



- ### Simple Data Types
- **Integral**
    - char
      - unsigned
    - short
      - unsigned
    - int
      - unsigned
    - long
      - unsigned
    - bool bYes= true/false
  - **Floating Point**
    - float
    - double
    - long double
  - **Enumeration**
    - enum
  - See p 497 and 500 for ranges
  - See p 498 to 502 for float compiler symbols
- Copyright © 2001 R.M. Laurie 2

### Identifier Naming Conventions

- 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)
b	Boolean (8 bits)
f	Float (32 bits)
df	Double Float (64 bits)
ldf	Long Double Float (80 bits)
un	Unsigned Integer
e	Enumeration

Copyright © 2001 R.M. Laurie 3

- ### Integral 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' c3rdLetter = "0x43;
            
```
- Copyright © 2001 R.M. Laurie 4

- ### Character Data
- Two character sets
    - EBCDIC = IBM Mainframe Computers
    - ASCII = The rest of the world
  - External Representation  
`char cSomeChar = 'A';`
  - Internal Representation  
`char cSomeChar = 65;`  
`char cSomeChar = 0x41;`
- Copyright © 2001 R.M. Laurie 5

- ### Character Data
- **Control Characters (escape sequence)**

```

\n = newline    \r = carriage return    \f = formfeed
\t = tab       \b = backspace    \a = bell
\' = quote     \" = double quote  \\ = backslash
\xddd = hexadecimal equiv.  \0 = null
            
```
  - **Example:**

```

cout <<"Hello\t"
    <<"I\'m Bob\n\a";
            
```
- Copyright © 2001 R.M. Laurie 6

## Character Functions

```
#include <cctype> (See p. A3)
isalnum(ch)
isalpha(ch)
iscntrl(ch)
isdigit(ch)
isxdigit(ch)
isprint(ch)

islower(ch)
isupper(ch)
ispunct(ch)

tolower(ch)
toupper(ch)

if(toupper(cInp) == 'Y')
```

Copyright © 2001 R.M. Laurie 7

## Integral 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 8

## Integral 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 9

## Integral Data Type Declarations - 4

- **bool**  
Reserves 8 bits of RAM memory
  - Values are true/false or 1/0
  - Examples:
 

```
bool bYes;
bool bFalse = 0, bOn = true, bClose = 1;
```
  - When used with cout will display 0 or 1
 

```
bool bYes = false;
cout << "Yes? = " << bYes; // Yes? = 0
```

Copyright © 2001 R.M. Laurie 10

## Float Data Type Declarations

- **float**  
Reserves 32 bits of RAM memory
  - Represents floating point values in range:  $\pm 3.402823 \times 10^{\pm 38}$
- **double**  
Reserves 64 bits of RAM memory
  - Represent floating point values in range:  $\pm 1.79769313486231 \text{ E} + 308$
  - Examples: 

```
double dfWeightMetricTonsEarth;
```
- **long double**  
Reserves 80 bits of RAM memory
  - Represent floating point values in range:  $\pm 1.797693134862315830 \text{ E} + 4932$
  - Examples:
 

```
long double dfWeightMetricTonsEarth;
```

Copyright © 2001 R.M. Laurie 11

## enum Data Type

- User defined data type whose domain is an ordered set of literal values expressed as identifiers.
- Syntax
 

```
enum eName { ID1, ID2, ID3, ID4, ID5 };
```
- Example:
 

```
enum ePrimeColors {RED, BLUE, YELLOW};
enum eVowels {A, E, I, O, U}
```
- Error:
 

```
enum eRGBColors {1RED, 2GREEN, 3BLUE}
enum eVowels {'A', 'E', 'I', 'O', 'U'}
```

Copyright © 2001 R.M. Laurie 12

### Storage of **enum** Type Variables

