## Data Types

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
<th>Bytes in Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>Character</td>
<td>1</td>
</tr>
<tr>
<td>int</td>
<td>Whole number</td>
<td>4 or 2 (natural size of integer in host machine)</td>
</tr>
<tr>
<td>float</td>
<td>Real number - Single precision floating point</td>
<td>Usually 4</td>
</tr>
<tr>
<td>double</td>
<td>Real number - Double precision floating point</td>
<td>Usually 8</td>
</tr>
<tr>
<td>short</td>
<td>Shorter than regular</td>
<td>Usually 2</td>
</tr>
<tr>
<td>long</td>
<td>Longer than regular</td>
<td>Usually 8</td>
</tr>
<tr>
<td>unsigned</td>
<td>No bits used for sign</td>
<td></td>
</tr>
<tr>
<td>signed</td>
<td>1 bit used for sign</td>
<td></td>
</tr>
</tbody>
</table>
Numeric Data Types

- **Integers**
  - short
  - int
  - long

- **Floating-point Values**
  - float
  - double
  - long
  - double
Data type: char

- 1 Byte or 8 bits
- Example: A, c, x, q
- Character is represented in memory as a binary number
- Value stored is determined by ASCII (American Standard Code for Information Interchange) code.
- Print format: %c

<table>
<thead>
<tr>
<th>Character</th>
<th>ASCII Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>' '</td>
<td>32</td>
</tr>
<tr>
<td>' '</td>
<td>42</td>
</tr>
<tr>
<td>'A'</td>
<td>65</td>
</tr>
<tr>
<td>'B'</td>
<td>66</td>
</tr>
<tr>
<td>'C'</td>
<td>67</td>
</tr>
<tr>
<td>'D'</td>
<td>68</td>
</tr>
<tr>
<td>'E'</td>
<td>69</td>
</tr>
<tr>
<td>'F'</td>
<td>70</td>
</tr>
<tr>
<td>'G'</td>
<td>71</td>
</tr>
<tr>
<td>'H'</td>
<td>72</td>
</tr>
<tr>
<td>'I'</td>
<td>73</td>
</tr>
<tr>
<td>'J'</td>
<td>74</td>
</tr>
<tr>
<td>'K'</td>
<td>75</td>
</tr>
<tr>
<td>'L'</td>
<td>76</td>
</tr>
<tr>
<td>'M'</td>
<td>77</td>
</tr>
<tr>
<td>'N'</td>
<td>78</td>
</tr>
<tr>
<td>'O'</td>
<td>79</td>
</tr>
<tr>
<td>'P'</td>
<td>80</td>
</tr>
<tr>
<td>'Q'</td>
<td>81</td>
</tr>
<tr>
<td>'R'</td>
<td>82</td>
</tr>
<tr>
<td>'S'</td>
<td>83</td>
</tr>
<tr>
<td>'T'</td>
<td>84</td>
</tr>
<tr>
<td>'U'</td>
<td>85</td>
</tr>
<tr>
<td>'V'</td>
<td>86</td>
</tr>
<tr>
<td>'W'</td>
<td>87</td>
</tr>
<tr>
<td>'X'</td>
<td>88</td>
</tr>
<tr>
<td>'Y'</td>
<td>89</td>
</tr>
<tr>
<td>'Z'</td>
<td>90</td>
</tr>
<tr>
<td>'a'</td>
<td>97</td>
</tr>
<tr>
<td>'b'</td>
<td>98</td>
</tr>
<tr>
<td>'c'</td>
<td>99</td>
</tr>
<tr>
<td>'d'</td>
<td>100</td>
</tr>
<tr>
<td>'e'</td>
<td>101</td>
</tr>
<tr>
<td>'f'</td>
<td>102</td>
</tr>
<tr>
<td>'g'</td>
<td>103</td>
</tr>
<tr>
<td>'h'</td>
<td>104</td>
</tr>
<tr>
<td>'i'</td>
<td>105</td>
</tr>
<tr>
<td>'j'</td>
<td>106</td>
</tr>
<tr>
<td>'k'</td>
<td>107</td>
</tr>
<tr>
<td>'l'</td>
<td>108</td>
</tr>
<tr>
<td>'m'</td>
<td>109</td>
</tr>
<tr>
<td>'n'</td>
<td>110</td>
</tr>
<tr>
<td>'o'</td>
<td>111</td>
</tr>
<tr>
<td>'p'</td>
<td>112</td>
</tr>
<tr>
<td>'q'</td>
<td>113</td>
</tr>
<tr>
<td>'r'</td>
<td>114</td>
</tr>
<tr>
<td>'s'</td>
<td>115</td>
</tr>
<tr>
<td>'t'</td>
<td>116</td>
</tr>
<tr>
<td>'u'</td>
<td>117</td>
</tr>
<tr>
<td>'v'</td>
<td>118</td>
</tr>
<tr>
<td>'w'</td>
<td>119</td>
</tr>
<tr>
<td>'x'</td>
<td>120</td>
</tr>
<tr>
<td>'y'</td>
<td>121</td>
</tr>
<tr>
<td>'z'</td>
<td>122</td>
</tr>
<tr>
<td>'0'</td>
<td>48</td>
</tr>
<tr>
<td>'1'</td>
<td>49</td>
</tr>
<tr>
<td>'2'</td>
<td>50</td>
</tr>
<tr>
<td>'3'</td>
<td>51</td>
</tr>
<tr>
<td>'4'</td>
<td>52</td>
</tr>
<tr>
<td>'5'</td>
<td>53</td>
</tr>
<tr>
<td>'6'</td>
<td>54</td>
</tr>
<tr>
<td>'7'</td>
<td>55</td>
</tr>
<tr>
<td>'8'</td>
<td>56</td>
</tr>
<tr>
<td>'9'</td>
<td>57</td>
</tr>
</tbody>
</table>
Data type: int

