**Managing input and output operations**

**Reading a Character:**

Reading a single character can be done by using the function getchar(). This can also be done with scanf().

**Syntax:** ch=getchar();

**Input/Output Statements:**

To perform the basic i/o functions C provides the library of functions. This library is called stdio.h.

**EX:** scanf(), printf(), getchar(), putchar(), gets(), puts(), etc. ---- IO ststements

There are two types of I/O ststements. They are:

* + Formatted I/O statement
  + Unformatted I/O statement

**1)Formatted statement:**

This enables the user to specify the type of the data & the way in which it should be read in or written out.

**EX:** scanf(), printf().-------- man 3 printf

**2)Unformatted Statement:**

This do not specify the type of data & the way it should read in or written out.

**EX:** getchar(), gets(), putchar(), puts(), etc.

**Scanf():**

**Syntax:** scanf(“Control String”, address\_list);

Where, “control string” is a sequence of one or more character groups.It specifies the type of the values which are to be supplied to the variables.(format specifiers)

“address\_list” are address of memory locations where the values of input variable should be stored.

**Format Specifiers:** %c,%d,%s,%f,%u(read a unsigned value)

**Integer Input:**

**EX:** num=386;

Scanf(“%3d”,&num);

Where, 3 is the field width of the input number. Other will in buffer

Note: no space or any other characters is used after the te last character group.

EX: scanf(“%d %d %d ”,&p,&q,&r);

**Formatted output:**

Printf(“%05d”, 685); ---- if the value has only 3 three digits rest will add with 0’s.

EX: vi p1.c gcc.p1.c----compile ./a.out-----output

#include <stdio.h>

int main()

{

int i;

float f;

char ch;

char str1[20];

double d;

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

printf("\nEnter the integer value: \n");

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

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

return 0;

}

Temporary Buffer: 4 ---refer the code in vi p1.c

fflush(stdin);------to remove---not work in linux

scanf(“ “)

**How to avoid numeric constants?--------Assignmnent**

**Scan the employee details and print them in a particular format**

**Id, name, gender, address, phone number, salary,designation**

SlNo ID NAME GENDER ADDRESS PHNO SALARY DESG

REFER--- vi p2.c

#include<stdio.h>

int main()

{

int ID;

char name[10];

char address[10];

char gender[10];

int phno;

int salary;

char designation[10];

/\*printf("\n Address os the variables:\n");

printf("\nAddress of i: %u",&i);\*/

printf("\n Enter the id:\n");

printf("%d", &ID);

printf("\n Enter the name:\n");

scanf("%s", name);

printf("\n Enter the gender:\n");

scanf("%s", gender);

printf("\n Enter the address:\n");

scanf("%s", address);

printf("\n Enter the phone number:\n");

scanf("%d", &phno);

printf("\n Enter the salary;\n");

scanf("%f",& salary);

printf("\n Enter the designation:\n");

scanf("%s",designation);

/\* printf("\n Employee ID: %d\n", ID);

printf("\n Employee Name: %s\n", name);

printf("\n Employee Gender: %s\n", gender);

printf("\n Employee Address: %s\n", address);

printf("\n Employee Phoneno: %d\n", phno);

printf("\n Employee salary: %f\n", salary);

printf("\n Employee designation: %s\n", designation);\*/

printf("\nSlNo | ID | NAME | GENDER | ADDRESS | PHNO |SALARY | DESIGNATION|\n");

printf("\n%03d | %d | %s | %s | %s | %d | %f | %s\n",1,ID,name,gender,address,phno,salary,designation);

printf("\n\n");

**ERROR:** stack smashing detected: terminated Aborted (core dumped)

To clean up the unreserved space we don’t have permissions(we access beyond the memory)

**ADDRESS OF STATIC VARIABLES STORED IN STACK:**

* Address of i: 1440203472
* Address of f: 76
* Address of ch:71
* Address of str1:88
* Address of d:80
* This is called as base address of a variable.
* We should not give the spaces in the scanf. If we give the space it will ask for the another input.

**SSCANF AND SPRINTF:**

To convert the string to int as well as int to string.

