Intermediate Programming
Day 5

Outline

- ASCII characters
- Arrays
- C strings
- Review questions

Last time

- Integer types:
  - [unsigned] char: [un]signed character   (typically 1 byte)
  - [unsigned] int: [un]signed integer   (typically 4 bytes)

- Floating-point types:
  - float: single-precision floating point number   (typically 4 bytes)
  - double: double-precision floating point number   (typically 8 bytes)

https://en.wikipedia.org/wiki/C_data_types

sizeof operator

- To determine the size of a type, you can use sizeof.

```c
#include <stdio.h>
int main(void)
{
    int x = 75;
    printf( "Size of char: %d\n" , sizeof( char ) );
    printf( "Size of int: %d\n" , sizeof( x ) );
    return 0;
}
```

```
>> ./a.out
Size of char: 1
Size of int: 4
>>
```

Characters

- Character type
  - a char variable holds a single character:
    - char digit = '4';
    - char bang = '!';
  - These *must* be single quotes; double quotes are for strings, not chars
  - Behind the scenes, char is just like int:

$$char\ digit = '4'-1;$$

  digit now contains the character '3'

ASCII

- The ASCII standard governs the mapping between characters and integers.

Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0	0	000	NULL	32	20	040	 	Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SoH	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	SoTxt	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	EoTxt	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EoT	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	Enq	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	Ack	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	Bell	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	Bsp	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	HTab	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LFeed	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VTab	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FFeed	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SOut	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SIn	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAck	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	Syn	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	EoTB	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	Can	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EoM	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	Sub	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	Esc	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FSep	60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|		
29	1D	035	GSep	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RSep	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	USep	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	Delete

charstable.com

ASCII

Q: What does this print?

```c
#include <stdio.h>
int main( void )
{

    char char_0 = '0';
    int int_0 = char_0 - '0';
    printf( "Character printed as character: %c\n" , char_0 );
    printf( "Character printed as integer: %d\n" , char_0 );
    printf( "Integer printed as integer: %d\n" , int_0 );
}
```

```
>> ./a.out
Character printed as character: 0
Character printed as integer: 48
Integer printed as integer: 0
>>
```

Outline

- ASCII characters
- **Arrays**
- C strings
- Review questions

Static arrays

- Static arrays are declared/accessed using square brackets:

```c
#include <stdio.h>
int main(void)
{
        int values[2];
        values[0] = 0;
        values[1] = 130;
        printf( "Array values: %d %d\n" , values[0] , values[1] );
        return 0;

}
```

```
>> ./a.out
Array values: 0 130
>>
```

Static arrays

- Static arrays are declared/accessed using square brackets:
- C/C++ **does not** stop you from accessing values outside the array:

```
#include <stdio.h>
int main(void)
{
        int values[2];
        values[0] = 0;
        values[1] = 130;
        printf( "Array values: %d %d\n" , values[0] , values[2] );
        return 0;

}
```

```
>> ./a.out
Array values: 0 0
>>
```

Static arrays

- Static arrays are declared/accessed using square brackets:
- C/C++ **does not** stop you from accessing values outside the array

```c
#include <stdio.h>
int main(void)
{
        int values[2];
        values[0] = 0;
        values[1] = 130;
        printf( "Array values: %d %d\n" , values[0] , values[1024] );
        return 0;
}
```

```
>> ./a.out
Array values: 0 813401299
>>
```

Static arrays

- Static arrays are declared/accessed using square brackets:
- C/C++ **does not** stop you from accessing values outside the array

```c
#include <stdio.h>
int main(void)
{
        int x = 100;
        int values[2];
        int y = 100;
        values[0] = 0 ; values[1] = 1 ; values[2] = 2;
        printf( "values = { %d , %d  } , y = %d\n" , values[0] , values[1] , y );
        return 0;
}
```

```
>> ./a.out
values = { 0 , 1 } , y = 2
>>
```

Static arrays

- Static arrays are declared/accessed using square brackets:
- C/C++ **does not** stop you from accessing values outside the array

```
#include <stdio.h>
int main(void)
{
    int x = 100;
    int values[2];
    int y = 100;
    values[0] = 0 ; values[1] = 1 ; values[1000000] = 2;
    printf( "values = { %d , %d } , y = %d\n" , values[0] , values[1] , y );
    return 0;
}
```

```
>> ./a.out
Segmentation fault (core dumped)
>>
```

Static arrays

- Static arrays are declared/accessed using square brackets:
- C/C++ **does not** stop you from accessing values outside the array:
- You can declare and assign array values at the same time
  - The array size is automatically determined from the assignment

```
#include <stdio.h>
int main(void)
{
        int values[] = { 0 , 130 };
        printf( "Array values: %d %d\n" , values[0] , values[1] );
        return 0;
}
```

```
>> ./a.out
Array values: 0 130
>>
```

Static arrays

- You can determine the size of a static array using the $sizeof$ operator

```
#include <stdio.h>
int main(void)
{
        int values[] = { 0 , 130 };
        printf( "Array size: %d\n" , sizeof( values ) );
        return 0;

}
```

```
>> ./a.out
Array size: 8
>>
```

Q: Why does the array have size 8 if it only has two entries?

