**COP 3331 Week 2 Examples:**

1. Function Example (with random number generation)

#include <iostream>

#include <cstdlib> //contains prototypes for functions srand and rand

#include <ctime> // needed for time function

using namespace std;

int larger (int x, int y); // function prototype

int main()

{

// seed random number generator with system clock

srand(static\_cast <unsigned int> (time(0)));

int number1 {1 + rand() % 100};

int number2 {1 + rand() % 100};

int lrg {larger(number1, number2)}; // yup, that's a function call

// in an initialization list

cout << "The numbers selected were: " << number1 << " and "

<< number2 << endl;

cout << "The larger number is " << lrg << endl;

}

int larger (int x, int y)

{

if (x > y)

return x;

return y;

}

1. Function Example (Value Parameters vs. Reference Parameters)

#include <iostream>

using namespace std;

int squareByValue(int); // prototype (value pass)

void squareByReference(int&); // prototype (reference pass)

int main()

{

int x{2}; // value to square using squareByValue

int z{4}; // value to square using squareByReference

// demonstrate squareByValue

cout << "x = " << x << " before squareByValue\n";

cout << "Value returned by squareByValue: "

<< squareByValue(x) << endl;

cout << "x = " << x << " after squareByValue\n" << endl;

// demonstrate squareByReference

cout << "z = " << z << " before squareByReference" << endl;

squareByReference(z);

cout << "z = " << z << " after squareByReference" << endl;

}

int squareByValue(int number)

{

return number \*= number;

}

void squareByReference(int& numberRef)

{

numberRef \*= numberRef;

}

1. Function and Scope Example

#include <iostream>

using namespace std;

void useLocal();

void useStaticLocal();

void useGlobal();

int x{1}; // global variable

int main()

{

cout << "global x in main is " << x << endl;

int x{5}; // local variable to main

cout << "local x in main's outer scope is " << x << endl;

{ // block starts a new scope

int x{7}; // hides both x in outer scope and global x

cout << "local x in main's inner scope is " << x << endl;

}

cout << "local x in main's outer scope is " << x << endl;

useLocal(); // useLocal has local x

useStaticLocal(); // useStaticLocal has static local x

useGlobal(); // useGlobal uses global x

useLocal(); // useLocal reinitializes its local x

useStaticLocal(); // static local x retains its prior value

useGlobal(); // global x also retains its prior value

cout << "\nlocal x in main is " << x << endl;

}

void useLocal()

{

int x{25}; // initialized each time useLocal is called

cout << "\nlocal x is " << x << " on entering useLocal" << endl;

++x;

cout << "local x is " << x << " on exiting useLocal" << endl;

}

void useStaticLocal()

{

static int x{50}; // initialized first time useStaticLocal is called

cout << "\nlocal static x is " << x << " on entering useStaticLocal"

<< endl;

++x;

cout << "local static x is " << x << " on exiting useStaticLocal"

<< endl;

}

void useGlobal()

{

cout << "\nglobal x is " << x << " on entering useGlobal" << endl;

x \*= 10;

cout << "global x is " << x << " on exiting useGlobal" << endl;

}

1. Scope Resolution Operator Example

#include <iostream>

using namespace std;

int number{7}; // global variable named number

int main()

{

double number{10.5}; // local variable named number

// display values of local and global variables

cout << "Local double value of number = " << number

<< "\nGlobal int value of number = " << ::number << endl;

}

1. Function Overloading Example

// Overloaded square functions.

#include <iostream>

using namespace std;

int square(int x)

{

cout << "square of integer " << x << " is ";

return x \* x;

}

double square(double y) {

cout << "square of double " << y << " is ";

return y \* y;

}

int main()

{

cout << square(7); // calls int version

cout << endl;

cout << square(7.5); // calls double version

cout << endl;

}

1. Array Example (Standard and with Template)

#include <iostream>

#include <iomanip> // for setw

#include <array>

using namespace std;

int main()

{

const int SIZE = 3;

int a[SIZE] = {8, 1, 3};

cout << "Array a: ";

for (int i = 0; i < SIZE; i++)

cout << setw(5) << a[i];

cout << endl << endl;

array <int, 3> b {4, 2, 7};

cout << "Array b: ";

for (size\_t i {0}; i < b.size() ; i++)

cout << setw(5) << b[i];

cout << endl;

}