**UNFORMATED:** Do not mention the format specifiers.

unformated i/o

getc, putc => read/write single char -> 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

**EX:**

#include <stdio.h>

int main()

{

char ch;

printf("\nEnter the character:");

ch=getchar();

printf("\nRead char:");

putchar(ch);

putchar('A');

putchar (65);

printf("\n\n");

return 0;

}

**EX:**

#include <stdio.h>

int main()

{

char ch;

ch = fgetc(stdin); (or) getc(stdin)

printf("\nRead character:");

putc(ch,stdout);

printf("\n\n");

return 0;

}

**STRING HANDLING:**

#include<stdio.h>

int main()

{

char empName[20];

gets(empName);

puts(empName);

if(strcmp(gets(buff),"quit")==0)

{

quit code

}

else

{

other than quit code

}

return 0;

}

**EX:**

#include<stdio.h>

int main()

{

char empName[20];

int i;

fgets(empName,5,stdin);

puts(empName);

return 0;

}

char buff[1024]—bits={‘\0’,}---null value

string concatenation------strcat

strcat(buff,”001”);--- the buffer is null now we add the 001.

fgets(str1,4,stdin)------read the input

**EX:**

#include<stdio.h>

#include<string.h>

int main()

{

char str1[20]="101";

char ch;

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

fgets(str1,4,stdin);

puts("\nSlNo | ID | NAME |\n");

strcat(buff,"01 | ");

strcat(buff,str1);

strcat(buff,"| ");

strcat(buff,"bhima");

strcat(buff,"shankar");

strcat(buff," |");

puts(buff);

return 0;

}

**DECISION MAKING AND BRANCHING**

**Branching Statements:** To check multiple statements.

* if statement
* if else
* Nested if else
* Switch statement

**If statement:**

**Syntax:** if(condition)

{

If block of statements;

}

**EX:**

#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;

}

**if else statement:**

**Syntax:**

if(condition)

{

}

else { }

**Nested if else: tcp and udp in networks**

**Syntax:**

if(cond1)

{

if(cond2)

{

}

else

{

}

}

else

{

if(cond3)

{

}

else{

}

}

**EX:** Greatest of 5 numbers by using nested if

#include <stdio.h>

int main() {

int a,b,c,d,e;

scanf("%d%d%d%d%d",&a,&b,&c,&d,&e);

if(a>b)

{

if(a>c)

{

if(a>d)

{

if(a>e)

{

printf("a is bigger");

}

else

{

printf("e is bigger");

}

}

else

{

if(d>e)

{

printf("d is bigger");

}

else

{

printf("e is bigger");

}

}

}

else

{

if(c>d)

{

if(c>e)

{

printf("c is bigger");

}

else

{

printf("e is bigger");

}

}

else

{

if(d>e)

{

printf("d is bigger");

}

else

{

printf("e is bigger");

}

}

}

}

else

{

if(b>c)

{

if(b>d)

{

if(b>e)

{

printf("b is bigger");

}

else

{

printf("e is bigger");

}

}

else

{

if(d>e)

{

printf("d is bigger");

}

else

{

printf("e is bigger");

}

}

}

else

{

if(c>d)

{

if(c>e)

{

printf("c is bigger");

}

else

{

printf("e is bigger");

}

}

else

{

if(d>e)

{

printf("d is bigger");

}

else

{

printf("e is bigger");

}

}

}

}

printf("\n\n");

return 0;

}

**If else ladder:**

**Syntax:**

if(cond1)

{

}

else if(cond2)

{

}

else if(cond3)

{

}

else

{

}

#include <stdio.h>

int main()

{

int a=100;

int b=2;

int c=300;

int d=40;

int e=50;

if(a>b && a>c && a>d && a>e)

{

printf("\nA is the greatest");

}

else if(b>a && b>c && b>d && b>e) or (b>c && b>d && b>e)

{

printf("\nB is greatest");

}

else if(c>a && c>b && c>d && c>e) or ( c>d && c>e)

{

printf("\nC is greatest");

}

else if(d>a && d>b && d>c && d>e) or (d>e)

{

printf("\nD is greatest");

}

else

{

printf("\nE is greatest");

}

return 0;

}

**EX:**

#include<stdio.h>

int main()

{

char clrCode;

printf("\n Enter the color:\n");

clrCode = getchar();

if(clrCode == 'B')

printf("\nBlack\n");

else if(clrCode == 'N')

printf("\nBrown\n");

else if(clrCode == 'R')

printf("\n Red\n");

else if(clrCode =='O')

printf("\n Orange\n");

else

printf("\n Enter the correct color code\n");

printf("\n Program Ended\n");

return 0;

}

**SWITCH:** Case value should be numeric or ASCII value because of binary search.

* It is forming a table of unique values.
* Cannot use duplicate values.

**Syntax:**

Switch(cond)

{

Case 1:

---

Break;

Case 2:

---

Break;

Default:

---- break; (optional) }

**EX:**

#include<stdio.h>

int main()

{

char clrCode;

printf("\n Enter the character:\n");

//clrCode = getc(stdin);

switch(clrCode = getc(stdin))

{

case 'o':

printf("\n Color is Orange\n");

break;

case 'r':

printf("\n Color is Red\n");

break;

default:

printf("\n Enter the correct color code\n");

}

printf("\n Program Ended\n");

return 0;

}

**DEBUGGING:** gcc -g file name for gebugging gdb ./a.out

Process exited normally---After debugging