Introduction to C++

A Brief Summary of GNU C++

Ashik Iqubal

Department of Physics
Ramakrishna Mission Vivekananda University
Belur Math, Howrah

ashik.iqubal@gmail.com

August 31, 2012

C++: Data Types

- Integral Types

Boolean	bool
Integer	short, int, long, unsigned short, unsigned int, unsigned long
Enumeration	enum
Character	char, unsigned char, wchar_t

- Floating point

Floating	float
Double	double
Long Double	long double

C++: Data Type Examples

Boolean	bool variable = true/false
	bool flag = true ;
Enumerate	enum typename = {enumerator list}
	enum Season = {SUMMER, WINTER, AUTUMN, SPRING} ;
Character	char variable = ‘x‘
	char c = ‘A‘;
Constant	const var = num
	const double pi = 3.14159265358979323846 ;
Array	var[array size]
	int ia[10] ;
	int ar[3] = { 2, 4, 5 } ;
	Array indexing starts from 0, ie, the first element of the array is ia[0], ar[0], etc.

C++: Arithmetic Operators

+	Addition
–	Subtraction
*	Multiplication
/	Division
%	Modulus or Remainder Operator

C++: Iterative Operators

- var++
  post increment operator (performs operation on the object after resulting value is used in the surrounding context)
- ++var
  pre increment operator (performs operation on the object before resulting value is used in the surrounding context)
- var– –
- – –var

C++: Composite Operators

Composite Assignment Operators are += , -= , *= , /= , %=
When applied to a variable on the left, each applies the indicated arithmetic operation to it using the value of the expression on the right.

Example

n *= 2; // multiplies n by 2

C++: Conditional Statement

Code

```
if (condition) expression ;
```

expression is evaluated if *condition* is true or if the value of condition is non-zero

Code

```
if (condition1) expression1;
else expression2 ;
```

Code

```
if (condition1) expression1 ;
else if (condition2) expression2 ;
else if (condition3) expression3 ;
else expression4 ;
```

C++: Nested Conditional Statement

Code

```
if (condition1)
if (condition2) expression2 ;
else expression3 ;
else expression4 ;
```

Match each *else* with last unmatched *if*

C++: Switch Statement

Code

```
switch (expression) ;
{
case constant-1 : expressionlist-1 ;
case constant-2 : expressionlist-2 ;
case constant-3 : expressionlist-3 ;
......
case constant-n : expressionlist-n ;
}
```

Evaluates *expression*, looks for it's value among *case constants* and evaluates the corresponding *expressionlist*. *Expression* and *constants* must be integral types.

C++: Conditional Statement Operator

Code

```
condition ? expression1 : expression2
```

Evaluates *expression1* if *condition* is true, else evaluates *expression2*

Example

```
min = ( x < y ? x : y )
```

C++: Statement Block

Sequence of statements that can be used like a single one anywhere in the program.

Code

```
{ expression1 ; expression2; expression3; ........; }
```

C++: Comparison Operators

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

C++: Logical Operators

&&	AND
||	OR
!	NOT

C++: While Loops

Code

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

While *condition* is true or non-zero, *statement* is executed repeatedly (till *condition* is false or zero)

Code

```
while (condition) { if (break-condition) break ; statement ; }
```

Code

```
while (condition) { if (break-condition) exit(0) ; statement ; }
```

statement is executed repeatedly as long as *condition* is true, except, the loop is terminated immediately once *break-condition* is true

C++: Do While Loops

Code

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

(Repeatedly) executes *statement* then evaluates *condition* until *condition* is false or zero

C++: For Loops

Code

for (initialization; condition; update) statement ;

1. Evaluate *initialization*
2. If *condition* is true, terminate the loop
3. Execute *statement*
4. Evaluate the *update* expression
5. Repeat steps 2 - 4

Example

```
for ( index = 9 ; index >= 0 ; – –index )
cout << ia[index] << " " ;
```

C++: Break, Continue and Goto Statements

- *break* statement exits loops block, jumping immediately to the next statement outside of the loop.
- *continue* statement continues the loop after skipping the remaining statements in its current iteration.
- *goto* statement transfers control to another part of the program

C++: Goto Statements

Code

if (condition) goto *label* ;

....

......

....

label : statement

C++: Functions

Code

```
return-type function-name(argument declarations)
{ declarations and statements ;
return expression ; }
```

If the function does not return any value, then it's *return-type* must be declared *void*

Example

```
int fact(int n)
{ return ( n > 1 ) ? n*fact(n-1) : 1 ; }
```

C++: Functions contd.

Functions need to be declared in the main() before using

Code

```
return-type function-name(argument type)
```

Example

```
int min(int, int)
```