```
enum MonthType { JAN, FEB, MAR, APR, MAY, JUN,
                JUL, AUG, SEP, OCT, NOV, DEC };
```

Diagram illustrating the storage of enum values:
 

- JAN is stored as 0
- FEB is stored as 1
- MAR is stored as 2
- APR is stored as 3
- etc.
- DEC is stored as 11

Copyright © 2001 R.M. Laurie | 13

### Use Type Cast to Increment **enum** Type Variables

```
enum MonthType { JAN, FEB, MAR, APR, MAY, JUN,
                JUL, AUG, SEP, OCT, NOV, DEC };
```

```
MonthType thisMonth;
MonthType lastMonth;

lastMonth = OCT;
thisMonth = NOV;
lastMonth = thisMonth;
thisMonth = thisMonth++; // COMPILE ERROR !
thisMonth = MonthType( thisMonth + 1 ); // uses type cast
```

Copyright © 2001 R.M. Laurie | 14

### typedef Statement

- User defined data type using typedef statement creates Synonym
- Syntax  
typedef ExistingDataType NewDataType;
- Example:

```
typedef int Boolean;
const Boolean TRUE=1;
const Boolean FALSE=0;

Boolean bAnswer;

bAnswer = FALSE;
```

Copyright © 2001 R.M. Laurie | 15

### Additional Operators

- Combined Assignment  
+= -= \*= /= %=
- Bitwise
  - << >> Shift Bits
  - & AND Bits
  - | OR Bits
  - ^ Exclusive OR Bits
  - ~ Complement Bits
- Conditional  
?:

Copyright © 2001 R.M. Laurie | 16

### Combined Assignment Example

```
A += 2;   A=A+2;
B -= 1;   B=B-1
C *= 4;   C=C*4
D /= 2;   D=D/2
E %= 5;   E=E%5
```

Copyright © 2001 R.M. Laurie | 17

### Bitwise Operators

- Four primary logic operations

AND &	A	B	A&B
	0	0	0
	0	1	0
	1	0	0
	1	1	1

OR	A	B	A B
	0	0	0
	0	1	1
	1	0	1
	1	1	1

Complement ~	A	~A
	0	1
	1	0

Exclusive OR ^	A	B	A^B
	0	0	0
	0	1	1
	1	0	1
	1	1	0

Copyright © 2001 R.M. Laurie | 18

### Bitwise Example

```

#include <iostream>
using namespace std;
int main(void)
{
    int nA=6, nB=3;

    cout << (nA | nB) << endl;
    cout << (nA & nB) << endl;
    cout << (nA ^ nB) << endl;
    cout << ~nB << endl;

    cout << (nB <= 2) << endl;
    cout << (nB <= 2) << endl;
    cout << (nB >= 4) << endl;
    return 0;
}
    
```

7
2
5
-4
12
48
3

Copyright © 2001 R.M. Laurie 19

### Conditional Operator

**ExpressionCondition ? ExpressionTrue : ExpressionFalse**

**Examples on p. 515**  
 Best to Avoid using this Operator  
 Use if else statements instead

Copyright © 2001 R.M. Laurie 20

### Casting Numerical Data

- (int) cast operator converts floating point value to integer
- (float) cast operator converts integer value to floating point
- (unsign long) fDistance
 

```

int intVar;
float floatVar = 104.8 ;
intVar = int ( floatVar ) ; // functional notation, OR
intVar = ( int ) floatVar ; // prefix notation uses ( )
            
```

Figure 2.5: Floating Point Number Representation

31	30	24	23	22	0
Sign	Exponent	Sign	Mantissa		

Copyright © 2001 R.M. Laurie 21

### C++ Operator Precedence

(highest to lowest)

<u>Operator</u>	<u>Associativity</u>
( )	Left to right
unary: ++ -- ! + - (cast) sizeof	Right to left
* / %	Left to right
+ -	Left to right
< <= > >=	Left to right
== !=	Left to right
&&	Left to right
	Left to right
? :	Right to left
= += -= *= /= %=	Right to left

Copyright © 2001 R.M. Laurie 22