

Lafore: Object-Oriented Programming in C++ (4th)

Chapter 2: C++ Programming Basics

2.1-First.cpp

```
#include <iostream> // contains information to support input / output
using namespace std;

void main() // entry point for every C and C++ program (console)
{
cout << "Every age has a language of its own\n"; // output to standard device
} // end main()
```

2.2-Comments.cpp

```
// comments.cpp -- demonstrates comments
#include <iostream> // preprocessor directive
using namespace std; // "using" statement

void main() // function name "main"
{ // start function body
cout << "Every age has a language of its own\n"; // statement
} // end main() // end function body
```

2.3-Intvars.cpp

```
// intvars.cpp -- demonstrates integer variables
#include <iostream>
using namespace std;

void main()
{
int nVar1; // declare and define nVar1 (allocates space in memory)
int nVar2; // declare and define nVar2 (allocates space in memory)

nVar1 = 20; // assign value to nVar1 (stores 20 in memory location)
nVar2 = nVar1 + 10; // assign value to nVar2 (takes value in nVar1 adds 10 to it)
cout << "nVar1+10 is "; // output text
cout << nVar2 << endl; // output contents of nVar2
} // end main()
```

2.4-Charvars.cpp

```
// charvars.cpp -- demonstrates character variables
#include <iostream> // for cout, etc.
using namespace std;

void main()
{
char cVar1 = 'A'; // declare and define char variable as letter A
char cVar2 = '\t'; // declare and define char variable as tab

cout << cVar1; // display character
cout << cVar2; // display character
cVar1 = 'B'; // set char variable to char constant
cout << cVar1; // display character
cout << '\n'; // display newline character
} // end main()
```

2.5-Fahren.cpp

```
// fahren.cpp -- demonstrates cin, newline
#include <iostream>
using namespace std;

void main()
{
int nTempFahr; // for temperature in fahrenheit

cout << "Enter temperature in fahrenheit: ";
cin >> nTempFahr;
int nTempCelsius = (nTempFahr-32) * 5 / 9;
cout << "Equivalent in Celsius is: " << nTempCelsius << '\n';
} // end main()
```

2.6-Circarea.cpp

```
// circarea.cpp -- demonstrates floating point variables
#include <iostream>                // for cout, etc.
using namespace std;

void main()
{
float fRadius;                    // variable of type float
const float PI = 3.14159F;       // type const float

cout << "Enter radius of circle: "; // prompt
cin >> fRadius;                  // get radius
float fArea = PI * fRadius * fRadius; // find fArea
cout << "Area is " << fArea << endl; // display answer
} // end main()
```

2.7-Width1.cpp

```
// width1.cpp -- demonstrates need for setw manipulator
#include <iostream>
using namespace std;

void main()
{
long IPopulation1=2425785, IPopulation2=47, IPopulation3=9761;

cout << "LOCATION " << "POP." << endl
    << "Portcity " << IPopulation1 << endl
    << "Hightown " << IPopulation2 << endl
    << "Lowville " << IPopulation3 << endl;
} // end main()
```

2.8-Width2.cpp

```
// width2.cpp -- demonstrates setw manipulator
#include <iostream>
#include <iomanip> // for setw()
using namespace std;

void main()
{
long IPopulation1=2425785, IPopulation2=47, IPopulation3=9761;

cout << setw(8) << "LOCATION"<< setw(12) << "POPULATION" << endl
     << setw(8) << "Portcity " << setw(12) << IPopulation1 << endl
     << setw(8) << "Hightown " << setw(12) << IPopulation2 << endl
     << setw(8) << "Lowville " << setw(12) << IPopulation3 << endl;
} // end main()
```

2.9-Signtest.cpp

```
// signtest.cpp -- tests signed and unsigned integers
#include <iostream>
using namespace std;

void main()
{
int nSignedVar = 1500000000; // signed (1,500,000,000)
unsigned int unUnSignedVar = 1500000000; // unsigned

nSignedVar = (nSignedVar * 2) / 3; // calculation exceeds range
unUnSignedVar = (unUnSignedVar * 2) / 3; // calculation within range

cout << "nSignedVar = " << nSignedVar << endl; // wrong value
cout << "unUnSignedVar = " << unUnSignedVar << endl; // correct value
} // end main()
```

