

LECTURE 2:

DATA TYPES (REVISITED)

COMP1002J: Introduction to Programming 2

Dr. Brett Becker (brett.becker@ucd.ie)

Beijing Dublin International College

Reminder!

- Labs start next week (week 3)!

Data Types

- **Data types** describe the kind of data that can be:
 - Stored in a variable
 - Passed as a parameter to a function
 - Returned by a function
 - Etc.
- **C has 4 primitive (basic) data types:**
 - `char` – represents characters (or small numbers)
 - `int` – represents integer (whole) numbers
 - `float` – represents real (floating-point) numbers
 - `double` – double-precision floating point
- **Some questions:**
 - What is the biggest (positive or negative) integer number?
 - How are these numbers chosen?

Data Types

- Data is stored in the computer's memory.
 - Memory consists of bytes.
 - 1GB = 1,000,000,000 bytes
 - Each byte can hold an 8-bit number.
 - The biggest integer value that can be stored in a byte is 255, or 127 (stored as a signed integer)
 - In a *signed* value, one bit is used to record the sign (whether the number is positive or negative).
 - For *unsigned* values, all values are ≥ 0 .
- But you have written programs that use integer values like 1000, maybe even bigger... how does that work?

Integer Data Types

- Integer numbers are represented by combining more than one byte.
 - For example, two bytes can be used to store integer values from 0 up to $2^{16}-1$ (65,535) – unsigned.
 - Two bytes can also be used for the range -2^{15} (-32,768) to $2^{15}-1$ (32,767) – signed
- **But:** the number of bytes used for an `int` value is not always the same!
 - It can change for different systems, different compilers, etc.

Integer Data Types

- In C, it is common for the `int` type to use 4 bytes (32 bits) to store integer values, with one bit used for the sign.
 - The maximum signed 4-byte integer is: $2^{31}-1$ (= 2,147,483,647)
 - The minimum signed 4-byte integer is: -2^{31} (= -2,147,483,648)

Integer Data Types

- We can check how many bytes a variable takes up by using the `sizeof(...)` operator:

```
#include <stdio.h>
#include <limits.h>

int main()
{
    int x;
    printf("The size of an int is: %d\n", sizeof(x));
    printf("The size of a float is: %d\n", sizeof(float));
}
```

- The output of this program (on my computer) is:

```
The size of an int is: 4
The size of a float is: 4
```

Integer Data Types

- In C, there are a number of **different integer data types**.
 - The difference is the number of bytes used to store the value.
 - Different number of bytes means different max and min!
- These types exist to allow you to be efficient in your use of memory.
- **Remember:** C is used for both desktop applications and embedded applications – and many **IoT applications!**
- When designing programs (especially for embedded applications) you should always consider what the **biggest value** will be of each numeric variable and choose the **smallest type** possible for that variable.

Integer Data Types

short int	int
signed short unsigned short	signed int unsigned int
long int	long long int
signed long unsigned long	signed long long unsigned long long
char	
signed char unsigned char	

Integer Data Types

- To help with representing the range of values that can be held by integer data types, C includes the “limits.h” library.
- This library defines a number of constants that represent the maximum and minimum values:
 - Signed int:
 - INT_MIN is equal to - 2147483648 (- 2^{31})
 - INT_MAX is equal to 2147483647 ($2^{31} - 1$)
 - Unsigned int:
 - It ranges from 0 to UINT_MAX = 4294967295 ($2^{32} - 1$)

Integer Data Types

- Short Data
 - A short uses **at least** 2 bytes for storage
- SHRT_MIN and SHRT_MAX
 - Are constants for minimum and maximum values of signed short
 - SHRT_MIN is equal to -32768 (-2^{15})
 - SHRT_MAX is equal to 32767 ($2^{15} - 1$)
- USHRT_MAX
 - It specifies the maximum value of unsigned short
 - It is equal to 65535 ($2^{16} - 1$)

Integer Data Types

- long long Data
 - Data of long long integer type contains **at least** 64 bits (8 bytes) for storage.
 - They have similar representation as the data type of int
 - A literal can be declared long by adding an “L” (e.g. 3L, 23452L) or long long by adding “LL”
- LLONG_MIN and LLONG_MAX
 - They are minimum and maximum values of signed long long
 - LLONG_MIN is equal to - 9223372036854775808LL (- 2^{63})
 - LLONG_MAX is equal to 9223372036854775807LL ($2^{63} - 1$)
- ULLONG_MAX
 - It specifies the maximum value of unsigned long long
 - ULLONG_MAX = $2^{64} - 1$

Integer Literals

- An integer can be specified in decimal, binary, octal, or hexadecimal
 - A leading 0 on an integer constant indicates an octal integer.
 - A leading 0x or 0X indicates a hexadecimal integer.
 - A leading 0b or 0B indicates a binary integer.
 - An 'e' can be used to define an exponential number ($\times 10^i$)
- Example
 - 30 (decimal) = 0b11110 or 0B11110 (binary)
 = 036 (octal)
 = 0X1E or 0x1E (hexadecimal)
 = 3E1 = 3e1 (exponential)

Differences between platforms

- The exact size of each data type depends on the compiler and the computer it is running on.
- `short` and `int` must be **at least** 16 bits (2 bytes)
- `long` must be **at least** 32 bits (4 bytes)
- `short` **cannot be longer** than `int`.
- `int` **cannot be longer** than `long`.

