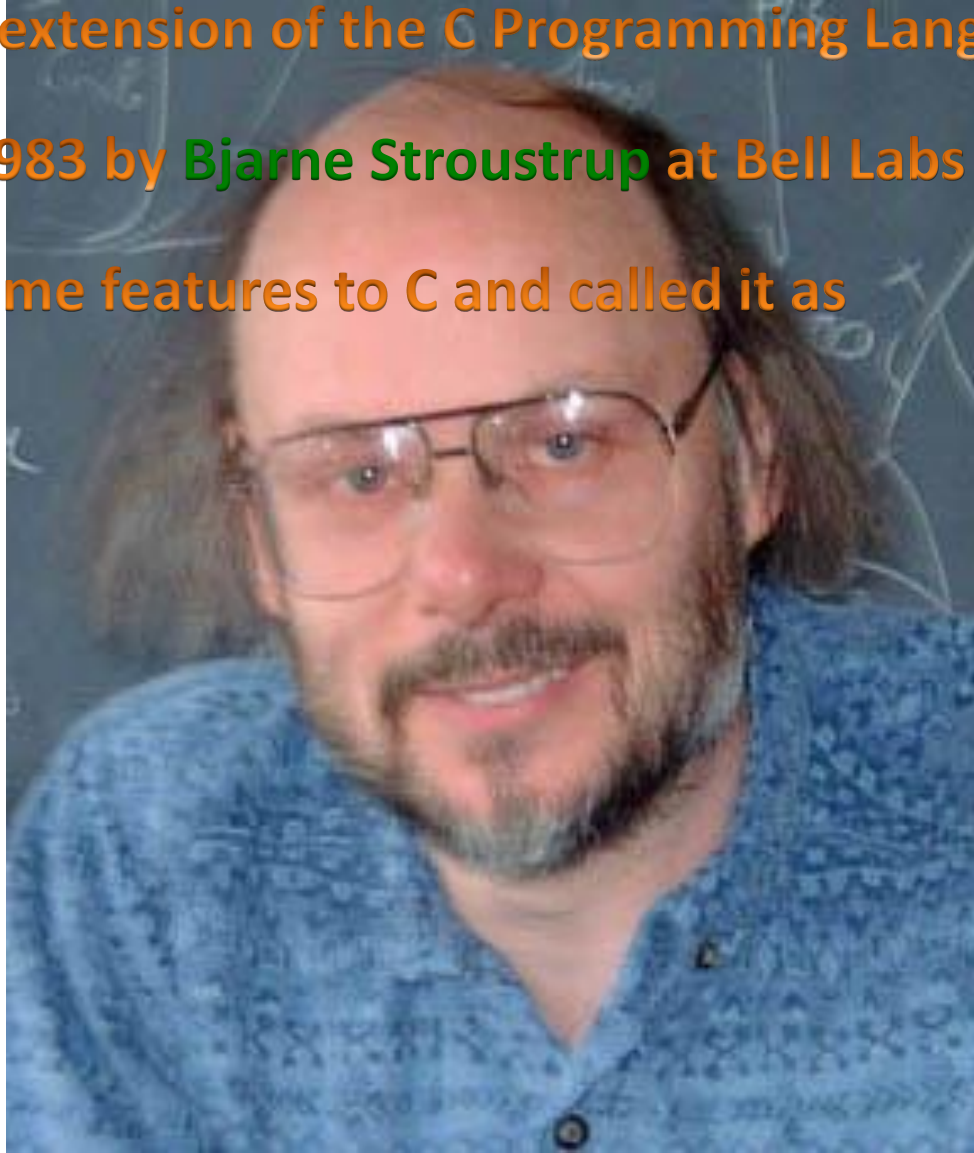
# Microsoft®

*Your potential. Our passion.™*



# History of C++

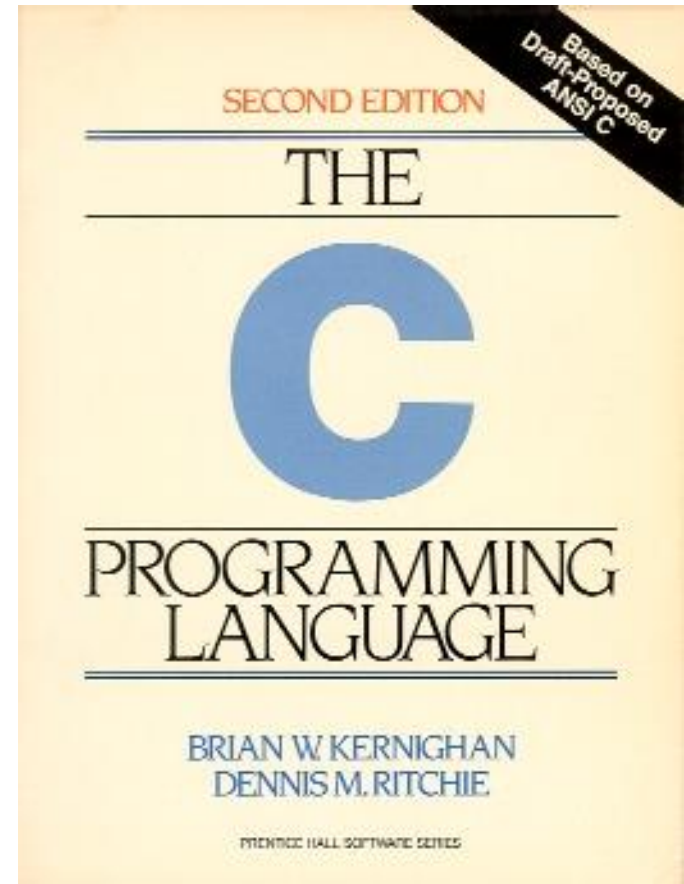
- Basically an extension of the C Programming Language
- Created in 1983 by Bjarne Stroustrup at Bell Labs
- He added some features to C and called it as "C with Classes".





# Getting started

- C / C++ program consists of
  - Character set
  - Identifiers and keywords
  - Constants and Variables
  - Data types
  - Operators and Expressions
  - Statements



# Character set

## Alphabets

Upper case	A B C ..... X Y Z	26
------------	-------------------	----

Lower case	a b c ..... x y z	26
------------	-------------------	----

## Digits

0 1 2 3 ..... 9	10
-----------------	----

## Special symbols

~ ` " ! @ # \$ % ^ & * ( ) _ + - = { } [ ] < > / \   , . : ; ?	31
--	----

C is case sensitive

**Total → 93 characters**

# Identifiers

- **Names given to various program elements like**
  - **Variables**
  - **Constants**
  - **Functions**
  - **Arrays**
- **Should begin with an alphabet**
- **Can consist of alphabets, digits & underscore**
- **Maximum 32 characters**
- **Meaningful – should represent the purpose for which it is used**



# Identifiers

## Identifier for a variable to store acceleration due to gravity

accng	Possible
Accn_g	Possible
Accng	Possible
Accn-g	Not possible
981accng	Not possible
Accng_981	possible

# Keywords

- Identifiers reserved for specific usages in C / C++
- Have standard predefined meanings
- Can be used only for the intended purpose

# Keywords

Keywords			
auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

# Constants

```
graph TD; A[Constants] --> B[Numeric constants]; A --> C[Character constants]; B --> D[Integer constant]; B --> E[Real constant]; C --> F[Single character constant]; C --> G[String constant];
```

Numeric  
constants

Character  
constants

Integer  
constant

Real  
constant

Single  
character  
constant

String  
constant

# CONSTANTS

Integer constant	Real constant	Single character constant	String constant
123	12.3	'1'	"123"
12	12.	'a'	"a"
1	1.0	'A'	"A"
1345	13.45	'%'	"%"
		Takes corresponding ASCII value. Hence used in expressions for calculations	No meaning. A group of single character constants

# Variables

- Identifier used to represent a single data item
- Variable names correspond to locations in the computer's memory
- Every variable has a name, a type, a size and a value
- Once data is assigned to a variable, the same data can be accessed later by referring to the variable name
- Eg. Sum, avg, total, .....



