Tutorial 2 - DSA

7th May 2022

Strings | Unions | Structs

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY **DELHI**

Strings

What are strings?

Strings are just an array of characters.

So what is a character?

A character is just 1 byte (8 bits) of information.

What is the difference between a char and an int?

An int is 4 bytes. A char is 1 byte.

Okay, what if the int was 1 byte? How would the compiler know the difference between a char and an int?

It doesn't! And it doesn't need to. Why?

Because there is no difference! Char is just as much a number as int is, it's just **commonly** used to store characters, so it's called a char. When you store a character in c, it just stores the corresponding ASCII value of that character.

ASCII? Come again?

Dec	Char	Dec	Char	Dec	Char	Dec	Char
0	NUL (null)	32	SPACE	64	@	96	`
1	SOH (start of heading)	33	!	65	A	97	a
2	STX (start of text)	34	"	66	B	98	b
3	ETX (end of text)	35	#	67	C	99	c
4	EOT (end of transmission)	36	$	68	D	100	d
5	ENQ (enquiry)	37	%	69	E	101	e
6	ACK (acknowledge)	38	&	70	F	102	f
7	BEL (bell)	39	'	71	G	103	g
8	BS (backspace)	40	(	72	H	104	h
9	TAB (horizontal tab)	41	)	73	I	105	i
10	LF (NL line feed, new line)	42	*	74	J	106	j
11	VT (vertical tab)	43	+	75	K	107	k
12	FF (NP form feed, new page)	44	,	76	L	108	l
13	CR (carriage return)	45	-	77	M	109	m
14	SO (shift out)	46	.	78	N	110	n
15	SI (shift in)	47	/	79	O	111	o
16	DLE (data link escape)	48	0	80	P	112	p
17	DC1 (device control 1)	49	1	81	Q	113	q
18	DC2 (device control 2)	50	2	82	R	114	r
19	DC3 (device control 3)	51	3	83	S	115	s
20	DC4 (device control 4)	52	4	84	T	116	t
21	NAK (negative acknowledge)	53	5	85	U	117	u
22	SYN (synchronous idle)	54	6	86	V	118	v
23	ETB (end of trans. block)	55	7	87	W	119	w
24	CAN (cancel)	56	8	88	X	120	x
25	EM (end of medium)	57	9	89	Y	121	y
26	SUB (substitute)	58	:	90	Z	122	z
27	ESC (escape)	59	;	91	[	123	{
28	FS (file separator)	60	<	92	\	124	|
29	GS (group separator)	61	=	93	]	125	}
30	RS (record separator)	62	>	94	^	126	~
31	US (unit separator)	63	?	95	_	127	DEL

Source: trust me bro

So can I store characters in int data type?

Absolutely! But why would you do that –_–

It would waste memory and would be confusing to everybody.

Let’s see some boring syntax

```
char c = 'A';
```

```
char c = 65;
```

These two lines are identical to the compiler.

Let’s see some boring syntax

```
    char c = 'Z';
    printf("%c\n",c); // Z
    printf("%d\n", c); // 90
```

Back to strings

Okay, so since strings are just arrays of chars, let's declare them as such.

```
char c[] = "BCS S6E5 when?";

char c[50] = "abcd";
```

In the first case the length of the array is decided appropriately.

In the second case, what are the values stored at all the 50 bytes? How does the compiler know where the string ends?

Null *terminator*

All C strings are terminated with the value 0, represented by the char ‘\0’.

This is how the compiler knows that the string ends one byte before the null value.

Let’s smash some stacks then?

<< TO STACK SMASHING >>

Now let’s do the same stuff with pointers 😈

Why, you ask?

Because...

