C Programming for Engineers

Data Types, Decision Making

UNIVERSITY AT ALBANY
State University of New York

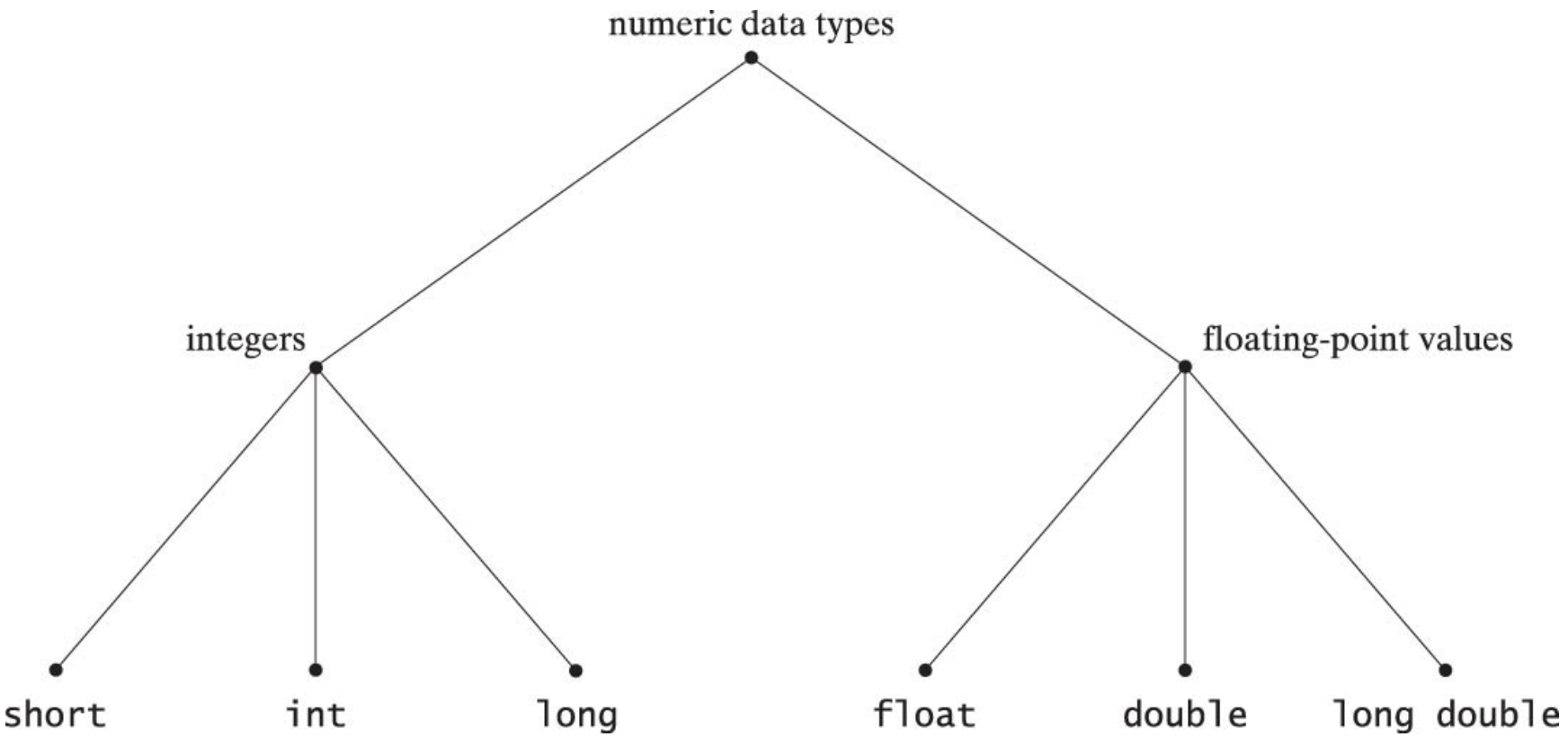
ICEN 360– Spring 2017

Prof. Dola Saha

Data Types

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

Numeric Data Types



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

Character	ASCII Code
' '	32
'*'	42
'A'	65
'B'	66
'Z'	90
'a'	97
'b'	98
'z'	122
'0'	48
'9'	57

