Chapter 3
Basic Data Types
Topics

- Scalar Types in C
- Integers
- Bit Operations
- Floating Point Types
- Conversions
Scalar Types in C

- The amount of memory available for a variable depends on its type.
- Each type has a specific size in the memory of a computer.
- Each type has a specific format in the memory of a computer.
- A Scalar Type contains a single memory value.
Computer Memory

- Computers store two things in memory:
  - The executable code and
  - the data values
- A *bit* is a memory area that contains either a single value of 0 or a single value of 1.
- A *byte* consists of 8 bits of memory.
- A *word* is a set of one or more bytes
Integers and Integer Bases

- An integer (\textit{int}) is a whole number, ranging from the negative to positive range.
- ..., \(-n, -n+1, -n+2, \ldots, -2, -1, 0, 1, \ldots, n-1, n-2, n, \ldots\)
- An integer is stored as a bit pattern in binary notation.
Integers and Integer Bases

- Base notation is how a number is represented when recorded.
- The decimal system uses a base ten notation.
Example of Base 10 (Decimal)

One-hundred and twenty-six, 126, is a representation of a base 10 number.
The right most column represents $10^0$, or 1
The middle column represents $10^1$, or 10
The leftmost column represents $10^2$, or 100
So, $1 \times 100 + 2 \times 10 + 6 \times 1$ equals 126.
In computers, integers tend to be represented in three other bases.

- **Binary** represents the base 2 notation
- **Octal** represents the base 8 notation
- **Hexadecimal** represents base 16 notation
Integers and Integer Bases

• For Binary, each column represents a power of 2
  • The rightmost column is $2^0$, or 1
  • The next column is $2^1$, or 2
  • The next column is $2^2$, or 4
  • The next column is $2^3$ or 8
  • Each further column is $2^{(n+1)}$

• What is 0101 binary in decimal?
Integers and Integer Bases

• For Octal, each column represents a power of 8
  • The rightmost column is $8^0$, or 1
  • The next column is $8^1$, or 8
  • The next column is $8^2$, or 64
  • The next column is $8^3$ or 512
  • Each further column is $8^{(n+1)}$

• What is 1010 octal in decimal?
Integers and Integer Bases

- For Hexadecimal, each column represents a power of 16
  - The rightmost column is $16^0$, or 1
  - The next column is $16^1$, or 16
  - The next column is $16^2$, or 256
  - The next column is $16^3$ or 4096
  - Each further column is $16^{(n+1)}$

- What is 1101 hexadecimal in decimal?
# Integer and Integer Bases

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
<th>Octal</th>
<th>Hexadecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>1010</td>
<td>12</td>
<td>A</td>
</tr>
<tr>
<td>11</td>
<td>1011</td>
<td>13</td>
<td>B</td>
</tr>
<tr>
<td>12</td>
<td>1100</td>
<td>14</td>
<td>C</td>
</tr>
<tr>
<td>13</td>
<td>1101</td>
<td>15</td>
<td>D</td>
</tr>
<tr>
<td>14</td>
<td>1110</td>
<td>16</td>
<td>E</td>
</tr>
<tr>
<td>15</td>
<td>1111</td>
<td>17</td>
<td>F</td>
</tr>
<tr>
<td>16</td>
<td>10000</td>
<td>20</td>
<td>10</td>
</tr>
</tbody>
</table>
The Integer Types

- An integer is usually of the size of one word on a computer.
- The size of word changes from machine to machine.
- A 16 bit machine has a range from -32768 to +32767
- A 32 bit machine has a range from -2147483648 to +2147483647
Integer Types

• Three types of integers
  • a regular integer usually takes up a word of space.
  • a short integer, declared `short` or `short int` takes up less memory space
  • a long integer, declared `long` or `long int` takes up more memory space
  • Unsigned integer, declared `unsigned int`
Typically, an `int` in a system is already the type of `long` or `short`.

- In Visual Studio, a short is a 2-byte word.
- In Visual Studio, a long is a 8-byte word.
- Which of these is type `int` in Visual Studio?
Input and Output of Integer Types

For scanf and printf, the following format strings can be used for integer types:

- long in decimal %ld
- long in octal %lo
- long in hexadecimal %lx or %lX
- short in decimal %hd
- short in octal %ho
- short in hexadecimal %hx or %hX
The Sizeof Operator

- The `sizeof` operator gives the number of bytes associated with a type or a variable.

```c
int x;
mysize = sizeof(x);
mysize = sizeof(long int);
  - mysize would be the size of an integer
```

- `mysize = sizeof(intvar+96);`
  - uses the memory space of an integer, it still has the size of an integer.
A Closer Look at Integers

- The size of the integer will differ from machine to machine.
- The method of storing an integer will be consistent from machine to machine.
Binary Representation of an Integer

- For the following, the size of an integer will be assumed to be 16 bits.
- The value of the integer will be represented in the lower 15 bits of the binary number.
- The first bit of the integer will represent the sign bit.
Excel Spreadsheet Example

• An integer uses the highest bit of its word to represent the sign of the value.
• A 16-bit integer will have a range of \(-2^{15}\) to positive \(2^{15}-1\), which is \(-32768\) to \(32767\).
• When the sign bit becomes active, with no subsequent bits, the value is the maximum negative number on this scale.
Integer Representation
Character Variables

