**DAY -07**

**STRING HANDLING**

**String Handling:**

* Gets- it can assign as well as store it In the buffer.
* Fgets o gets reads until a new line

-----------------🡪

**Managing input and output operations**

**Reading a character**

-reading a single charater can be done using the function getchar() and also scanf()

ch = getchar();

-i/o statements - to perform basic i/o functions c provide library funtcions called stdio.h

ex: scanf(), printf(), getchar(), putchar(), getch(), gets(), puts() etc

* There are two types of i/o:

1. Formatted i/o
2. Unformatted i/o

**1) Formatted I/O Statements** - this enables the user to specify the type of data and the way in which it should be read or written out

EX : scanf(), printf()

**2) Unformatted I/O Statements** - this do not specify the type of data and the way in which it should be read or written out

EX : getchar(), putchar(), getch(), gets(), puts()

**Scanf():-**

Syntax

scanf("Control String", address\_list);

where,

"control string" is a sequence of one or more character groups. (specifies the type of values)

"address list" are address of memory locations where the values of input variable should be stored.

%c - single character

%d - decimal integer

%f - floating point value

%u - unsigned value

Integer input

Ex : num = 386;

scanf("%3d",&num);

where, 3 is the field width of input number, other 4 and 5 are in temporary buffer wherein th erest of the place will be filled with 0.

Note:No space or any other character is used after te last character group.

Ex: scanf("%d%d%d ",p,q,r)

**Decision Making & Branching:**

• When we have multiple conditions we do branching statements.

• Branching: roots and diversions

• If(cond)

{

If block of statememnt

}

• Boolean is not a datatype.

--to test multiple condition.

1) if

2) if else

3) nested if else

4) switch statement

---if is used like if you are above 18 then you can vote and you are Indian citizen (2 things)

-if it is single statement then no need of braces

if(cond)

statement

#include <stdio.h>

int main()

{

int age;

scanf("%d",&age);

if(age>=18)

{

printf("\nYou are eligible to vote");

}

printf("\nIndian Citizen\n");

return 0;

}

o/p

12

Indian Citizen

21

You are eligible to vote

Indian Citizen

**Nested if:**

**Syntax:**

If (cond1)

{

If(cond2)

{

}

Else

{

}

}

Else{

If(cond3)

{

}

Else

{

}

}

**Nested if:** where the parent cond is not tested again

**Code:**

#include<stdio.h>

Int main()

{

Int I;

Float f;

Char c;

Char str1[20];

Double d;

Printf("enter the proper values:\n");

Printf("Enter the integer value: ");

Printf("\n Integer value: \05d\n",i);

Return 0;

}

Gcc p1.c

./a.out

How to avoid numeric constants?

Printing all the character which are present in the string

printf - used for deferencing the value

---🡪Checking in different ways

#include <stdio.h>

int main()

{

int i;

float f;

char ch;

char str1[20];

double d;

printf("\nEnter the proper values\n");

printf("\nEnter integer value: ");

scanf("%3d",&i);

printf("\nInteger value: %05d\n",i);

return 0;

}

if taken as above

input - 12345

output - 00123 (4 and 5 are in buffer)

if 5 is given rather than 05 then

i/p - 123456

o/p - 123 (spaces are occupied)

if only d is given rather than 3d then

i/p - 123456

o/p - 123456 (everything will be printed)

**Note :** While scanning the string we should not specify the address of operator

ASCII value of enter key is 10

**Note:** for single character value - it checks in the temporary buffer first then if nothing available then comes to give user input.

To avoid this - fflush stdin( it is not working in linux environment)

So we can avoid by clearing of the buffer ie we need to add scanf(" ") where a space is used

it does not work for float because it only works for single character

**-- INSERT --**

**Note :** Space or new line character is end of string (anything given after space will be in buffer)

^ - is used to find until a patten has matched or new line is found

#include <stdio.h>

int main()

{

char name[20];

char temp[20];

scanf("%[^\n]s",name);

printf("\nName: %s,name);

scanf("%s", temp);

printf(\*"\nTemp: %s,temp);

printf("\n\n");

return 0;

}

Output = Anjalli Boyapalli

if it is out of size - stack smashing detected is the error we get even though it gives proper output

this is because to clean up the memory permission is not there as it is there in java(garbage collector)

Address of i: 1440203472

Address of f: 1440203476

Address of ch: 1440203471

Address of str1: 1440203488

Address of d: 1440203480

This is called as base address of any variable(starting address)

**Code:**

#include <stdio.h>

int main()

{

int i, j;

scanf("%d%d",&i,&j);

printf("\ni=%d\tj=%d",i,j);

printf("\n\n");

return 0;

}

input can be given in same line or different line

Output - i=20 j=30

Should never have spaces in scanf and also \t and \n need to be avoided(unprintable characters (space will specify that you have an another input)

- "%d %d" is ok but "%d%d " is not ok also " %d%d" is ok only when there is integer

-itoa is used to convert integer to alpha and atoi for viseversa - to use these we need to have stdlib

-Converting an integer to string can be done using buffer and sprintf

#include <stdio.h>

#include <stdlib.h>

int main()

{

int i,j;

char w[5] = "2002";

char buff[1024];

char buff1[1024];

int res=0;

scanf("%d%d",&i,&j);

printf("\ni=%d\tj=%d",i,j);

sprintf(buff,"\ni=%d\tj=%d",i,j);

printf("\nBuffer value: %s",buff);

sprintf(buff1,"%d",i);

printf("\nBuff1: %s",buff1);

res = atoi("2002")+2;

printf("\nResult = %d",res);

printf("\n\n");

return 0;

}

**2)Unformatted**

* getc, putc ==>read or write single character -> stream
* getchar, putchar ==> read/write single char => std i/o
* gets, puts ==> read/write for a string -> stream
* getch() ==> DOS platform ==> not echo read char to the screen

**Code:**

#include <stdio.h>

int main()

{

char ch;

printf("\nEnter the character: ");

ch = getchar();

printf("\nread char: ");

putchar(ch);

printf("\n\n");

return 0;

}

Output: C when Cis given as o/p

**note** - when menu based items are used it is bettre to use getc rather than scanf

gets -- can assign and store it in the buffer

fgets should be specified with how many char and from where to be read-- it also stores one extra line

These both will read until the new line

--scanf reads until blank space and new line...to overcome this we use pattern matching

**Code:**

#include <stdio.h>

#include <string.h>

int main()

{

char str1[20];

char ch;

char buff[1024] = {'\0',};

puts("\nSlno | ID | Name | \n");

strcat(buff, "001 | ");

strcat(buff,str1);

strcat(buff," | ");

strcat(buff,"Anjalli");

strcat(buff,"Boyapalli");

strcat(buff," | ");

puts(buff);

}

**Code:**

#include <stdio.h>

#include <string.h>

int main()

{

char str1[20];

char ch;

char buff[1024] = {'\0',};

fgets(str1,3,stdin);

puts("\nSlno | ID | Name |\n");

strcat(buff, "001 | ");

strcat(buff,str1);

strcat(buff," | ");

strcat(buff,"Anjalli");

strcat(buff,"Boyapalli");

strcat(buff," |");

puts(buff);

}