\*To free the memory that we have allocated to pointer.

=>#include <stdio.h>

#include<stdlib.h>

int main() {

int \*Ptr=NULL;

Ptr=(int \*)malloc(3\*sizeof(int));

free(Ptr);

free(Ptr);

printf("\n\n");

return 0;

}

Strings:\*Collection of characters

Declaration of characters:

char Name[5]={'H','e','l','l','o'};

Declaration of Strings:

char name[6]={'H','e','l','l','o','\0'};

char name[6]="Hello";=>Here compiler its self will allocate null

char greet[]="Hello World";=>here we are declaring without size and compiler

its self will allocate null and gives size.

char str1[21];

char Names[][20];=>here it takes 19 column size and a null for each row

char Names[10][50];=>here there are 10 names each of max cap of 50 characters

\*collection of charactfers with null

\*if we are printing characters there is no need of null only with string we

need null

=>#include <stdio.h>

int main() {

char Name[5];

int i;

for(i=0;i<20;i++)

scanf("%c",&Name[i]);

for(i=0;i<5;i++)

printf("\n%c=%d",Name[i],Name[i]);

printf("\n\n");

scanf("\n%s",Name);

puts(Name);

printf("\n\n");

return 0;

}

Here we are giving size as 5 but in loop we have given 20 in loop and it will give illegl values

=>#include <stdio.h>

int main() {

char Name[5];

int i;

for(i=0;i<20;i++)

scanf("%c",&Name[i]);

Name[5-1]='\0';

for(i=0;i<5;i++)

printf("\n%c=%d",Name[i],Name[i]);

printf("\n\n");

scanf("\n%s",Name);

puts(Name);

printf("\n\n");

return 0;

}=>same as above

=>strcpy( char \*dest, const char \*src);function copies the string pointed to by src,

including the terminating null byte('\0') to the buffer pointed to by dest.

#include <stdio.h>

#include<string.h>

int main() {

char s1[20],s2[20];

char d1[40],d2[40];

int ret=0;

scanf("%s%s",s1,s2);

printf("\ns1=%s\ts2=%s",s1,s2);

/\*

ret =strcmp(s1,s2);

printf("\nRet=%d",ret);

\*/

strcpy(d1,s1);

printf("\nd1=%s",d1);

printf("\n\n");

return 0;

}

=>#include <stdio.h>

#include<string.h>

#include<stdlib.h>

int main() {

char s1[20],s2[20];

char d1[40],d2[40];

char \*ptr=NULL;

int ret=0;

scanf("%s%s",s1,s2);

printf("\ns1=%s\ts2=%s",s1,s2);

ptr=(char\*)malloc(strlen(s1)+1);

/\*

ret =strcmp(s1,s2);

printf("\nRet=%d",ret);

strcpy(d1,s1);

printf("\nd1=%s",d1);

\*/

ptr=strcpy(d1,s1);

printf("\ns1=%d",d1);

printf("\nptr=%s",ptr);

printf("\nAddress of d1=%u\nAdress of ptr=%u\n",&d1,ptr);

printf("\n\n");

return 0;

}

=>srtncpy(char \*dest, const char \*src, size);function is similar,except that at most n bytes of

src are copied.

=>strcat

=>strncat

\*Note:Destination should always greater than source

=>strcmp:if s1 is grater than s2 then it will give positive values,s1 is less

than s2 it will give negative and if both are equal it will give zero.

#include <stdio.h>

#include<string.h>

int main() {

char s1[20],s2[20];

char d1[40],d2[40];

int ret=0;

scanf("%s%s",s1,s2);

printf("\ns1=%s\ts2=%s",s1,s2);

ret =strcmp(s1,s2);

printf("\nRet=%d",ret);

printf("\n\n");

return 0;

}

=>strncmp

\*Note:Here it compares ASCII values of string

=>memset and memcpy:when we want to initialize structures we use these.

=>strchr(to find first occurence of character from main string)

=>strrchr(to find last occurence of character from main string)

=>strstr(to find first occurence of string from main string)

=>strtok(tokenizing of strings,to see occurence of each character and stop at ;(Delimiter) )

=>strlen(excluding the null character)

=>strerror(capture string handling errors)

Assignment

\*reverse a string

#include <stdio.h>

#include<string.h>

int main() {

char s1[8];

int i,length,k;

printf("Enter a string:");

scanf("%s",s1);

length=strlen(s1);

for(i=length-1;i>=0;i--)

{

printf("\nreversing a string:%c",s1[i]);

}

return 0;

}

\*reverse a string at kth interval

UDT(user defined datatypes):

1.structures(struct):In structures we can not define functions

\*always define structure in header for accesing it everywhere

Syntax:

struct tagname

{

members of structure

};

EX:

struct square

{

int len;

int breadth;

}

EX:

struct chair

{

int nolegs;

char make[20];

char material[20];

char clr[20];

float price;

}

=>//demo on struct

#include <stdio.h>

#include<string.h>

struct Employee

{

int eId;

char eName[20];

float esal;

char eGender;

char eAddress[20];

};

typedef struct Employee Emp;

int main() {

Emp e1;

e1.eId=101;

e1.esal=100000;

strcpy(e1.eName,"Bhima");

strcpy(e1.eAddress,"rjpt");

e1.eGender='M';

printf("\nEmployee Details are\n");

printf("\nID:%d",e1.eId);

printf("\nName:%s",e1.eName);

printf("\nGender:%c",e1.eGender);

printf("\nAddress:%s",e1.eAddress);

printf("\nSalary:%f",e1.esal);

printf("\n\n");

return 0;

