

Language Constructs

Chapter 2



What is Covered?

- Hello World Program
- C++ language constructs
 - Identifiers,
 - Literals
 - Keywords, etc.
- Conditional Program Statements
- Creating Loops



Introduction

- C++ supports full C syntax
- This chapter uses procedure-oriented programming
- Chapter 4 onwards use object-oriented programming



Hello World Program

```
#include "stdafx.h"
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello World\n";
    return 0;
}
```

- Compiling
 - C:>cl Hello.cpp
- Running
 - D:\PHI\C++book>Hello
- Output
 - Hello World



C++ Program Structure

- ***Function Prototype***

```
return-type functionName (argument-list)
{
    // program statements;
}
```

- The `functionName` is unique within the entire application
- The `argument-list` specifies function arguments or parameters
 - Each argument consists of argument name and its data type
- The `return-type` specifies the data type for the return value
- The `void` return-type indicates that function does not return any value



Program Structure

```
// header files
// function prototypes
void function1();
void function2();
void function3();
int main (int argc, char *argv[])
{
    function1();
    function2();
    ...
    return 0;
}
```



Program Structure - Continued

```
void function1 ()
{
    function3();
    ...
}
void function2()
{
    // some implementation
}
void function3()
{
    // some implementation
}
```



Comments

- Single line comments
 - Starts with `//`, ends with `newline` character
- Multi-line comments
 - Starts with `/*`, ends with `*/`



Accepting User Input

```
#include "stdafx.h"
#include <iostream>
using namespace std;
int main()
{
    char    name[15];
    cout << "Enter your name: ";
    cin >> name;
    cout << "Hello " << name << "!\n";
    return 0;
}
```



Identifiers

- Identifiers are used for creating variables, functions, classes and so on



Rules for Creating Identifiers

- Can contain
 - Characters,
 - Digits and
 - Underscore
- Cannot start with a digit
- Can be of any length
- Few words are reserved as keywords.
 - You cannot declare identifiers having name same as a keyword



Examples of Valid Identifiers

<code>intVariable</code>	an identifier containing combination of upper and lower case characters
<code>_myVariable</code>	an identifier can begin with an underscore
<code>_YourVariable</code>	an identifier can contain underscore; Note the second underscore in the example
<code>B4U</code>	an identifier can contain a digit
<code>DoubleVariable</code> , <code>doubleVariable</code>	the two identifiers shown here are distinct, as C++ is case sensitive



Examples of Invalid Identifiers

4uonly	identifier cannot start with a digit
main	main is a reserved keyword in C++



Literals

- Are referred to as *constants*
- Can be
 - Characters
 - Strings
 - Integer Numbers
 - Floating Point Numbers



String and Character Constants

- **String Constants**
 - Specified in double quotes
 - Ex: “True”, “False”, “Hello
- **Character Constants**
 - Specified in single quotes
 - Ex: ‘T’, ‘F’



Escape Sequences

Escape Code	Description
<code>\n</code>	Newline consists of Carriage Return (CR) and Line Feed (LF)
<code>\r</code>	Carriage return
<code>\t</code>	Tab
<code>\v</code>	Vertical tabulation
<code>\b</code>	Backspace
<code>\f</code>	Page feed
<code>\a</code>	Alert (beep)
<code>\'</code>	Single quote (')
<code>\"</code>	Double quote (")
<code>\?</code>	Question mark (?)
<code>\\</code>	Inverted slash (\)



Integer Constants

- Represent integers that cannot be modified by program code
 - Ex: 100, -345, 34
- Constant Expression
 - Ex: 36+64
- Octal Constants prefixed with 0
 - Ex: 0232, 0125
- Hex Constants prefixed with 0x or 0X
 - Ex: 0xABC, 0x100



Floating Point Constants

- Floating Point Constants:
 - 3.14159
 - 40.0
 - 6.02e23 // expressed in scientific notation



Keywords

asm	auto	break	case	catch
char	class	const	continue	default
delete	do	double	else	enum
extern	float	for	friend	goto
if	inline	int	long	new
operator	private	protected	public	register
return	short	signed	sizeof	static
struct	switch	template	this	throw
try	typedef	union	unsigned	virtual
void	volatile	wchar_t	while	



Invalid Use of Keywords

- Following statements generate compilation error
 - `int virtual;` // virtual is a reserved keyword
 - `char private;` // private is a reserved keyword



Data Types

Data type	Bytes	Description	Range- Signed	Range- Unsigned
char	1	Represents ASCII character set.	-128 to 127	0 to 255
short int or short	2	Is larger than or equal to the size of char type	-32768 to 32767	0 to 65535
int	2	The range for this data type is System dependent.	-32,768 to 32,768	0 to 65, 535
bool	1	Can have one of the two values (True or False).		true or false
long int	4	Is is larger than or equal to the size of int data type.	2147483648 to 2147483647	- 0 to 4294967295
float	4	Is the smallest floating type		3.4E- 38 to 3.4E+38