Now let’s do the same stuff with a pointer

Or, we’ll do that at the end if we have time remaining, else left as an exercise

String functions you should have a look at

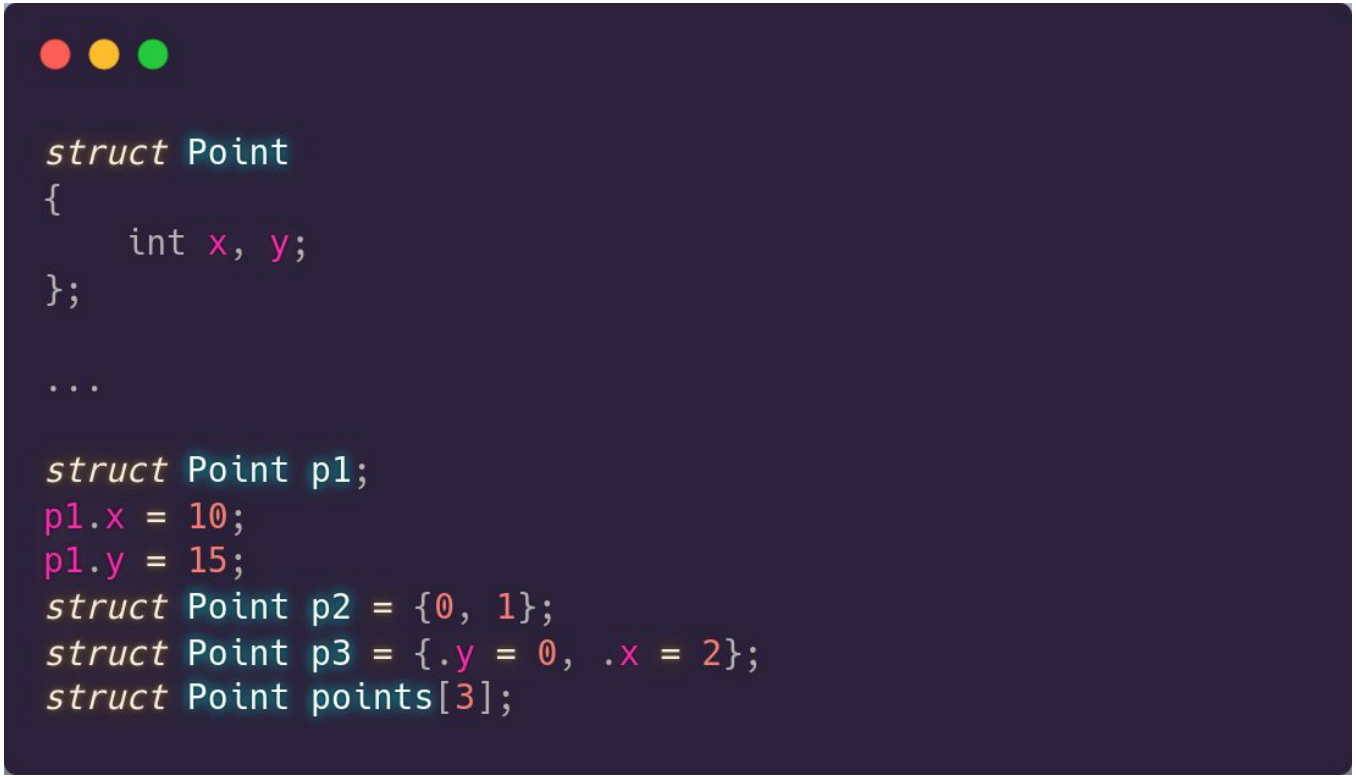
Function	Description
strlen()	Can compute the length of the string
Strcpy()	Can copy the content of a string to another
Strcat()	Is used to concatenate or join two strings
Strcmp()	Can compare two strings
Strlwr()	Can convert the string to lowercase
Strupr()	Is used to convert the letters of string to uppercase
Strrev()	Is used to reverse the string

Structures

Structures

- User defined data type.
- Collection of variables under a single name.
- Uses the ‘struct’ keyword.

Structures - Initialization



```c
struct Point
{
    int x, y;
};

...

struct Point p1;
p1.x = 10;
p1.y = 15;
struct Point p2 = {0, 1};
struct Point p3 = {.y = 0, .x = 2};
struct Point points[3];
```

Unions

Unions

- User defined data type similar to structures.
- All members share the same memory location.
- Uses the union keyword.

Unions - Initialization

```
union UnionDemo
{
    int x;
    int arr[10];
    char y;
};
...

union UnionDemo U2;

U2.x = 54;
printf("U2.x = %d", U2.x);
```