- Standard Integer
- Limited by size of memory
- Usually 4 bytes
- Value stored in binary
- 1 bit for sign (0 for positive, 1 for negative)
- Range: -2147483648, 2147483647
- Print format: %d
- Use unsigned to use all the bits
Integer will not suffice – real applications

- Calculate area of a circle
- Calculate average of grades in class

42.686908, -73.823919
Float, Double

- Real number, analogous to scientific notation
- Storage area divided into three areas:
  - Sign (0 for positive, 1 for negative)
  - Exponent (repeated multiplication)
  - Mantissa (binary fraction between 0.5 and 1)

The mantissa and exponent are chosen such that the following formula is correct

\[ real\ number = mantissa \times 2^{exponent} \]
Float, Double

- **Float (single precision)**
  - 1 bit sign, 8 bits exponent, 23 bits mantissa

- **Double (double precision)**
  - 1 bit sign, 11 bits exponent, 52 bits mantissa

- Depends on hardware

- Print format: %f (for float) %lf (for double)
Short, Long, Long Double

- **Short**
  - Usually 2 bytes whole number
  - Print format: %d

- **Long**
  - Usually 8 bytes whole number
  - Print format: %ld

- **Long Double**
  - Usually 16 bytes fractional
  - Print format: %Lf
#include <stdio.h>
#include <float.h>
#include <limits.h>

int main(void)
{
    char myChar;
    printf("Size of Char = %ld\n", sizeof(myChar));
    int myInt;
    printf("Size of Int = %ld\n", sizeof(myInt));
    short myShortInt;
    printf("Size of Short = %ld\n", sizeof(myShortInt));
    long myLongInt;
    printf("Size of Long = %ld\n", sizeof(myLongInt));
    float myFloat;
    printf("Size of Float = %ld\n", sizeof(myFloat));
    double myDouble;
    printf("Size of Double = %ld\n", sizeof(myDouble));

    long double myLongDouble;
    printf("Size of Long Double = %ld\n", sizeof(myLongDouble));

    printf("INT_MAX = %d\n", INT_MAX);
    printf("SHORT_MAX = %d\n", SHRT_MAX);
    printf("LONG_MAX = %ld\n", LONG_MAX);
    printf("MAX FLOAT = %f\n", FLT_MAX);
    printf("MAX DOUBLE = %f\n", DBL_MAX);
}

