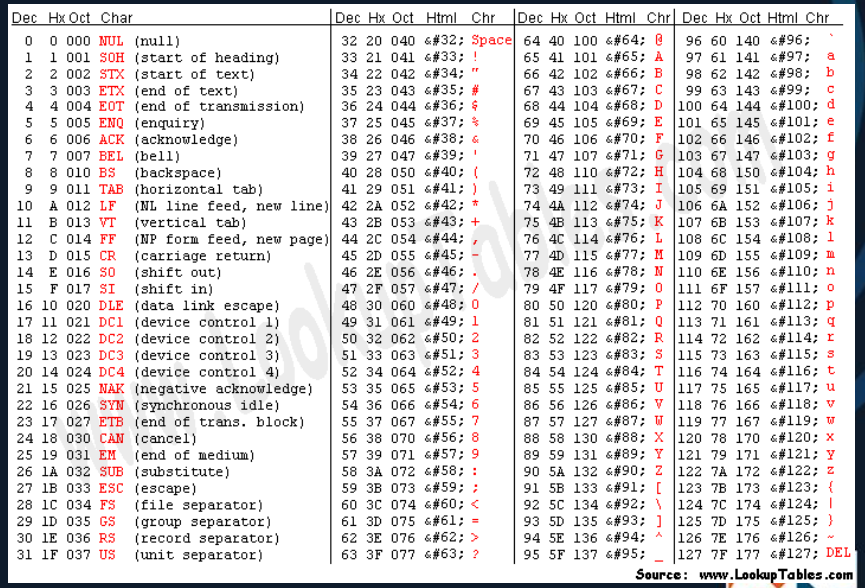
Character Arrays and String Manipulation

Contents

- ASCII
- Arrays of characters
- Null terminator
- Character functions
  - strlen
  - strcmp
  - strcpy
  - strcat

ASCI

- ASCII is a cypher that lets us store characters inside a computer.
- All variable types in C store numbers.
- ASCII maps the numbers 0-127 to different characters



Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

ASCII

- Each number from 0-127 has a corresponding letter.
- 0 – null character
- 1-31 – special characters
- 32 – 64 punctuation
- 48 – 57 – numbers
- 65 – 90 upper case letters
- 91 – 96 more punctuation
- 97-122 lower case letters
- 123 – 127 even more punctuation

Arrays of Characters

- Arrays of characters work the same as regular arrays for the most part.

```
//Uninitialized array of 16 characters
char name0[16];
//Initialized with a maximum space of 8
char name1[8] = "John";
//Initialized to be 7 characters – 6 for the name plus one for the null character
char name2[] = “Egbert";
```

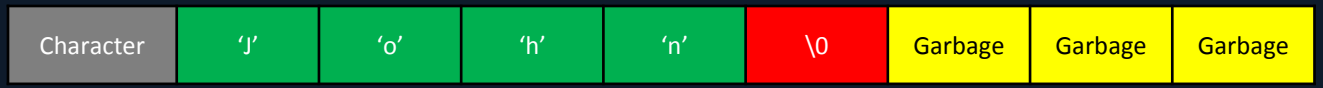
Null Terminator

- Character arrays are special.
- A character array uses something called a null terminator.
- With a regular array, you have a block of memory.
  - To keep track of how long it is, you need to have a separate variable.

Null Terminator

- Unlike other arrays, in a character array the end of a string is indicated by the value 0.
- You can’t do this with most arrays as 0 could be a valid value in the middle of the array.
- In ASCII, however, 0 is a reserved value that specifically means the end of the string.

```
char name[8] = "John";   //initialized with a maximum space of 8
```



Null Terminator

- Important things to know:
  - You still need to create an array that's big enough.
  - You only have the space you allocate.

- Be careful not to overwrite the null character.
  - Many functions expect the null terminator to be there and will crash you program (in the best case) if its not there.

