ECOR 1606 Final Exam Crib Sheet

CONTROL STRUCTURES

simple

if:

```
if (boolean exp) {
   statements // body
}
```

if-then-else:

```
if (boolean exp) {
   statements // true part
} else {
   statements // false part
}
```

multi-way if:

```
if (boolean exp) {
   statements // part 1
} else if (boolean exp) {
   statements // part 2
} else if (boolean exp) {
   statements // part 3
… // and so on
} else { // an else part is
            optional
   statements // else part
}
```

while loop (pre-test):

```
while (boolean exp) {
   statements // body
}
```

do-while loop (post-test):

```
do {
   statements // body
} while (boolean exp);
```

for loop:

```
for (exp1; exp2; exp3) {
   statements // body
}
```

is equivalent to:

```
exp1;
while (exp2) {
   same statements
   exp3;
}
```

break statement:

```
break;
```

– causes an exit from the enclosing loop.

continue statement:

```
continue;
```

– sends control back to the top of the enclosing loop.

CALL BY …

```
int sample (int a, int &b, int c[]);
```

"a" is call-by-value (just like a regular variable but given a value when the function is called).
"b" is call-by-reference. all operations on "b" actually operate on the variable supplied.
"c" is call-by-reference. all operations on "c" actually operate on the array supplied.

ARRAYS

```
// sample declaration
int a[4] = {1, 2, 4, 12 };

// typical use
sum = 0;
for (i = 0; i < array_size; i++) {
   sum += a [i];
}

// typical function
void write_array(int a[], int n) {
   int i;
   for (i = 0; i < n; i++) {
      cout << a[i] << endl;
   }
}
```

MODEL PROGRAM

```
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;

int add(int x, int y) {
     int result;
     result = x + y;
     return result;
}

int  main() {
     int a, b, c;
     cout << "Enter two values: ";
     cin >> a >> b;
     c = add(a, b);
     cout << "The answer is " << c << endl;
     system("PAUSE");
     return 0;
}
```

EXPRESSIONS

```
 <   is less than
 >   is greater than
<=   is less than or equal to
>=   is greater than or equal to
==   is equal to
!=   is not equal  to
 ||  OR (either  side is true)
&&   AND (both sides are true)
 !   NOT (changes true to false, false to true)
```

```
 %   modulus (gives remainder from division)
X++  means "use value of X, then increment X"
X--  means "use value of X, then decrement X"
++X  means "increment X, then use new value"
--X  means "decrement X, then use new value"
X += Y  is equivalent  to X = X + (Y)
      (same idea for -=, *=. and /=)
```

INPUT (use iostream, fstream)

```
ifstream xin;       // declares an input stream object (for reading files)
xin.open(string);// attaches the input stream object to the file specified
xin.fail()          // returns true if the stream is in the failed state (something's gone wrong)
xin.eof()           // returns true if the program has tried to read past the end of the file
xin.clear()         // resets the failure flag
xin.ignore(count, ch);     // discards input characters until  "count" characters have been discarded
                           // or  character "ch" has been discarded (whichever comes first)
```

OUTPUT (use iostream, fstream, iomanip)

```
ofstream xout;   // declares an output stream object (for writing files)
xout << setiosflags(ios::fixed|ios::showpoint);  // forces use of non-scientific notation and the display of zeroes
                                                 // to the right of the decimal point. this remains in effect until changed.
xout << setprecision(value);       // selects the number of digits to be displayed to the right of the decimal point
                                   // when outputting double values.  the choice remains in effect until changed.
xout << setw (value) << …          // indicates that the next value output is to occupy the specified number of
                                   // columns.  a one shot deal – affects only the next value output
xout << setfill(ch);               // selects the fill character to be used when padding output values to a specified width.
                                   // the choice remains in effect until changed.
xout.fill();                       // returns the current fill character.
```

LIBRARY FUNCTIONS

```
double fabs(double x);   returns the absolute value of "x", for real numbers
int abs(int x);      returns the absolute value of "x", for integers
double log (double x);   natural log (log base e)
double log10(double x); log base 10
double exp(double x);    returns "e" to power of  "x"
double sqrt(double x);   returns the square root of "x"
double pow (double x, double y);          returns  "x" to the power of "y"
double sin(double x);    returns the sine of "x" (note:  "x" is in radians)
double cos(double x);    returns the cosine of "x" (note:  "x" is in radians)
double asin(double x);   returns the inverse sin of "x" (in radians)
double acos(double x);   returns the inverse cosine of "x" (in radians)
double sinh(double x);   hyperbolic sin
double cosh(double x);   hyperbolic cosine
```