# Variables

- `int sum;`



`sum = 10`

0	0	0	0	1	0	1	0
$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$

$$2^3 * 1 + 2^2 * 0 + 2^1 * 1 + 2^0 * 0 = 10$$

# Data types

- **Character - char**
  - Holds single character constants
  - 8 bit (1 byte) memory
  - -128 to 127
- **Integer - int**
  - Holds integer quantities
  - 16 bit (2 bytes) memory
  - -32768 to 32767

# Data types

- **Floating point – float**

- Holds real quantities with a fractional component
- 8 digit precision
- 32 bit (4 bytes) memory
- $3.4 \times 10^{-38}$  to  $3.4 \times 10^{+38}$

- **Double precision floating point – double**

- Holds real quantities with a fractional component
- 64 bit (8 bytes) memory
- $1.8 \times 10^{-308}$  to  $1.8 \times 10^{+308}$

# Declaration of variables

- Variables should be **declared** prior to their use
  - Indicates which type of data is to be contained by the variable
- **Syntax**
  - Data type v1, v2, v3;
- **Examples**
  - char name, letter;
  - int sum, remainder;
  - float avg, area, volume;
  - double root, value;

# Operators

- Symbols which tell the computer to perform specific mathematical operations
- Operands
- Unary and binary operators
- Arithmetic, Relative, Logical
- Equality, Assignment

# Arithmetic Operators

Operator	Action
-	Subtraction Unary minus
+	Addition Unary plus
*	Multiplication
/	Division
%	Modulus division
--	Decrement
++	Increment



# Hierarchy of Arithmetic Operators

- ++, --, -, + (unary)
- \*, /, %
- +, - (binary)
- Associativity → left to right

# Increment / decrement operators

- **Action**

- ++ → adds one to the operand

- -- → subtracts one from the operand

- **Two types**

- Pre increment (++a) and post increment (a++)

- Pre decrement (--a) and post decrement (a--)

# Increment / decrement operators

- Examples

**A = 10**  
**B = ++A**  
**A = 10**  
**A = A+1 = 11**  
**B = 11**

**A = 10**  
**B = A++**  
**A = 10**  
**B = 10**  
**A = A+1 = 11**

**A = 10**  
**B = --A**  
**A = 10**  
**A = A-1 = 9**  
**B = 9**

**A = 10**  
**B = A--**  
**A = 10**  
**B = 10**  
**A = A-1 = 9**

# Increment / decrement operators

- Practice problem

- ✓ `P = 1;`

- ✓ `Q = P++;`

- ✓ `Q = P--;`

- ✓ `Q = --P;`

- ✓ `Q = ++P;`

- Obtain the values stored in P and Q after the execution of this program segment.

# Increment / decrement operators

Answer

P = 1;

Q = P++;

Q = P--;

Q = --P;

Q = ++P;

P	Q
1	-
2	1
1	2
0	0
1	1

# Increment / decrement operators

- Practice problem 2

- ✓ `P = 1;`

- ✓ `Q = --P;`

- ✓ `Q = ++P;`

- ✓ `Q = P++;`

- ✓ `Q = P--;`

- Obtain the values stored in P and Q after the execution of this program segment.



# Relational Operators

Operator	Action
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equal to
!=	Not equal to

} EQUALITY OPERATORS

# Relational Operators

- **Practice session**

- `int a =10, b =13;`
- `float f = 22.5;`
- `char p = 'a'; //assigns an ascii eqInt to p, p=97,`

Expression	Interpretation	Value
<code>a &gt; b</code>	False	0
<code>b &lt; f</code>	True	1
<code>b &gt;= f</code>	False	0
<code>p &lt;= f</code>	False	0
<code>(a + b) == f</code>	False	0
<code>(f - 'b') != 'c'</code>	True	1

# Logical Operators

Operator	Action
&&	AND
	OR
!	NOT

# Truth Table for Logical Operators

A	B	A && B	A    B	!A	!B
1	1	1	1	0	0
1	0	0	1	0	1
0	1	0	1	1	0
0	0	0	0	1	1

# Hierarchy of relational and logical operators

- !
- >, <, >=, <=
- ==, !=
- &&
- ||
- Associativity → left to right

# Assignment Operators

Operator	Operation	Action
<b>=</b>	$A = B$	$A = B$
<b>+=</b>	$A += B$	$A = A + B$
<b>-=</b>	$A -= B$	$A = A - B$
<b>*=</b>	$A *= B$	$A = A * B$
<b>/=</b>	$A /= B$	$A = A / B$
<b>%=</b>	$A \% = B$	$A = A \% B$