Output of size

Size of Char = 1
Size of Int = 4
Size of Short = 2
Size of Long = 8
Size of Float = 4
Size of Double = 8
Size of Long Double = 16
INT MAX = 2147483647
SHORT MAX = 32767
LONG MAX = 9223372036854775807
MAX FLOAT = 340282346638528859811704183484516925440.000000
MAX DOUBLE = 1797693134862315708145274237317043567980705675258
48274797826204144723168738177180919299881250404026184124858368
## Ranges

### Whole Number

<table>
<thead>
<tr>
<th>Type</th>
<th>Range in Typical Microprocessor Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>short</td>
<td>−32,767 .. 32,767</td>
</tr>
<tr>
<td>unsigned short</td>
<td>0 .. 65,535</td>
</tr>
<tr>
<td>int</td>
<td>−2,147,483,647 .. 2,147,483,647</td>
</tr>
<tr>
<td>unsigned</td>
<td>0 .. 4,294,967,295</td>
</tr>
<tr>
<td>long</td>
<td>−2,147,483,647 .. 2,147,483,647</td>
</tr>
<tr>
<td>unsigned long</td>
<td>0 .. 4,294,967,295</td>
</tr>
</tbody>
</table>

### Real Number

<table>
<thead>
<tr>
<th>Type</th>
<th>Approximate Range*</th>
<th>Significant Digits*</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>10^{-37} .. 10^{38}</td>
<td>6</td>
</tr>
<tr>
<td>double</td>
<td>10^{-307} .. 10^{308}</td>
<td>15</td>
</tr>
<tr>
<td>long double</td>
<td>10^{-4931} .. 10^{4932}</td>
<td>19</td>
</tr>
</tbody>
</table>

*In a typical microprocessor-based C implementation*
Class Assignment

- Write a program to convert temperature in Fahrenheit to Celsius according to the following formula. Take user’s input for the temperature in Fahrenheit.

\[ C = \frac{5 \times (F - 32)}{9} \]
Write a program that has a constant for PI (3.14159) and variables radius, area, and circumference as double. Take radius as input from the user and calculate circumference and area according to the formula below.

\[
\text{Circumference} = 2\pi r \\
\text{Area} = \pi r^2
\]
Review Questions

- State True or False:
  - Short takes more memory space than Integer (int)
  - Float and double are real number representations in C
  - Char is represented in memory by ASCII
  - Print format for char is %d
  - Print format for double is %f
  - Float and double has 2 parts: exponent and mantissa
Review Questions / Answers

- State True or False:
  - Short takes more memory space than Integer (int)  
    - Answer: FALSE
  - Float and double are real number representations in C  
    - Answer: TRUE
  - Char is represented in memory by ASCII  
    - Answer: TRUE
  - Print format for char is %d  
    - Answer: FALSE
  - Print format for double is %f  
    - Answer: TRUE
  - Float and double has 2 parts: exponent and mantissa  
    - Answer: FALSE
What is the error in code?

```c
#include <stdio.h>

int main ( void )
{
    printf("Hello World");
}
```
What is the error in code?