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

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)

type `double` format

- 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

Size and limits

```c
#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 = 179769313486231570814527423731704356798070567525
48274797826204144723168738177180919299881250404026184124858368
```

Ranges

Whole Number

Type	Range in Typical Microprocessor Implementation
`short`	–32,767 .. 32,767
`unsigned short`	0 .. 65,535
`int`	–2,147,483,647 .. 2,147,483,647
`unsigned`	0 .. 4,294,967,295
`long`	–2,147,483,647 .. 2,147,483,647
`unsigned long`	0 .. 4,294,967,295

Real Number

Type	Approximate Range*	Significant Digits*
`float`	10^{-37} .. 10^{38}	6
`double`	10^{-307} .. 10^{308}	15
`long double`	10^{-4931} .. 10^{4932}	19

*In a typical microprocessor-based C implementation

Class Assignment

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

$$C = \frac{5 \times (F - 32)}{9}$$

Class Assignment

- 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.

$$Circumference = 2\pi r$$
$$Area = \pi r^2$$

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) — FALSE
  - Float and double are real number representations in C — TRUE
  - Char is represented in memory by ASCII — TRUE
  - Print format for char is %d — FALSE
  - Print format for double is %f — TRUE
  - Float and double has 2 parts: exponent and mantissa — FALSE

What is the error in code?

```
1  #include <stdio.h>
2
3  int main ( void )
4  (
5     printf("Hello World");
6  )
```

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: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

```c
#include <stdio.h>

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

What is the error in code?

```
1  #include <stdio.h>
2
3  int main ( void )
4  {
5     printf("Hello World")
6  }
```

What is the error in code?

```
1  #include <stdio.h>
2
3  int main ( void )
4  {
5     printf("Hello World")
6  }
```

Compilation Error

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

Correct Code

```
1  #include <stdio.h>
2
3  int main ( void )
4  {
5     printf("Hello World");
6  }
```

What is the error in code?

```
1  #include <stdio.h>
2
3  int main ( void )
4  {
5     printf("Hello World);
6  }
```

What is the error in code?

```
1  #include <stdio.h>
2
3  int main ( void )
4  {
5     printf("Hello World);
6  }
```

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

```
1  #include <stdio.h>
2
3  int main ( void )
4  {
5     printf("Hello World");
6  }
```

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

Keywords

auto	do	goto	signed	unsigned
break	double	if	sizeof	void
case	else	int	static	volatile
char	enum	long	struct	while
const	extern	register	switch	
continue	float	return	typedef	
default	for	short	union	

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.

If Statement

```c
#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

Algebraic equality or relational operator	C equality or relational operator	Example of C condition	Meaning of C condition
Relational operators			
>	>	x > y	x is greater than y
<	<	x < y	x is less than y
≥	>=	x >= y	x is greater than or equal to y
≤	<=	x <= y	x is less than or equal to y
Equality operators			
=	==	x == y	x is equal to y
≠	!=	x != y	x is not equal to y

Precedence of Operators

Operators	Associativity
()	left to right
* / %	left to right
+ -	left to right
< <= > >=	left to right
== !=	left to right
=	right to left

Example C Program

```
1  // Fig. 2.13: fig02_13.c
2  // Using if statements, relational
3  // operators, and equality operators.
4  #include <stdio.h>
5
6  // function main begins program execution
7  int main( void )
8  {
9     printf( "Enter two integers, and I will tell you\n" );
10    printf( "the relationships they satisfy: " );
11
12    int num1; // first number to be read from user
13    int num2; // second number to be read from user
14
15    scanf( "%d %d", &num1, &num2 ); // read two integers
16
17    if ( num1 == num2 ) {
18       printf( "%d is equal to %d\n", num1, num2 );
19    } // end if
20
```

Example C Program… continued

```
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.”