**WORKING WITH SIMPLE DATA TYPES and NUMERIC EXPRESSIONS**

**C++ INTEGRAL TYPES -> CHAR, SHORT, INT, LONG, BOOL and UNSIGNED INTEGERS\* C++ FLOATING TYPES -> FLOAT, DOUBLE and LONG DOUBLE**



C++ Data Types

C++ Simple Data Types

**simple**

**structured**

**simple types**

**integral enum**

**char short int long bool**

**floating**

**array struct union class**

**integral floating**

**char short int long bool enum float double long double**

**float double long double**

**address**

**unsigned**

**pointer reference**

1 4

**Some C++ Operators**

**Precedence Operator Description**

|  |  |
| --- | --- |
| ***Higher***  ***Lower*** | ( ) Function call |
| + Positive  - Negative |
| \* Multiplication  / Division  % Modulus(remainder) |
| + Addition  - Subtraction |
| = Assignment |

**Operators can be**

**binary involving 2 operands 2 + 3 unary involving 1 operand - 3**

**ternary involving 3 operands *later***

8 18



**Precedence**



 **Higher Precedence determines which operator is applied first in an expression having several operators**

**Associativity**

 **Left to right associativity means that in an expression having 2 operators with the same priority, the left operator is applied first**

 **In C++ the binary operators**

**\*, /, %, +, - are all left associative**

 **Expression 9 - 5 - 1 means(9 - 5) - 1**

**4 - 1**

**3**

19 20

**Evaluate the Expression**

7 \* 10 - 5 % 3 \* 4 + 9

(7 \* 10) - 5 % 3 \* 4 + 9

70 - 5 % 3 \* 4 + 9

70 -(5 % 3) \* 4 + 9

70 - 2 \* 4 + 9

70 -( 2 \* 4) + 9

70 - 8 + 9 (70 - 8 ) + 9

62 + 9

71

21

**Parentheses**

 **Parentheses can be used to change**

**the usual order**

 **Parts in() are evaluated first**

 **Evaluate** (7 \*(10 - 5) % 3) \* 4 + 9

(7 \* 5 % 3 ) \* 4 + 9

( 35 % 3) \* 4 + 9

2 \* 4 + 9

8 + 9

17

22

**Recall Assignment Operator Syntax**

***What value is stored?***

**Variable = Expression**

 **First, Expression on right is evaluated**

 **Then the resulting value is stored in**

**the memory location of Variable on left**

**NOTE: An automatic type coercion occurs after evaluation but before the value is stored if the types differ for Expression and Variable**

float a;

float b;

a = 8.5;

**a**

b = 9.37;

a = b; **b**

**8.5 a ?**

**9.37 b ?**

23 24

***What is stored?***

***What is stored?***

float someFloat;

**?**

someFloat

someFloat = 12; // Causes implicit type conversion

int someInt;

**?**

someInt

someInt = 4.8; // Causes implicit type conversion

**12.0 4**

someFloat

25

someInt

26



Files used to demonstrate topic using a C++ compiler

Datatypes.cpp - demonstrates how to declare and use different C++ simple datatypes Precedence.cpp - program walkthrough applying operator precedence rules Expressions.cpp – program walkthrough of several arithmetic expressions

Typeconversion.cpp – demonstrates type coercion (implicit conversion) and type casting (explicit conversion)

// Program datatype demonstrates how to declare and use different C++ simple data types

#include <iostream>

using namespace std;

int main()

{

// This section uses an integer data type. int oranges;

oranges = 20;

cout << "Today we sold " << oranges << " bushels of oranges.\n";

// This section has variables of several of the integer types. int checking;

unsigned int miles;

long days;

checking = -100; miles = 3567; days = 189000;

cout << "We have made a long journey of " << miles;

cout << " miles.\n";

cout << "Our checking account balance is " << checking;

cout << "\nAbout " << days << " days ago Columbus ";

cout << "stood on this spot.\n";

// This section demonstrates boolean variables. bool boolValue;

boolValue = true;

cout << boolValue << endl;

boolValue = false;

cout << boolValue << endl;

// This section uses character literals. char letter;

letter = 'A';

cout << letter << '\n';

letter = 'B';

cout << letter << '\n';

// This section also uses character literals but shows the ASCII value (sse Appendix E pg. 1029)

letter = 65;

cout << letter << endl;

letter = 66;

cout << letter << endl;

// This section uses floating point data types. float distance;

double mass;

double shares = 220.0;

double avgPrice = 14.67;

distance = 1.495979E11;

mass = 1.989E30;

cout << "The Sun is " << distance << " meters away.\n"; cout << "The Sun\'s mass is " << mass << " kilograms.\n"; cout << "There were " << shares << " shares sold at $"; cout << avgPrice << " per share.\n";

//this section demonstrates the size of integers, long integers, and long doubles per your machine long double apple;

cout << "The size of an integer is " << sizeof(int);

cout << " bytes.\n";

cout << "The size of a long integer is " << sizeof(long);

cout << " bytes.\n";

cout << "An apple can be eaten in " << sizeof(apple);

cout << " bytes!\n";

system ("PAUSE");

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Program precedence demonstrates the precedence of operators

#include <iostream>

using namespace std;

int main ()

{

cout << 4 + 3 \* 5 << endl; cout << (4 + 3) \* 5 << endl; cout << 4 \* 5 % 3 + 2 << endl;

cout << (4 \* (5 / 3) + 2) << endl;

system("PAUSE");

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Program expressions demonstrates arithmetic expressions

#include <iostream>

#include <iomanip> // For setw() and setprecision()

using namespace std;

int main()

{

int i = 4; int j = 17; float z = 2.6;

cout << fixed << showpoint; // Set up floating pt.

// output format

cout <<"Variables declared and assigned below"<<endl;

cout << "int i = 4;";

cout << "int j = 17;";

cout <<"float z = 2.6;"<<endl<<endl;

cout<<"Testing various arithmetic expressions"<<endl;

cout << "i/ float (j)= "<< i/ float (j) <<endl; cout << "1.0 / i +2 = "<< 1.0 / i +2 <<endl; cout << "z \* j = "<< z \* j <<endl;

cout << "i + j % i = "<< i + j % i <<endl; cout << "(1/2)\*i = "<< (1/2)\*i <<endl; cout << "2\*i+j-1 = "<< 2\*i+j-1 <<endl; cout << "j/2 = "<< j/2 <<endl;

cout << "2\*3-1%3 = "<< 2\*3-1%3 <<endl;

cout << "i % j/i = "<< i % j/i <<endl;

cout << "int(z + 0.5) = "<< int(z + 0.5) <<endl;

system("PAUSE");

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Program type\_conversion demonstrates type coercion (implicit conversion) and type casting (explicit type conversion

#include <iostream>

#include <iomanip> // For setw() and setprecision()

using namespace std;

int main()

{

float someFloat = 12.0; float anotherFloat = 5.5; int someInt = 4;

cout << fixed << showpoint; // Set up floating pt.output format someFloat = 3 \* someInt + 2; //type coercion

cout << someFloat <<endl;

someFloat = float (3\*someInt + 2); //type casting or conversion cout << someFloat <<endl;

someInt = 5.2 /someFloat - anotherFloat; //type coercion cout <<someInt <<endl;

someInt = int(5.2 /someFloat - anotherFloat); //type casting or conversion cout <<someInt <<endl;

return 0;

}