Standard ASCII Chart

(128 Characters from 0 to 127)

| 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 |

Extended ASCII Chart

256 Characters from 0 to 255

0 to 127 are same as above and next are comes under extended, not all can print them.

char m;

m =’A’;

**For char type value we will use single ( ‘ ) inverted comma.**

Char have 1 byte

8 bits 1 sign bit and 7 data bits

So when we assign value to m then binary value of ascii code of the given symbol will be placed

Like we given input as A then ascii value of A is 65. So binary of 65 will be assigned to m

When we print m, with %c then it will print A

And if we print m with %d thhen it will pring 65

Now what if we give m = 128

As we know char have 1 byte means 8 births in which 1 is sign and rest 7 are data bits

Range -2^7 to 2^7-1

Now if we give m = 128 means 8th bit is 1 rest will be 0

| 2^7 | 2^6 | 2^5 | 2^4 | 2^3 | 2^2 | 2^1 | 2^0 |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

When we will print

1st will be consider as sign bit

So answe will be negative

10000000

01111111 ->2’s

+ 1

—----------

10000000 that is value of 128

So final value will be -128

#include<stdio.h>

int main()

{

char m ;

m = 128;

if(m == 128){

printf("Value of m is 128\n");

}else{

printf("Value of m is not 128\n");

}

if(m == -128){

printf("Value of m is -128\n");

}else{

printf("Value of m is not -128\n");

}

return 0;

}

/\*\*

\* @Output

Value of m is not 128

Value of m is -128;

\*/

But when we use

Unsigned char m;

Then its range will become 0 to 255

Then if we will give m = 130 then it will pring 130;

What if we put more then 255

#include<stdio.h>

int main()

{

unsigned char m;

m = 255;

printf("%d\n",m); // 255

m++;

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

return 0;

}

Now if we make look to print all ascii values

Unsigned char ml

M = 0;

while(m<=255){

printf(“%d, %c”, m , m);

m++;

}

M will become 255 but when we will make m++ will it be 256?

| 256 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

When we will print it will take only right 8 digit out of 9 i.e 00000000 that will become 0

So that ascii loop will become infinity.

What to do then?

We can use int

#include<stdio.h>

int main(){

int m;

m = 0;

while(m<=255){

printf("(%d: %c) ", m,m);

m++;

}

return 0;

}

After 127 all os will not print.

We can place number in palace of characters in char type but within range of ascii code like

Char m;

M = 56; or M = ‘A’; both are same. Because ultimarlty when you put a then it will store the ascii value that is 65;

Trick:

/\*\*

Enter a number and if someone pring small then print capital

\*

\*/

#include<stdio.h>

int main(){

char m;

printf("Enter a character: ");

scanf("%c", &m);

if(m>=97 && m <=122){

// m = m-32;

m = m-' '; // when we place space it will be replace with 32 ascii

}

printf("You entered: %c\n", m);

return 0;

}