

THE C++ LANGUAGE

- Minimalist approach, Uses library functions
- C++ is case sensitive.
- All C++ keywords in lower case.
- C++ program is series of statements.
- Comments syntax:
 - /* ... */ for C
 - // for C++ comment to end of line
- White space = spaces, tabs, and new lines.

C++ IDENTIFIERS

- Describe data object and functions
- Max Length = 32 Characters
- C++ Identifier Naming Restrictions:
 - A to Z a to z 0 to 9 _
 - No spaces allowed
 - Must begin with letter or underscore
 - Case sensitive
 - Not a reserved word (See Appendix A)

Identifier Naming Conventions:

- Library functions use lower case letters
- Avoid using _ for first letter
- User functions use title case no spaces
- Constants use ALL CAPITAL letters.
- Use descriptive names (self documenting)
- Data object begins with lower case letter(s) to indicate data type per following table, and remaining name in title case
 - e.g. cGrade fGradePointAverage InWidgets

Identifier Naming Conventions (Cont'd)

- Data object begins with following lower case letter(s) to indicate data type

PREFIX	DATA TYPE (Bit Length for Turbo C++)
c	Char (8 bits)
n	Integer (1 word = 16 bits or 32 bits)
sn	Short Integer (16 bits)
ln	Long Integer (32 bits)
f	Float (32 bits)
df	Double Float (64 bits)
un	Unsigned Integer
sz	String terminated with zero byte

DECLARATION STATEMENTS

- Reserves space in memory for data object
- Specifies data type and data object identifier
- Terminate using a semicolon;
As with all statements in C++
- Use identifier naming convention
- Optionally may declare more than one data object of the same data type
- Optionally, may include data object initialization in declaration statement

Copyright © 2001 R.M. Laurie 5

Integer Data Type Declarations - 1

●char

Reserves 8 bits of RAM memory which can represent:

- ASCII character
- Signed integer in range 127 to -128 (Default)
- Unsigned integer in range 255 to 0

■ Examples:

```
char cLetter;
char cGradePoints, cGrade;
unsigned char cWeekNumber = 1;
char c1stLetter = 65, c2ndLetter = 'B';
```

Copyright © 2001 R.M. Laurie 6

Integer Data Type Declarations - 2

●int

Reserves one **word** of RAM memory which can represent:

- 16 bits for Turbo C++
 - ◆ Signed integer 32,767 to -32,768 (Default)
- 32 bits for Visual C++
 - ◆ Signed integer $\pm 2,147,483,648$ (Default)

■ Examples:

```
int nScore;
int nTotalScore, nClassMedian;
unsigned int unHeight = 100, unWidth = 50;
```

Copyright © 2001 R.M. Laurie 7

Integer Data Type Declarations - 3

●long

Reserves 32 bits of RAM memory

- Signed integer $\pm 2,147,483,648$ (Default)

■ Examples:

```
long lnSSN;
long lnAltitude, lnDistance = 0;
```

●short

Reserves 16 bits of RAM memory

- Signed integer 32,767 to -32,768 (Default)

■ Example:

```
short snNumber = 1;
```

Copyright © 2001 R.M. Laurie 8

Float Data Type Declarations

•float

Reserves 32 bits of RAM memory

- Represents floating point values in range:
 $\pm 3.402823466 \times 10^{\pm 38}$
- Examples:
float fWeight = 2.35;

•double

Reserves 64 bits of RAM memory

- Represent floating point values in range:
 $\pm 1.7976931348623158 \text{ E} + 308$
- Examples:
double dfWeightMetricTonsEarth;

Copyright © 2001 R.M. Laurie 9

Constant Data Type Declarations

•const

Reserves the appropriate RAM memory for the associated data type and will not let program change its value.

- Identifier naming convention is all capitals
- Examples:
const char DOLLAR = '\$';
const float PI = 3.14159;
const int MAXSCORE = 100;

Copyright © 2001 R.M. Laurie 10

OPERATORS

- Used to perform operations on data objects and assign result to a variable
- Order defined in Appendix B table defines precedence
- Note that the assignment operator "=" is defined last and therefore done last.
- For readability insert parenthesis if order is not apparent in source code.