Compilation Error

/home/ubuntu/workspace/code_slides/compError.c:5:4: error: expected declaration specifiers or ‘...’ before ‘printf’
printf("Hello World");

/home/ubuntu/workspace/code_slides/compError.c:3:5: error: ‘main’ declared as function returning a function
int main ( void )
  ^

/home/ubuntu/workspace/code_slides/compError.c: In function ‘main’:
/home/ubuntu/workspace/code_slides/compError.c:6:1: error: expected ‘{’ at end of input
}  ^

Correct Code

```
#include <stdio.h>
int main ( void )
{
    printf("Hello World");
}
```
What is the error in code?

```c
#include <stdio.h>

int main ( void )
{
    printf("Hello World")
}
```
What is the error in code?

Compilation Error

```
/home/ubuntu/workspace/code_slides/compError.c:10:10: error: expected ';' before '}' token
```

Correct Code

```
#include <stdio.h>

int main ( void )
{
    printf("Hello World");
}
```
What is the error in code?

```c
#include <stdio.h>

int main ( void )
{
    printf("Hello World\n");
}
```
What is the error in code?

```c
#include <stdio.h>

int main ( void )
{
    printf("Hello World");
}
```

Compilation Error

```
/home/ubuntu/workspace/code_slides/compError.c: In function ‘main’:
/home/ubuntu/workspace/code_slides/compError.c:5:11: warning: missing terminating " character [enabled by default]
    printf("Hello World");

/home/ubuntu/workspace/code_slides/compError.c:5:4: error: missing terminating " character
    printf("Hello World");

/home/ubuntu/workspace/code_slides/compError.c:6:1: error: expected expression before ‘}’ token
}

/home/ubuntu/workspace/code_slides/compError.c:6:1: error: expected ‘;’ before ‘}’ token
```

Correct Code

```c
#include <stdio.h>

int main ( void )
{
    printf("Hello World");
}
```
Common Errors

- Omitting the parentheses after main.
- Omitting or incorrectly typing the opening brace { that signifies the start of a function body.
- Omitting or incorrectly typing the closing brace } that signifies the end of a function.
- Misspelling the name of a function; for example, typing pintf ( ) instead of printf ( ).
- Forgetting to close the message to printf ( ) with a double quote symbol.
- Omitting the semicolon at the end of each C statement.
- Adding a semicolon at the end of the #inc 1ude preprocessor command.
- Forgetting the \n to indicate a new line.
- Incorrectly typing the letter 0 for the number zero (0), or vice versa.
- *Incorrectly typing the letter l for the number 1, or vice versa.*
C Keywords

- Reserved words of the language, special meaning to C compiler
- Do not use these as identifiers, like variable names

<table>
<thead>
<tr>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto</td>
</tr>
<tr>
<td>break</td>
</tr>
<tr>
<td>case</td>
</tr>
<tr>
<td>char</td>
</tr>
<tr>
<td>const</td>
</tr>
<tr>
<td>continue</td>
</tr>
<tr>
<td>default</td>
</tr>
</tbody>
</table>

*Keywords added in C99 standard*

_Bool _Complex _Imaginary inline restrict

*Keywords added in C11 standard*

_Alignas _Alignof _Atomic _Generic _Noreturn _Static_assert _Thread_local*
Decision Making - Example

- **Check condition**
  - Is the distance between Albany to NYC more than Albany to Buffalo?
  - Is John’s grade greater than 60?

- **Perform Tasks based on decision**
  - If Albany to NYC is shorter, then I will drive to NYC
  - If Amy’s grade is greater than 60, then she passes

- **Otherwise**
  - I will drive to Buffalo
  - She fails
Decision Making

- Executable Statements in C
  - Perform actions
  - Makes decisions (based on condition)

- **if statement** allows a program to make a decision based on the truth or falsity of a statement of fact called a condition.
#include <stdio.h>

int main ( void )
{
    int integer1 = 5;
    int integer2 = 10;
    if (integer1 > integer2)
    {
        printf("This statement is not printed if the condition is False\n");
    }
    printf("This statement is always executed as it is outside the if statement\n");
}
If Statement