2.10-Mixed

```
// mixed.cpp -- shows mixed expressions
```

```
#include <iostream>
```

```
using namespace std;
```

```
void main()
```

```
{
```

```
int nCount = 7;
```

```
float fAverageWeight = 155.5F;
```

```
double dfTotalWeight = nCount * fAverageWeight; // implied conversion of data
```

```
cout << "dfTotalWeight=" << dfTotalWeight << endl;
```

```
cout << "Size of nCount: " << sizeof(nCount) << " bytes" << endl;
```

```
cout << "Size of fAverageWeight: " << sizeof(fAverageWeight) << " bytes" << endl;
```

```
cout << "Size of dfTotalWeight: " << sizeof(dfTotalWeight) << " bytes" << endl;
```

```
} // end main()
```

2.11-Cast.cpp

```
// cast.cpp -- tests signed and unsigned integers
```

```
#include <iostream>
```

```
using namespace std;
```

```
void main()
```

```
{
```

```
int nVar = 1500000000;
```

```
// 1,500,000,000
```

```
nVar = (nVar * 10) / 10;
```

```
// result too large
```

```
cout << "nVar = " << nVar << endl;
```

```
// wrong answer
```

```
nVar = 1500000000;
```

```
// nVar = (static_cast<double>(nVar) * 10.0) / 10.0; // newer cast to double
```

```
nVar = (double(nVar) * 10.0) / 10.0;
```

```
// older cast to double
```

```
cout << "nVar = " << nVar << endl;
```

```
// right answer
```

```
} // end main()
```

2.12-Remaind.cpp

```
// remaind.cpp -- demonstrates remainder operator
#include <iostream>
#include <stdlib.h>
using namespace std;

void main()
{
cout << 6 % 8 << endl           // 6
   << 7 % 8 << endl           // 7
   << 8 % 8 << endl           // 0
   << 9 % 8 << endl           // 1
   << 10 % 8 << endl;        // 2

cout << "Numbers from 0 to 32767 (0x7fff)\n";
cout << rand( ) << endl;      // random number generator
cout << rand( ) << endl;
cout << rand( ) << endl;
cout << rand( ) << endl;

cout << "Numbers from 0 to 99\n";
cout << rand( ) % 100 << endl;
cout << rand( ) % 100 << endl;
cout << rand( ) % 100 << endl;
cout << rand( ) % 100 << endl;
} // end main()
```

2.13-Assign.cpp

```
// assign.cpp -- demonstrates arithmetic assignment operators
#include <iostream>
using namespace std;

void main()
{
    int nAnswer = 27;

    nAnswer += 10;           // same as: nAnswer = nAnswer + 10;
    cout << nAnswer << ", ";
    nAnswer -= 7;           // same as: nAnswer = nAnswer - 7;
    cout << nAnswer << ", ";
    nAnswer *= 2;           // same as: nAnswer = nAnswer * 2;
    cout << nAnswer << ", ";
    nAnswer /= 3;           // same as: nAnswer = nAnswer / 3;
    cout << nAnswer << ", ";
    nAnswer %= 3;           // same as: nAnswer = nAnswer % 3;
    cout << nAnswer << endl;
} // end main()
```

2.14-Increm.cpp

```
// increm.cpp -- demonstrates the increment operator
#include <iostream>
using namespace std;

void main()
{
    int nCount = 10;

    cout << "nCount=" << nCount << endl;           // displays 10
    cout << "nCount=" << ++nCount << endl;         // displays 11 (prefix)
    cout << "nCount=" << nCount << endl;           // displays 11
    cout << "nCount=" << nCount++ << endl;         // displays 11 (postfix)
    cout << "nCount=" << nCount << endl;           // displays 12
} // end main()
```

2.15-sqrt.cpp

```
// sqrt.cpp -- demonstrates sqrt() library function
#include <iostream>           // for cout, etc.
#include <cmath>             // for sqrt()
using namespace std;

void main()
{
    double dNumber, dAnswer;           // sqrt() requires type double

    cout << "Enter a Number: ";
    cin >> dNumber;                   // get the value for dNumber
    dAnswer = sqrt(dNumber);          // find square root
    cout << "Square root is " << dAnswer << endl; // display it
} // end main()
```