Printing Character Arrays

- In the previous lecture, we covered the fact the you can't just use an array with std::cout – you have to iterate over all the elements.
- This isn't true for character arrays.
- You can give std::cout a character array and it will print all characters until the null terminator.

```
int numbers[] = {4, 1, 2, 9, 6};
char name[] = "Jarrad";

std::cout << numbers << std::endl; //DOES NOT WORK

std::cout << name << std::endl; //Does work
```

C Runtime Library Functions

- Built into the C standard library, there are a number of functions that let you examine and modify character strings.
  - strlen (String Length)
  - strcmp (String Compare)
  - strcpy (String Copy)
  - strcat (String Concatenate)

strlen – String Length

- strlen is a function that takes in a string and returns the number of characters stored until the null termination character

```
//There is space for 16 chars reserved
char name[16] = "Matthew"; //the array looks like this M-a-t-t-h-e-w-0

strlen(name); //strlen returns 7 because that's how many characters are before the 0
```

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

strlen – String Length

- strlen works by just iterating through the characters in the string, counting how far its gone until the null termination character

```
FUNCTION strlen(string)
    SET character_index to 0

    WHILE string[character_index] != 0
        character_index++
    ENDWHILE

    RETURN character_index
ENDFUNCTION
```

strcmp – String Compare

- strcmp compares if two character arrays are equal to each other.
- In C, you CANNOT just use the equality operator (==) on arrays as it only returns true if the two arrays are the exact same array, not if their contents are the same.
- To check if two strings are the same, you need to use strcmp instead.
- strcmp returns 0 if the two strings are the same, 1 if the first string is lexicographically after the second, and -1 if the first is lexicographically before the second.

strcmp – String Compare

```
char name1[] = "John";
char name2[] = "John";
char name3[] = "Jess";

if (name1 == name2)
{
    //DOES NOT WORK
}

if (strcmp(name1, name2) == 0)
{
    //DOES WORK
}

//Returns 1
strcmp(name1, name3);

//Returns -1
strcmp(name3, name1)
```

strcmp – String Compare

```
FUNCTION strcmp(string1, string2)
    SET character_index to 0
    SET result to 0
    SET running to true

    WHILE running
        IF string1[character_index] == 0 AND string2[character_index] == 0 THEN
            SET result to 0
            BREAK
        ELSEIF string1[character_index] > string2[character_index] THEN
            SET result to 1
            BREAK
        ELSE IF string1[character_index] < string2[character_index] THEN
            SET result to -1
            BREAK
        ELSE
            ++character_index
        ENDIF
    ENDWHILE
    RETURN result
ENDFUNCTION
```

strcpy – String Copy

- strcpy copies one string over another string.
- It overwrites any data that was previously in the destination string
- You need to use this any time you want to initialize one string with another string.

strcpy

```
char name1[] = "John";
char name2[64]; //Make sure you allocate enough space for what you want to copy in!!!

name2 = name1; //DOES NOT WORK

//name2 is now John
strcpy(name2, name1);
```

```
FUNCTION strcpy(destination, source)
    SET character_index to 0
    WHILE source[character_index] != 0
        SET destination[character_index] to source[character_index]
        ++character_index
    ENDWHILE

    SET destination[character_index] to 0
    RETURN destination
ENDFUNCTION
```

srtcat – String Concatenate

- Copies a string to the end of another other string.
- Unlike strcpy, it preserves the contents of the destination string.

srtcat – String Concatenate

```
//Make sure you allocate enough space for the new string to be added to the end!!!
char name1[64] = "Dennis";
char name2[] = "Richie";

name1 += name1; //DOES NOT WORK

//name2 is now John
strcat(name1, name2);
```

```
FUNCTION strcpy(destination, source)
    SET destination_index to strlen(destination) + 1
    SET source_index to 0

    WHILE source[source_index] != 0
        SET destination[destination_index] to source[source_index]
        ++destination_index
        ++source_index
    ENDWHILE
    SET destination[destination_index] to 0
    RETURN destination
ENDFUNCTION
```

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Summary

- Character arrays are a bit different from other arrays because of the null terminator
- There are a number of simple C functions that help you do common operations on character arrays.
- There is a C++ class we will cover later called std::string that can simplify a lot of these operations.