**Associativity → right to left**

**Hierarchy → last**



# Expressions & Statements

$$F = m a$$

$$F = m * a$$

$$V = a^3$$

$$V = a * a * a$$

$$area = \pi r^2$$

$$area = pi * r * r$$

# Expressions & Statements

$$Torque = \frac{2 m_1 m_2}{m_1 + m_2} g$$

$$\text{torque} = 2 * m1 * m2 * g / (m1 + m2) \quad m_1, m_2 : \text{not possible !!!}$$

$$PE = \frac{1}{2} k (x_2^2 - x_1^2)$$

$$PE = k * (x2 * x2 - x1 * x1) / 2$$

$$s = \frac{v^2}{2 \mu g}$$

$$s = v * v / (2 * \mu * g)$$

$\mu, \theta$ : not possible !!!

# Structure of a C program

- Program is written as a set of functions.
- Main function → most important
- Program execution begins with main

# General format

Preprocessor directives

Function declarations

Global variable declaration

`void main()`

`{`

Local variable declaration;

Statements and function calls;

`}`

Other function definitions

# General format for simple programs

**Preprocessor directives**

**main()**

**{**

**Local variable declaration**

**Statements**

**Input statement**

**Execution statement**

**Output statement**

**}**

# Precompiler directives

- Examples

- `#include<iostream.h>`

- Includes the header file `iostream.h` to the code

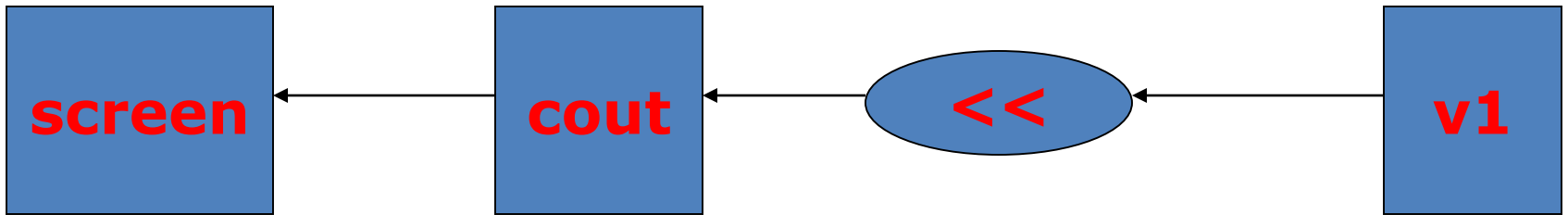
- `#define PI 3.14285714`

- Assigns the value `3.14285714` to the variable `PI`

# Output statement

- **Syntax**
  - `cout<<v1;`
  - `<<`
    - Insertion or put to operator
    - Inserts the contents of the variable on its right to the cout object
  - `cout`
    - Object in `iostream.h`
    - Sends the output to the standard output device

# Output operation





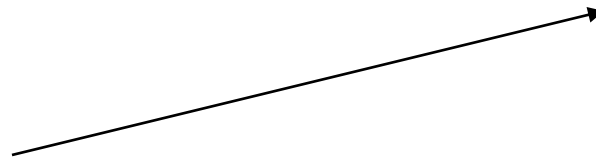
# Output statement

- **Example 1**

- `A = 12;`

- `cout<<A;`

- **Output**



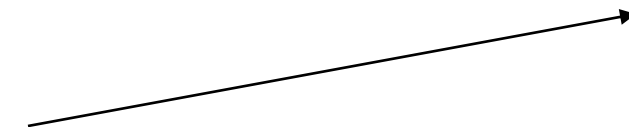
**12**

- **Example 2**

- `A = 12;`

- `cout<< "A = " <<A;`

- **Output**



**A=12**

# Output statement

- **Example 3**

- A = 12; B = 13;

- cout<<A;

- cout<<B;

- Output



**1213**

- **Example 4**

- A = 12; B = 13;

- cout<< "A = " <<A;