Boolean Types

- In C, true and false are represented by numbers:
 - 0 is false
 - Any other number is true
- The boolean operators we have seen before (==, !=, >, <, etc.) actually return 1 for true and 0 for false.

```
int answer = ( 5 == 2 );  
printf( "%d\n", answer );
```

- This means that we can use any numeric value in a while() loop or if() statement.

```
#include <stdio.h>
int main() {
    int total = 0;
    int finished = 0;
    while( !finished ) {
        printf( "Enter a number: " );
        int num;
        scanf( "%d", &num );
        if ( num < 0 )
            finished = 1;
        else
            total += num;
    }
    printf( "Total is: %d\n", total );
}
```

File: boolean_1.c

However...

- Although this works, it can be confusing
 - Especially to someone who didn't write the code

```
int finished = 0;  
while( !finished )
```

Boolean Type

- To make it simpler, a boolean type was introduced: this only has two possible values: 1 or 0
 - Value **1** stands for **true** and **0** for **false**
 - So, if you write the comparison (10 == 10), then this comparison is evaluated to 1.
 - Alternatively, if you write (5 > 15), then this comparison is evaluated to 0.
- The **constants** `true` and `false` are defined in “`stdbool.h`”
 - A user defined type, `bool` is also specified in this header file and can be used to declare variables with boolean data type:

```
bool b;  
b = (x > 5);
```

So...

- We can use this to make the code we just saw much more readable.

```
bool finished = false;  
while( !finished ) {
```

```
#include <stdio.h>
#include <stdbool.h>
int main() {
    int total = 0;
    bool finished = false;
    while( !finished ) {
        printf( "Enter a number: " );
        int num;
        scanf( "%d", &num );
        if ( num < 0 )
            finished = true;
        else
            total += num;
    }
    printf( "Total is: %d\n", total );
}
```

File: boolean_2.c

Char Data Type

- Char Data
 - Stores characters such as letters and punctuation
 - A character is stored as an integer according to a certain numerical code such as the ASCII code that ranges from 0 to 127, which only requires 7 bits to represent it
 - Typically, a char constant or variable occupies 1 byte (8 bits)
- CHAR_MIN and CHAR_MAX
 - CHAR_MIN = -128 (-2^7)
 - CHAR_MAX = 127 ($2^7 - 1$)
- UCHAR_MAX
 - UCHAR_MAX = 255 ($2^8 - 1$)

String Literals

- String
 - A character string literal is a sequence of 0 or more characters enclosed in double quotes
 - Remember that strings represented as character arrays end with `'\0'`
- Examples
 - `char strin_1 [6] = "abcde" // The last element is '\0'`
 - `char strin_2 [] = "This is a string."`

Real Data Types

- Floating-Point Types

- In the C language the floating-point numbers have 3 types:
- **float**: uses 32 bits (4 bytes) for its storage
- **double**: uses 64 bits (8 bytes) for its storage
- **long double**: has at least as many bits as double. It depends on the compiler and the hardware.

- Float Data Type

- The minimum and maximum positive values of float data type are defined as constants: FLT_MIN and FLT_MAX (defined in “float.h”)

- Double data Type

- The minimum and maximum values of double data type are defined as constants: DBL_MIN and DBL_MAX

Real Data Types

```
#include <float.h>
#include <stdio.h>
main() {
    printf( "FLT_MIN: %f\n", FLT_MIN );
    printf( "FLT_MAX: %f\n", FLT_MAX );
    printf( "DBL_MIN: %lf\n", DBL_MIN );
    printf( "DBL_MAX: %lf\n", DBL_MAX );
}
```

File: reals.c

Importance of knowing data types

- It is important to choose the right data type to use in your programs.
- What happens if you use a data type that is too small for your data?
- Consider this:
 - `int i = INT_MAX + 1;`
 - Clearly `i` cannot store a value larger than the maximum possible value.
 - Instead, it *overflows* and becomes the smallest possible value (i.e. `INT_MIN`)
 - This can cause programs to behave unexpectedly!

Overflows

```
#include <stdio.h>
#include <limits.h>

int main()
{
    int x = INT_MAX;
    printf( "x is : %d\n", x );
    x = x + 1;
    printf( "x is : %d\n", x );

    int y = INT_MIN;
    printf( "y is : %d\n", y );
    y = y - 1;
    printf( "y is : %d\n", y );
}
```

File: overflow.c

Overflows

- Output:

x is : 2147483647

x is : -2147483648


y is : -2147483648

y is : 2147483647

We would expect this to
be 2147483648



We would expect this to
be -2147483649



Overflows

- I normally get a few questions each year on exactly what happens during integer overflow in C.
- It turns out that the explanation is not very straight forward!
 - The Wikipedia article is pretty good
https://en.wikipedia.org/wiki/Integer_overflow
 - But for the real detail see this link (also on moodle)
<https://www.cs.utah.edu/~regehr/papers/overflow12.pdf>
- I was surprised to see that the topic was complex enough to warrant a paper at a scientific conference as late as 2012!