Data Types

Data type	Bytes	Description	Range- Signed	Range- Unsigned
double	8	Has size larger than or equal to the type float but smaller than or equal to the long double type.		1.7E- 308 to 1.7E+308
long double	8	Is a floating type		3.4E-4932 to 1.1E+4932
wchar_t	2	Represents a wide character or multi-byte character type.		0 to 65,535
enum	16	Enum is a user-defined type consisting of a set of named constants called enumerators		-32,768 to 32,767



Operators in C++

- *Increment and Decrement Operators*
- *Mathematical / Arithmetic Operators*
- *Relational Operators*
- *Logical Operators*
- *Bitwise and Bitshift Operators*
- *Unary Operators*
- *Conditional or Ternary Operator*
- *Comma Operator*
- *Casting Operator*
- *sizeof operator*



Mathematical / Arithmetic Operators

+	Addition
-	Subtraction
/	Division
*	Multiplication
%	Modulus



Relational Operators

<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equivalent
!=	Not equivalent



Logical Operators

&&	AND
	OR
!	NOT



Bitwise and Bitshift Operators

&	AND
	OR
^	XOR
~	NOT
<<	Left-shift
>>	Right-shift



Unary Operators

-	Unary minus
+	Unary plus
++	Increment
--	Decrement
&	address-of – provides the computers internal location of the variable.
*	de-reference – returns the value of the variable located at the address that follows it.
new	Memory allocation operator
delete	Memory release operator



Conditional or Ternary Operator

?:

Syntax: Condition ? expression1 : expression2
(Ex: $s = (x < 0) ? -1 : x * x$)



Comma Operator

,

e.g: `A = (B++, C++, D++, E++);`



Casting Operator

()	A = (int) B;
-----	--------------



sizeof Operator

sizeof()

e.g: `cout << sizeof(double);`



Increment and Decrement Operators

++	Auto-increment
--	Auto-decrement



Program Statements

- May be
 - Variable declaration statement
 - Output statement
 - Complex statement such as evaluating conditions, defining program loops and so on



Variable Declaration Statements

- Syntax

```
datatype variable1, variable2;
```

Ex:

```
int IntegerVar1 = 10, IntegerVar2 = 200;
```



Conditional Statements

- *The **if** statement*
- *The **if-else** statement*
- *The **if-elseif-else** statement*



The if statement

Syntax:

```
if (condition)
    statement1;
    statement2;
```



The if-else statement

Syntax:

```
if (condition)
    statement1;
else
    statement2;
```



The if-elseif-else statement

Syntax:

```
if (condition1)
    statement1;
elseif (condition2)
    statement2;
else
    statement3;
```



Another Example

Syntax:

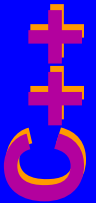
```
if (condition1)
    statement1;
elseif (condition2)
    statement2;
elseif (condition3)
    statement3;
elseif (condition4)
    statement4;
...
else
    statement5;
```



The switch statement

Syntax:

```
switch (expression)
{
    case 'value1':
        statement1;
        ...
        break;
    case 'value2':
        statement2;
        ...
        break;
    case 'value3' :
        statement3;
        ...
        break;
    ...
    default:
        statementN;
        break;
}
```



Program Example

```
// SwitchCase.cpp
// Switch ..case example
#include "stdafx.h"
#include <iostream>
using namespace std;
void main()
{
    int day;
    cout << "Enter the day of the Week: ";
    cin>> day;
    switch(day)
```



Program Example - Continued

```
{
case 1:
    cout << "The day is Monday"<<endl;
    break;
case 2:
    cout << "The day is Tuesday"<<endl;
    break;
case 3:
    cout << "The day is Wednesday"<<endl;
    break;
case 4:
    cout << "The day is Thursday"<<endl;
    break;
```



Program Example - Continued

case 5:

```
cout << "The day is Friday"<<endl;  
break;
```

case 6:

```
cout << "The day is Saturday"<<endl;  
break;
```

case 7:

```
cout << "The day is Sunday"<<endl;  
break;
```

default:

```
cout << "You have entered incorrect day"<<endl;
```

```
}
```

```
}
```



Loops

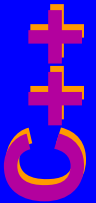
- *The **while** statement*
- *The **do-while** statement*
- *Infinite Loops*
- *The **for** loop*



The while statement

Syntax:

```
while (condition)
    statement;
```



Program Example

```
//program to demonstrate while loop
#include "stdafx.h"
#include <iostream>
using namespace std;
void main()
{
    char reply = 'y';
    int num;
```



Program Example - Continued

```
while (reply == 'y' || reply == 'Y')
{
    cout<< "Enter a number: ";
    cin>>num;
    cout<< "The square of "<< num <<" is "<<
        num*num <<endl;
    cout<< "Do you want to continue (y/n)? ";
    cin>>reply;
}
}
```



