Additional information on variable use and their presence in memory

CS 10A – VARIABLES IN MEMORY

Additional Variable Options

- Covered: int, char, float, double, string
- There are other ways to use these variables and how they're utilized is important.
- Proper use of variables and their multiple options can prevent software-breaking bugs, such as preventing negative numbers from being used in places where they should never even occur. Or just to make your coding life more convenient.
- We can increase or decrease the amount of memory space a variable can occupy. In some applications, we can also change where in memory a variable is stored. This can affect the performance specs in applications with limited resources.

Measuring Memory in Computers

- Computers run in binary. Every value is either a 1 or a 0.
- A single bit represents one place for a 1 or 0 to occur.
- 8 consecutive bits is known as a byte.
- 1 kilobyte (kB) = 1000 bytes
  - 1 megabyte (MB) = 1000 kB
  - 1 gigabyte (GB) = 1000 MB
  - 1 terabyte (TB) = 1000 GB
- These definitions are based on the decimal standard.

Measuring Memory – 1000 or 1024?

Decimal (Metric) Standard

Bytes	Abbr.	Unit
1000	kB	kilobyte
1000^2	MB	megabyte
1000^3	GB	gigabyte
1000^4	TB	terabyte

Binary (IEC and JEDEC) Standard ($2^{10} = 1024$)

Bytes	Abbr. IEC	Unit IEC	Abbr. JEDEC	Unit JEDEC
1024	KiB	kibibyte	KB	kilobyte
1024^2	MiB	mebibyte	MB	megabyte
1024^3	GiB	gibibyte	GB	gigabyte
1024^4	TiB	tebibyte	-	-

This tera-ble joke is a *bit* overrated.

Modifying Variable Types

- long – increases memory used for variable type
- short – decreases memory used for variable type
- signed – default, allows negative numbers
- unsigned – positive numbers only
- const – variable cannot be changed during execution
  - Useful for defining constants to be used in math or physics
- …and others that we’ll probably touch upon later.

Representing Negative Numbers in Binary

- Negative numbers in binary use a conversion system called the Two's Compliment. Calculation is required.
- Convert hex and oct base to binary before using this.
- The MSB indicates the sign of the number: 0/1 = +/-.
- To interpret a negative binary value, invert all bits, then add 1.
  - 0b0111: The MSB is 0, so it's just 111 and thus 7 in base 10.
  - 0b1001: The MSB is 1, so invert the bits to get 0110, then +1 to get 0111, which is 7. Thus, in signed 4-bit memory, 0b1001 represents -7.
  - In a signed 4 bit value, the full allowed range is [-8, 7], which is still 16 values. This is because we have to spend a bit to represent all possible negative values. If unsigned, it would be [0, 2^4 - 1] = [0, 15].

Negative Numbers in Computer Memory

- Due to memory limitations, there is a finite minimum in signed int and char variables. The MSB is used to represent +/-, so half of all possible values are positive, and the other half are negative.
  - min = $-(2^{places - 1})$ // places and bits are synonymous in this context
  - max = $2^{places - 1} - 1$
  - The number of total possible values remain the same, as seen on the previously.
- When the maximum is met and then 1 is added, the value rolls over.
- In an unsigned value, the value resets to 0.
  - i.e. 1111 + 1 = 0000 = 0. The carried over 1 at the 5^{th} bit is dropped because there's no memory available to store it.
- In a signed value, the value goes to it's lowest value.
  - i.e. 0111 + 1 = 1000 = -8. Invert 1000 to get 0111, then +1 to get 1000, which is 8 in unsigned, so signed 1000 = -8.
  - Similarly, signed 1111 = -1 so if added another 1, then it becomes 0000 so -1+1 = 0.

Adding Negative Numbers in Binary

0b1100	→	- 4
+ 0b0011	→	+ 3
-------------		-------
0b1111	→	- 1

~~1~~ 1 1			← carry over
0b0110	→	6	
+ 0b1111	→	+ - 1	
-------------		------	
0b~~1~~0101	→	5	← 4 bits

Two's Complement allows us to perform subtraction in binary without having to "borrow" as we would in base 10 subtraction. We just add negative numbers instead of subtracting a positive number. Same rules apply.

Critical to adding negative numbers is being mindful of your memory size. In these examples, we're adding negative values in 4 bit memory, and so the output must also stay in 4 bits to keep the math consistent. Overflow bits are dropped.

Table of Some Variables and Memory

Type	Alt. Name	Memory (Bytes)
short int	int16	2
unsigned short int	uint16	2
int	int32	4
unsigned int	uint32	4
long int	int32	4
long long int	int64	8
char	byte	1
float		4
double		8
long double		12

Finding the Size of a Variable in C++

Program

```
int main()
{
        double x;
        cout << sizeof(string) << endl
              << sizeof(int) << endl
              << sizeof(x) << endl;
        // string's memory can vary
        // returned value is in bytes
        return 0;
}
```

Console

```
➢ ./a.exe
8
4
8
```

Using Variable Modifiers

```
// Variables are modified in the same line where they were declared
const double pi = 3.14159;        // constants are great for avoiding hard coding too
const long double e = 2.718281828459;
int main()
{
        unsigned int u_x;
        short int sh_y;
        // It's recommended that when naming vars, you mark how you modified it.

        unsigned char u_ch0;
        signed char ch0;          // char is not necessarily signed by default though
        return 0;
}
```

ASCII

- Now why would char be signed or unsigned? Because computers need to use binary to represent characters.
- A number in memory is converted to a character or action to be taken by the computer. Consider it a numeric ID for a character.
- ASCII stands for American Standard Code for Information Interchange. This is the simplest form of text representation that any computer can interpret.
- Notepad uses ASCII. Look up table. Job sites sometimes use this.
- Char is signed because most systems only support the original ASCII table of 128 characters. If a system can support the extended ASCII table as well, then the type can become unsigned to support another 128 characters.

ASCII in C++

Program

```
int ascii = 63;     // 63 is a question mark
int main()
{
        cout << (char) ascii << endl;
/*
This syntax can perform possible type
conversions. C++ does not like using ASCII, so
you have to force it to happen. If you try storing
char in an int variable or vice-versa, data is lost.
*/
        return 0;
}
```

Console

```
➢ ./a.exe
?
```