- cout<< "B = " <<B;

- Output



**A =12B =13**

# Escape sequences

## (Backslash character constants)


- Represent non printing characters
- Used in output statements

Escape sequence	Character
<code>\n</code>	New line
<code>\r</code>	Carriage return
<code>\a</code>	Alert
<code>\0</code>	Null
<code>\b</code>	Backspace
<code>\t</code>	Horizontal tab
<code>\v</code>	Vertical tab

# Output statement

- **Example 5**


- `A = 12; B = 13;`
- `cout<<A<<"\n";`
- `cout<<B;`
- **Output**



**12**  
**13**

- **Example 6**

- `A = 12; B = 13;`
- `cout<< "A = " <<A<<"\n";`
- `cout<< "B = " <<B;`
- **Output**

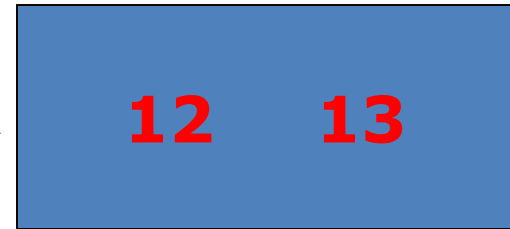


**A =12**  
**B =13**

# Output statement

- **Example 7**

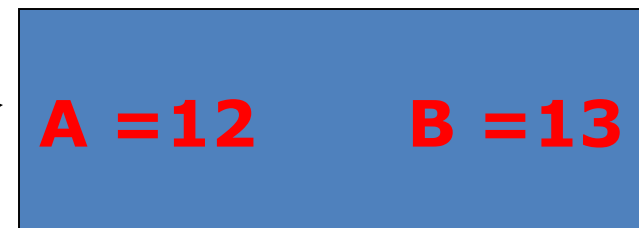
- `A = 12; B = 13;`
- `cout<<A<<"\t";`
- `cout<<B;`
- **Output**



**12      13**

- **Example 8**

- `A = 12; B = 13;`
- `cout<< "A = " <<A<<"\t";`
- `cout<< "B = " <<B;`
- **Output**



**A = 12      B = 13**

# Output statement

- **Example 7**

- `A = 12; B = 13;`
- `cout<<A<<"\t"<<B;`

Output



**12      13**

- **Example 8**

- `A = 12; B = 13;`
- `cout<< "A = " <<A<<"\t"<< "B = " <<B;`

– Cascading

– Output

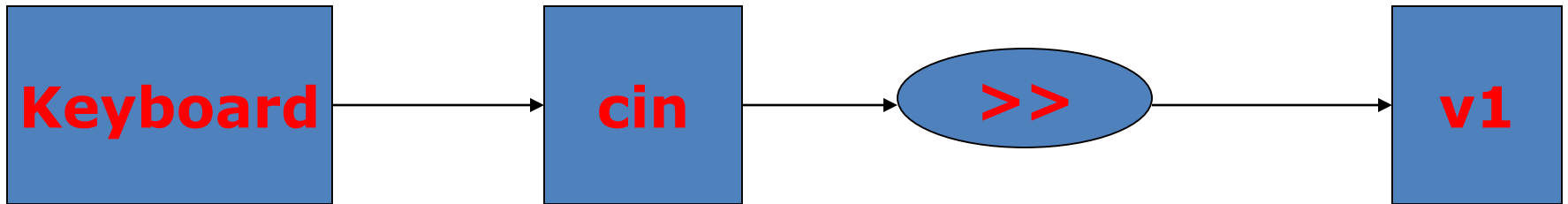


**A = 12      B = 13**

# Input statement

- **Syntax**
  - **cin>>v1;**
  - **cin**
    - **Object in iostream.h**
    - **Takes the value from a standard input device**
  - **>>**
    - **Extraction or get from operator**
    - **Gets the value from cin and assigns to the variable on its right**

# Input operation





# Input statement

- **Example 1**

- `cin>>A;`

- **Example 2**

- `cin>>A;`

- `cin>>B;`

- `cin>>C;`

- **Example 3**

- `cin>>A>>B>>C;`

- Cascading

# Comments

**// Single line**

**/\* This is the way for commenting Multiple lines\*/**