- There are 128 characters in ANSI C from 0-127 inclusive.
- Each of these characters can be treated as an integer.
- Each character can be represented by a single byte of 8 bits.
Character Variables

char Alpha;
Alpha= ‘A’;
printf(“Alpha is \" %c \" \n”,Alpha);
printf(“Alpha is %d \n”,Alpha);

printf(“Next letter is \’%c\’\n”,Alpha+1);
Results in :
    Next letter is ‘B’
Unsigned Types

- Unsigned variables increase the range of the type by using the sign bit to store data.
- A 16 bit integer has a range of -32768 to 32767.
- A 16 bit unsigned integer has a range of 0 to 65535
Bitwise Operations on an Integer

• ~ bitwise-negation
• & bitwise-and
• ^ bitwise-xor
• | bitwise-or
• >> shift right
• << shift left
Binary Operators

- Bitwise Negation "~"
  ```c
  int value = 242;
  printf("Value is %d\n", value);
  printf("~Value is %d\n", ~value);
  ```
  ```
  Value is 242
  ~Value is -243
  ```
Bitwise Operators

• Bitwise And “&”
  ```c
  int FirstValue=242;
  int SecondValue=115;
  ```

• FirstValue in Binary is:
  0000000011110010

• SecondValue in Binary is:
  0000000001110011

• Bitwise AND compares each column of the binary form. If both are 1, then the result is 1. If not, the result is 0

• Application:
  
  0000000011110010
  0000000001110011
  
  0000000001110010

• The resulting value as an integer becomes 114
Bitwise Operators

- Bitwise Xor “^”
- XOR returns TRUE if the inputs are different.
- 0^0 is 0
- 0^1 is 1
- 1^0 is 1
- 1^1 IS 0
Bitwise Operators

- Bitwise XOR “^”
  ```
  int FirstValue = 242;
  int SecondValue = 115;
  ```
- FirstValue in Binary is:
  ```
  0000000011110010
  ```
- SecondValue in Binary is:
  ```
  0000000001110011
  ```
- Bitwise XOR compares each column of the binary form. If both are the same, then the result is 0. If not, the result is 1.
- Application:
  ```
  0000000011110010
  0000000001110011
  ------------------
  0000000010000001
  ```
- The resulting value as an integer becomes 129
Bitwise Operators

- Bitwise OR “|”
  int FirstValue=242;
  int SecondValue=115;
- FirstValue in Binary is:
  0000000011110010
- SecondValue in Binary is:
  0000000001110011
- Bitwise OR compares each column of the binary form. If either one is 1, then the result is 1. If not, the result is 0
- Application:
  0000000011110010
  0000000001110011
  0000000011110011
  0000000011110011
- The resulting value as an integer becomes 243
Bitwise Operators

- `>>` right shift
- `<<` left shift
- Moves the binary value left or right by the given value.
Bitwise Operators

int FirstValue=144;
int ValueLeftShift=0;
int ValueRightShift=0;

• FirstValue in Binary is:
  0000000010010000

• Shift two positions to the right
  ValueRightShift=FirstValue>>2;

• The binary result is:
  0000000000100100

• The integer value is 36
  ValueLeftShift=FirstValue<<2;

• The binary result is:
  0000000100100000

• The integer value is 576
Floating Point Types

- Floating point numbers are more like regular arithmetic
- 9.81, 32.2 are examples of floating point numbers
- Floating point numbers are another type of variable.
Floating Point Types

• Declaration of Variables
  float Speed;
  double Acceleration=0.0;
  long double Pmantis=0.0;

• Assignment of Variables
  Speed=44.0f; Speed = 44;
  Acceleration=7.0;
  Pmantis=6.665L;
Floating Point Types

• Many, but not all, of the operators are the same for integers and floats
  • +, -, *, /
  • ==, !=, <, <=, >, >=
  • !, &&, ||

• But **NOT** the remainder function
  • %
Input and Output

- There are three floating point types
  - `float`
  - `double`
  - `long double`

- Notations
  - `'%f'` is for a decimal notation
  - `'%e'` is for an exponential notation
  - `'%g'` is for either decimal or exponential notation

- Format Strings for Floating Point Types
  - `"%f","%e","%g"` for a float
  - `"%Lf","%le","%Lg"` for a double
  - `"%Lf","%Le","%Lg"`, for a long double
Types of Expressions and Automatic Conversions

- The result of any logical operator is an int value.
- Most conversions in C are automatic, from highest to least priority:
  - long double
  - double
  - float
  - unsigned long
  - long
  - unsigned int
  - int
Automatic Conversions

• Exceptions
  • Any expression involving char, signed, unsigned char, short, or unsigned short, the dominating type will be either int or unsigned int
  • Expressions of long int and unsigned int, the dominating type could be either long int or unsigned long int.
Automatic Type Conversions

- Examples
  - $2 + 3.113$ becomes a double of $5.113$
  - ‘a’+1 becomes an integer 98, which can be formatted to the character ‘b’
Forced Type Conversions

• Many times an automatic conversion does not give the desired result.
• A **type cast** can be used to explicitly convert a value or the result of an expression to a known type.
• *(type)* expression is the usual format for the routine.
Force Type Conversions

- (float) 1 becomes 1.000000
- y = ((float) 'a') + 1 becomes 98.000000
- (int) 'B' becomes 66
- y = (int) 2.4 becomes 2