}

2.unions(union)

3.enums(enum)

. =>static var => namevar.membername

-> =>ptr var => namevar->membername

Assignments:

WAP to which should have the following functionalities

1. scan employee recs

2. display

3. search for a particulary employee based on

a. id

b. lName

4. search for employess whos salary fall in

between 6-10K and display only those records

5. update employee rec to change his/her lName

#include <stdio.h>

#include <string.h>

struct Employee

{

int id;

char fname[20];

char lname[20];

float salary;

char salvation[5];

};

typedef struct Employee emp;

void scanEmployee(emp e[], int n);

void displayEmployee(emp e[], int n);

int searchEmployeeid(emp e[], int n, int searchid);

void todisplayiddetails(emp e[], int n, int searchid);

int searchEmployeelname(emp e[], int n, char searchlname[20]);

void todisplaylnamedetails(emp e[], int n, char searchlname[20]);

int searchEmployeeBySalaryRange(emp e[], int n, float minSalary, float maxSalary);

void todisplaySalaryRangeDetails(emp e[], int n, float minSalary, float maxSalary);

int main()

{

emp e[3];

scanEmployee(e, 3);

displayEmployee(e, 3);

int searchid;

printf("Enter an id to search: ");

scanf("%d", &searchid);

todisplayiddetails(e, 3, searchid);

char searchlname[20];

printf("Enter a last name to search: ");

scanf("%s", searchlname);

todisplaylnamedetails(e, 3, searchlname);

float minSalary, maxSalary;

printf("\nEnter minimum salary: ");

scanf("%f", &minSalary);

printf("Enter maximum salary: ");

scanf("%f", &maxSalary);

printf("\nEmployees with salary between %.2f and %.2f:\n", minSalary, maxSalary);

todisplaySalaryRangeDetails(e, 3, minSalary, maxSalary);

return 0;

}

void scanEmployee(emp e[], int n)

{

int i;

for (i = 0; i < n; i++)

{

printf("Enter employee ID: ");

scanf("%d", &e[i].id);

printf("Enter first name: ");

scanf("%s", e[i].fname);

printf("Enter last name: ");

scanf("%s", e[i].lname);

printf("Enter salary: ");

scanf("%f", &e[i].salary);

printf("Enter salvation: ");

scanf("%s", e[i].salvation);

}

}

void displayEmployee(emp e[], int n)

{

int i;

for (i = 0; i < n; i++)

{

printf("\nEmployee ID: %d\n", e[i].id);

printf("First Name: %s\n", e[i].fname);

printf("Last Name: %s\n", e[i].lname);

printf("Salary: %.2f\n", e[i].salary);

printf("Salvation: %s\n", e[i].salvation);

}

}

int searchEmployeeid(emp e[], int n, int searchid)

{

int i;

for (i = 0; i < n; i++)

{

if (e[i].id == searchid)

{

return i;

}

else{

return -1;

}

}

}

void todisplayiddetails(emp e[], int n, int searchid)

{

int detailsfound = searchEmployeeid(e, n, searchid);

if (detailsfound != -1)

{

printf("\nEmployee ID: %d\n", e[detailsfound].id);

printf("First Name: %s\n", e[detailsfound].fname);

printf("Last Name: %s\n", e[detailsfound].lname);

printf("Salary: %.2f\n", e[detailsfound].salary);

printf("Salvation: %s\n", e[detailsfound].salvation);

}

else

{

printf("id details not found\n");

}

}

int searchEmployeelname(emp e[], int n, char searchlname[20])

{

int i;

for (i = 0; i < n; i++)

{

if (strcmp(e[i].lname, searchlname) == 0)

{

return i;

}

else

{

return -1;

}

}

}

void todisplaylnamedetails(emp e[], int n, char searchlname[20])

{

int namedetailsfound = searchEmployeelname(e, n, searchlname);

if (namedetailsfound != -1)

{

printf("\nEmployee ID: %d\n", e[namedetailsfound].id);

printf("First Name: %s\n", e[namedetailsfound].fname);

printf("Last Name: %s\n", e[namedetailsfound].lname);

printf("Salary: %.2f\n", e[namedetailsfound].salary);

printf("Salvation: %s\n", e[namedetailsfound].salvation);

}

else

{

printf("Entered name details not found\n");

}

}

int searchEmployeeBySalaryRange(emp e[], int n, float minSalary, float maxSalary)

{

int i;

for (i = 0; i < n; i++)

{

if (e[i].salary >= minSalary && e[i].salary <= maxSalary)

{

return i;

}

else

{

return -1;

}

}

}

void todisplaySalaryRangeDetails(emp e[], int n, float minSalary, float maxSalary)

{

int found = 0;

int i;

for (i = 0; i < n; i++)

{

if (e[i].salary >= minSalary && e[i].salary <= maxSalary)

{

printf("\nEmployee ID: %d\n", e[i].id);

printf("First Name: %s\n", e[i].fname);

printf("Last Name: %s\n", e[i].lname);

printf("Salary: %.2f\n", e[i].salary);

printf("Salvation: %s\n", e[i].salvation);

found = 1;

}

}

if (!found)

{

printf("No employees found with salary between minSalary, maxSalary");

}

}