String : Collection of Characters

There is no special data type for string, we can use character array.

Char can only store 1 character.

Now suppose if i want to store a string “Ujjain”

#include<stdio.h>

int main()

{

int m[6];

//m = 'UJJAIN'; // incorrect

//m = "UJJAIN"; // incorrect

m[0]='U';

m[1]='J';

m[2]='J';

m[3]='A';

m[4]='I';

m[5]='N';

for(int y = 0; y<6; y++){

printf("%c", m[y]);

}

return 0;

}

/\*

eg120.c:5:9: warning: multi-character character constant [-Wmultichar]

m = 'UJJAIN'; // incorrect

^

eg120.c:5:9: warning: character constant too long for its type

eg120.c:5:7: error: array type 'int[6]' is not assignable

m = 'UJJAIN'; // incorrect

~ ^

eg120.c:6:7: error: array type 'int[6]' is not assignable

m = "UJJAIN"; // incorrect

~ ^

2 warnings and 2 errors generated.

\*/

%s is the format specified for String.

Lets check old example

int x;

x = 30;

printf("%d", x);

We know in above example %d will get the address of x and it will convert the 1st sign and 31 data bits to number and pring.

char m;

m = 'A';

printf("%c", m);

In above lines we pass A but its ascii value 65 will be stored and its binary will be processed.

Now lets take example

int x;

printf("%p", &x);

We know %p will print the address of x in number formar hexa form of the number.

char m[6];

m[0]='U';

m[1]='J';

m[2]='J';

m[3]='A';

m[4]='I';

m[5]='N';

printf("%s", m); // m will be changed with &m[0]

~~M is string~~

~~We are not passing ujjain to m~~

M is not a string

M does not contain ujjain.

Because of m it will be changed to &m[0] because m represents collection

Because of %s printf will consider the received number as an address and will replace %s with a collection of characters starting from the specified address.

#include<stdio.h>

int main()

{

char m[6];

m[0]='U';

m[1]='J';

m[2]='J';

m[3]='A';

m[4]='I';

m[5]='N';

printf("%s", m); //

return 0;

}

/\*

UJJAIN

\*/

But when its end?

Question: why will printf stop? Because we are not passing any length in the variable and memory allocation in almost limitless. So where and how it will stop?

#include<stdio.h>

int main()

{

char m[6];

char g[6];

g[0]='A';

g[0]='B';

m[0]='U';

m[1]='J';

m[2]='J';

m[3]='A';

m[4]='I';

m[5]='N';

printf("%s\n", m[y]);

printf("%lu\n", &m);

printf("%lu\n", &g);

printf("%d\n", &g[2]);

return 0;

}

/\*

UJJAINAB

\*/

Any output may come also sometime termination of program.

Because of %s printf will consider the received number as an address and will replace %s with a collection of characters starting from the specified address.

-> printf will stop picking when it comes across 0 as a number

0 as number means number o because 0 in character means 47 as ascii. We ne need 0 as number

And \0 and 0 both are 0 when we print

#include<stdio.h>

int main(){

char m;

m = 'A';

printf("'A' Code is %d\n", m);

m = '0';

printf("'0' Code is %d\n", m);

m = 0;

printf(" 0 only Code is %d\n", m);

m = '\0';

printf(" \\0 Code is %d\n", m);

return 0;

}

/\*

'A' Code is 65

'0' Code is 48

0 only Code is 0

\0 Code is 0

\*/

#include<stdio.h>

int main()

{

char m[7]; // one extra for string terminator

m[0]='U';

m[1]='J';

m[2]='J';

m[3]='A';

m[4]='I';

m[5]='N';

// m[5]=0';

m[6]='\0'; // very very important to end the string with string terminator

printf("%s\n", m);

return 0;

}

/\*

UJJAIN

\*/

We can also initialise as we learnt in array initialisation

#include<stdio.h>

int main()

{

char m[] = {'U', 'J', 'J','A','I','N', '\0'};

printf("%s\n", m);

printf("Number of elemts in m %lu\n", sizeof(m)/ sizeof(m[0]));

return 0;

}

/\*

UJJAIN

Number of elemts in m 7

\*/

Or

#include<stdio.h>

int main()

{

char m[] = {"Ujjain"}; // size of m will be of 7 elements

// \0 will be added implicitly

printf("%s\n", m);

printf("Number of elemts in m %lu\n", sizeof(m)/ sizeof(m[0]));

for(int y =0; y<=6; y++){

printf("at %d index, number is %d & char is %c\n", y, m[y], m[y]);

}

return 0;

}

/\*

Ujjain

Number of elemts in m 7

at 0 index, number is 85 & char is U

at 1 index, number is 106 & char is j

at 2 index, number is 106 & char is j

at 3 index, number is 97 & char is a

at 4 index, number is 105 & char is i

at 5 index, number is 110 & char is n

at 6 index, number is 0 & char is

\*/

When if we pass more then range and give small length of string

#include<stdio.h>

int main()

{

char m[15] = {"Ujjain"}; // size of m will be of 7 elements

// \0 will be added implicitly

printf("%s\n", m);

printf("Number of elemts in m %lu\n", sizeof(m)/ sizeof(m[0]));

for(int y =0; y<=14; y++){

printf("at %d index, number is %d & char is %c\n", y, m[y], m[y]);

}

return 0;

}

/\*

Ujjain

Number of elemts in m 15

at 0 index, number is 85 & char is U

at 1 index, number is 106 & char is j

at 2 index, number is 106 & char is j

at 3 index, number is 97 & char is a

at 4 index, number is 105 & char is i

at 5 index, number is 110 & char is n

at 6 index, number is 0 & char is

at 7 index, number is 0 & char is

at 8 index, number is 0 & char is

at 9 index, number is 0 & char is

at 10 index, number is 0 & char is

at 11 index, number is 0 & char is

at 12 index, number is 0 & char is

at 13 index, number is 0 & char is

at 14 index, number is 0 & char is

\*/