Outline

- ASCII characters
- Arrays
- **C strings**
- Review questions

Strings

- Strings are arrays of <u>null-terminated</u> characters
  - The null termination is required to indicate where the string ends
    - The character '\0' has value 0, so either is fine

```c
#include <stdio.h>
int main(void)
{
        char str[] = { 'h' , 'e' , 'l' , 'l' , 'o' , '\0' };
        printf( "str: %s\n" , str );
        return 0;
}
```

```
>> ./a.out
str: hello
>>
```

Strings

- Strings are arrays of <u>null-terminated</u> characters
  - The null termination is required to indicate where the string ends
    - The character '\0' has value 0, so either is fine
    - The character '\n' is a new-line
    - The character '\t' is a tab
    - The character '\"' is a quote
    - etc.

```
#include <stdio.h>
int main(void)
{
        char str[] = { 'h' , 'e' , 'l' , 'l' , 'o' , '\0' };
        printf( "str: %s\n" , str );
        return 0;
}
```

```
>> ./a.out
str: hello
>>
```

Strings

- Strings are arrays of <u>null-terminated</u> characters
  - The null termination is required to indicate where the string ends
- Can use double-quotes to assign the string value

```
#include <stdio.h>
int main(void)
{
        char str[] = "hello";
        printf( "str: %s\n" , str );
        return 0;
}
```

```
>> ./a.out
str: hello
>>
```

Strings

- Strings are arrays of <u>null-terminated</u> characters
  - The null termination is required to indicate where the string ends

- Can use double-quotes to assign the string value
  - Multiple quoted strings are merged into one long string
    - Makes it possible to split text across multiple lines

```
#include <stdio.h>
int main(void)
{
        char str[] = "hel"
                        "lo";
        printf( "str: %s\n" , str );
        return 0;
}
```

```
>> ./a.out
str: hello
>>
```

String functions (declared in `string.h`)

- **strlen**: Get the length of a string

```
#include <stdio.h>
#include <string.h>
int main( void )
{
        char str[] = "hello";
        printf( "string length : %d\n" , strlen( str ) );
        return 0;
}
```

```
>> ./a.out
string length: 5
>>
```

String functions (declared in `string.h`)

- **strlen**: Get the length of a string

```
#include <stdio.h>
#include <string.h>
int main( void )
{
        char str[] = "hello";
        printf( "string length / size: %d %d\n" , strlen( str ) , sizeof( str ) );
        return 0;
}
```

```
>> ./a.out
string length / size: 5 6
>>
```

Q: Why are the length and size different?

String functions (declared in `string.h`)

- **strcpy**: Copy the contents of one string into the other
  - The target must be large enough to store the source <u>and its null-terminator</u>

```
#include <stdio.h>
#include <string.h>
int main(void)
{
        char source[] = "hello";
        char target[6];
        strcpy( target , source );
        printf( "string: %s\n" , target );
        return 0;
}
```

```
>> ./a.out
string: hello
>>
```

String functions (declared in `string.h`)

- **`strcmp`**: Compare two strings
  - returns < 0: If the first string comes before the second
  - returns > 0: If the second string comes before the first
  - returns   0: if the strings are equal

```
#include <stdio.h>
#include <string.h>
int main(void)
{
        char str1[] = "hello";
        char str2[] = "goodbye";
        printf( "compare( %s , %s ) = %d\n" , str1 , str2 , strcmp( str1 , str2 ) );
        return 0;
}
```

```
>> ./a.out
compare( hello , goodbye ) = 1
>>
```

String functions (declared in `string.h`)

- **strtok**: Tokenizes a string
- **strcat**: Concatenates two strings
- and much much more

String functions (declared in `stdlib.h`)

- **atoi**: converts a string into a an integer
- **atof**: converts a string into a (double-precision) floating point value

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
        char str[] = "120";
        int i = atoi( str );
        double d = atof( str );
        printf( "%s -> %d : %f\n" , str, i , d );
        return 0;

}
```

```
>> ./a.out
120 -> 120 : 120.000000
>>
```

Outline

- ASCII characters
- Arrays
- C strings
- **Review questions**

Review questions

1. When we declare an array in C, what are the initial values?

Review questions

2. What is the ASCII table?

Review questions

3. What is a null terminator?

Review questions

4. Consider a c-string as "ab\0cd\0", what is the string length?

Review questions

5. How do we check if two c-strings are the same? In addition, are these two strings the same: "ab\0cd\0" and "ab\0"?

Exercise 2-2

- Website -> Course Materials -> Ex2-2