Copyright © 2001 R.M. Laurie 11

Operators Precedence (Highest to Lowest)

()	Parenthesis defines order of operation
sizeof()	How many bytes
* / %	Multiplication, Division, Modulus
+ -	Addition, Subtraction
<< >>	Insertion, Extraction
=	Assignment

Copyright © 2001 R.M. Laurie 12

Operator Usage Examples

```

nScore = 93;
nTotalScore = nTotalScore + nScore;
nScoreCount = nScoreCount + 1;
fAvgScore = nTotalScore/nScoreCount;
nRemainder = nTotalScore % nSoreCount;
nSize = sizeof(nScoreCount);
cout << "Total = " << nTotal + 1;
cout << sizeof(nTotal)<< " Bytes";
cin >> nChipsBlack;

```

Copyright © 2001 R.M. Laurie 13

Program Output

- C++ does not have any formal way to output information for the user.
- However you may use library routines to communicate with the user.
- Stream I/O Functions are available in the `iostream.h` header file.
 - You must include this header file before `main()`
 - `cout` is used for information output

Copyright © 2001 R.M. Laurie 14

cout = Program Output - 1

- `cout` represents the default data output device that is by default the user display.
- The `cout` function with the insertion operator (`<<`) is used to send ASCII text to the user display.
- Examples:


```

cout << "Hello" << " Bob";
cout << sizeof(nTotal)<< " Bytes";
cout << "Total = " << nTotal + 1;

```

Copyright © 2001 R.M. Laurie 15

cout = Program Output - 2

- `endl` The new line command
 - Examples:


```

cout << "Hello" << endl;
cout << "123" << endl << "abc" << endl;

```
- Text string special characters

<code>\n</code> = newline	<code>\r</code> = carriage return	<code>\t</code> = tab
<code>\a</code> = bell	<code>\"</code> = double quote	<code>\?</code> = question
<code>\\</code> = backslash	<code>\'</code> = single quote	<code>\x###</code> = hex

 - Examples:


```

cout << "Hello\t" << "I'm Bob\n\a";
cout << "123" << endl << "abc\n";

```

Copyright © 2001 R.M. Laurie 16

THE C++ PROGRAM

General form. This program does nothing.

```
int main(void)    //Program start
{                //Program body start
    //STATEMENTS GO HERE

    return 0;    //Program ends
}                //Program body end
```

Copyright © 2001 R.M. Laurie 17

Program Comment Title Block

```
/* *****
 * PROJECT 1: THE FIRST PROGRAM
 * *****
 * This program prints out the size of several
 * different data types in bytes.
 *
 * Programmer: (Your Name?)
 * CMIS 102
 * Date: ?
 * ***** */
```

Copyright © 2001 R.M. Laurie 18

Declaration and Initialization Sections

```
#include <iostream.h>    // Note: Required...

int main(void)
{
    // DECLARATION SECTION
    short snSmallInteger;
    int nStdInteger, nSize;
    float fRealNumber;
    char cLetter;

    // INITIALIZATION SECTION
    nStdInteger = 100000;
    fRealNumber = -0.000123456789;
    cLetter = 'A';
```

Copyright © 2001 R.M. Laurie 19

Combined Declaration and Initialization

```
#include <iostream.h>    // Note: Required...

int main(void)
{
    // DECLARATION AND INITIALIZATION SECTION
    short snSmallInteger;
    int nStdInteger = 100000, nSize;
    float fRealNumber = -0.000123456789;
    char cLetter = 'A';
```

Copyright © 2001 R.M. Laurie 20

Processing Sections - 1

```
// PROCESSING SECTION
snSmallInteger = nStdInteger;
nSize = sizeof(snSmallInteger);

cout << "Short Integer = "<< snSmallInteger;
cout << " and is " << nSize << " Bytes."<< endl;

cout << "Standard Integer = "<< nStdInteger;
cout << " and is " << sizeof(nStdInteger) << "
Bytes."<< endl;

cout << "Real Number = "<< fRealNumber;
cout << " and is " << sizeof(fRealNumber) << "
Bytes."<< endl;
```

```
Short Integer = -31072 and is 2 Bytes.
Standard Integer = 100000 and is 4 Bytes.
Real Number = -0.000123457 and is 4 Bytes.
```

Copyright © 2001 R.M. Laurie | 21

Processing Sections - 2

```
nSize = sizeof(cLetter);
cout << "Letter = "<< cLetter;
cout << " and is " << sizeof(cLetter) << " Bytes."<<
endl;

cout << "Another Letter = "<< cLetter+10 << endl;
cout << "37 / 7 = "<< 37/7 << endl;
cout << "37 % 7 = "<< 37%7 << endl << endl;

return 0; // Note: ANSI version requires the
return
}
```

```
Letter = A and is 1 Bytes.
Another Letter = 75
37 / 7 = 5
37 % 7 = 2
```

Copyright © 2001 R.M. Laurie | 22