Program Output

```
Enter a number: 5
```

```
The square of 5 is 25
```

```
Do you want to continue (y/n)? Y
```

```
Enter a number: 10
```

```
The square of 10 is 100
```

```
Do you want to continue (y/n)? y
```

```
Enter a number: 20
```

```
The square of 20 is 400
```

```
Do you want to continue (y/n)? n
```



The do-while statement

Syntax:

```
do
    statement1;
while (condition);
```



Program Example

```
//program to demonstrate do while loop
#include "stdafx.h"
#include <iostream>
using namespace std;
void main()
{
    char reply;
    int num;
```



Program Example - Continued

```
do
{
    cout<< "Enter a number: ";
    cin>>num;
    cout<< "The square of "<<num <<" is
           "<<num * num<<endl;
    cout<< "Do you want to continue (y/n)?";
    cin>>reply;
}while(reply!='n');
}
```



Infinite Loops

Syntax:

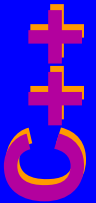
```
while (true)
{
    ...
}
```



Breaking Infinite Loops

Syntax:

```
while (true)
{
    ...;
    if (condition)
        break;
    ...;
}
```



Program Example

```
//program to demonstrate break statement in loops
#include "stdafx.h"
#include <iostream>
using namespace std;
void main()
{
    char reply;
    int count;
    int num;
    count = 1;
```



Program Example - Continued

```
do
{
    cout<< "Enter a number: ";
    cin>>num;
    cout<< "The square of "<<num <<" is "<<
        num * num<<endl;
    count++;
    if (count > 3)
        break;
    cout<< "Do you want to continue (y/n)? ";
    cin>>reply;
} while(reply!='n');
}
```



The for loop

Syntax:

```
for (initialization; condition; expression)
    statement;
```



Program Example

```
//program to demonstrate for loop
#include "stdafx.h"
#include <iostream>
using namespace std;
void main()
{
    cout<<"The squares of first 10 Natural Numbers are "<<endl;
    cout << "Number \t Square" << endl;
    for(int i=1;i<=10;i++)
    {
        cout<<i<<'\t'<<i * i<<endl;
    }
}
```



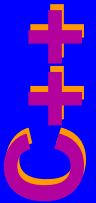
Infinite Loops

- **Ex:**
 - for (initialization; ; expression)
 - for (; ;)



The continue statement

- Skips the remaining program statements in the loop and jumps to the beginning of the loop for next iteration



Program Example

```
// program to demonstrate continue statement
// in loops
#include "stdafx.h"
#include <iostream>
using namespace std;
void main()
{
    char reply;
    int num;
```



Program Example - Continued

```
for (int i=0;i<3;i++)
{
    cout<< "Enter a number: ";
    cin>>num;
    if (num < 0)
        continue;
    cout<< "The square of "<<num <<
        " is :"<<num * num<<endl;
}
}
```



Program Output

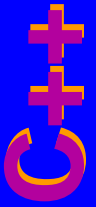
```
Enter a number: 5
```

```
The square of 5 is :25
```

```
Enter a number: -2
```

```
Enter a number: 10
```

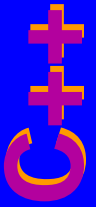
```
The square of 10 is :100
```



The goto statement

Example:

```
/*Line 10:*/ sum = num1 + num2;
/*Line 11:*/ goto PRINTRESULT;
/*
.
.
.*/
/*Line 15:*/ PRINTRESULT:
/*      : */ cout << "Result"<< sum;
```



Syntax

```
goto <label>;
```

```
<label>:
```



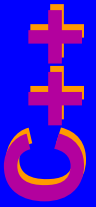
Program example

```
// Program to illustrate use of goto statement
#include "stdafx.h"
#include <iostream>
using namespace std;
void main ()
{
    int sum=0, i=0;
    int num;
```



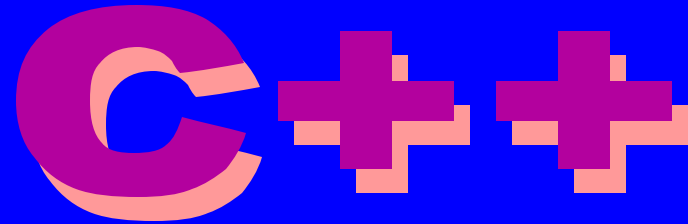
Program example - Continued

```
while (true)
{
    cout<< "Enter a number: ";
    cin>>num;
    if (num == -1)
        goto END;
    cout << "you entered number " << num << endl;
}
END:
    cout << "You entered -1, Quitting ..." << endl;
}
```



Summary

- What you learned?
 - Creating Hello World Program
 - Accepting User Input
 - C++ Language Constructs
 - Conditional Program Statements
 - Program Loops



Conclusion