- If the condition is **true** (i.e., the condition is met) the statement in the body of the `if` statement is executed.
- If the condition is **false** (i.e., the condition isn’t met) the body statement is not executed.
- Whether the body statement is executed or not, after the `if` statement completes, execution proceeds with the next statement after the `if` statement.
- Conditions in `if` statements are formed by using the equality operators and relational operators.
# Relational & Equality Operators

<table>
<thead>
<tr>
<th>Algebraic equality or relational operator</th>
<th>C equality or relational operator</th>
<th>Example of C condition</th>
<th>Meaning of C condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relational operators</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&gt;</td>
<td>&gt;</td>
<td>x &gt; y</td>
<td>x is greater than y</td>
</tr>
<tr>
<td>&lt;</td>
<td>&lt;</td>
<td>x &lt; y</td>
<td>x is less than y</td>
</tr>
<tr>
<td>&gt;=</td>
<td>&gt;=</td>
<td>x &gt;= y</td>
<td>x is greater than or equal to y</td>
</tr>
<tr>
<td>&lt;=</td>
<td>&lt;=</td>
<td>x &lt;= y</td>
<td>x is less than or equal to y</td>
</tr>
<tr>
<td>Equality operators</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>=</td>
<td>==</td>
<td>x == y</td>
<td>x is equal to y</td>
</tr>
<tr>
<td>≠</td>
<td>!=</td>
<td>x != y</td>
<td>x is not equal to y</td>
</tr>
</tbody>
</table>
## Precedence of Operators

<table>
<thead>
<tr>
<th>Operators</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>()</td>
<td>left to right</td>
</tr>
<tr>
<td>* / %</td>
<td>left to right</td>
</tr>
<tr>
<td>+ -</td>
<td>left to right</td>
</tr>
<tr>
<td>&lt; &lt;= &gt; &gt;=</td>
<td>left to right</td>
</tr>
<tr>
<td>== !=</td>
<td>left to right</td>
</tr>
<tr>
<td>=</td>
<td>right to left</td>
</tr>
</tbody>
</table>
Example C Program

```c
// Fig. 2.13: fig02_13.c
// Using if statements, relational operators, and equality operators.
#include <stdio.h>

// function main begins program execution
int main( void )
{
    printf( "Enter two integers, and I will tell you\n" );
    printf( "the relationships they satisfy: " );

    int num1; // first number to be read from user
    int num2; // second number to be read from user

    scanf( "%d %d", &num1, &num2 ); // read two integers

    if ( num1 == num2 ) {
        printf( "%d is equal to %d\n", num1, num2 );
    } // end if
```
Example C Program... continued

```c
21 if ( num1 != num2 ) {
22     printf( "%d is not equal to %d\n", num1, num2 );
23 } // end if
24
25 if ( num1 < num2 ) {
26     printf( "%d is less than %d\n", num1, num2 );
27 } // end if
28
29 if ( num1 > num2 ) {
30     printf( "%d is greater than %d\n", num1, num2 );
31 } // end if
32
33 if ( num1 <= num2 ) {
34     printf( "%d is less than or equal to %d\n", num1, num2 );
35 } // end if
36
37 if ( num1 >= num2 ) {
38     printf( "%d is greater than or equal to %d\n", num1, num2 );
39 } // end if
40 } // end function main
```
Example C Program .... Output

Enter two integers, and I will tell you
the relationships they satisfy: 3 7
3 is not equal to 7
3 is less than 7
3 is less than or equal to 7

Enter two integers, and I will tell you
the relationships they satisfy: 22 12
22 is not equal to 12
22 is greater than 12
22 is greater than or equal to 12

Enter two integers, and I will tell you
the relationships they satisfy: 7 7
7 is equal to 7
7 is less than or equal to 7
7 is greater than or equal to 7
Classroom Assignment

- Write a program that asks the user to enter two integers, obtains the numbers from the user, then prints the larger number followed by the words “is larger.”