Unions:Members of unions are access one member at a time.

union item

{

int m;

float x;

char c;

}

=>/\*

UNION DEMO

\*/

#include <stdio.h>

#include<string.h>

#include<stdlib.h>

typedef struct sEmployee

{

int id;

float sal;

char g;

char Name[20];

}sEMP;

typedef union uEmployee

{

int id;

float sal;

char g;

char Name[20];

}uEMP;

int main() {

sEMP es;

uEMP eu;

printf("\nSize of struct =%d",sizeof(es));

printf("\nSize of union =%d",sizeof(eu));

return 0;

}

output:Size of struct =32

Size of union =20

=> #include <stdio.h>

#include<string.h>

#include<stdlib.h>

typedef struct sEmployee

{

int id;

int sal;

char g;

char Name[20];

}sEMP;

typedef union uEmployee

{

int id;

float sal;

char g;

char Name[20];

}uEMP;

int main() {

sEMP es;

uEMP eu;

printf("\nSize of struct =%d",sizeof(es));

printf("\nSize of union =%d",sizeof(eu));

eu.id=101;

eu.sal=10000;=>here sal is overrided id so sal will print.

printf("\nID=%d",eu.sal);

return 0;

}

output:Size of struct =32

Size of union =20

ID=4202538

=>#include <stdio.h>

#include<string.h>

#include<stdlib.h>

typedef struct sEmployee

{

int id;

int sal;

char g;

char Name[20];

}sEMP;

typedef union uEmployee

{

int id;

float sal;

char g;

char Name[20];

}uEMP;

int main() {

sEMP es;

uEMP eu;

printf("\nSize of struct =%d",sizeof(es));

printf("\nSize of union =%d",sizeof(eu));

eu.id=101;

eu.sal=10000;

strcpy(eu.Name,"PatanMahida");

=>here name is string but we are giving integer format it will give ascii value of patanmahida

printf("\nName=%s",eu.Name);

output:Name=PatanMahida

printf("\nID=%d",eu.Name);

return 0;

}

=>#include <stdio.h>

#include<string.h>

#include<stdlib.h>

typedef struct sEmployee

{

int id;

int sal;

char g;

char Name[20];

}sEMP;

typedef union uEmployee

{

int id;

float sal;

char g;

char Name[20];

}uEMP;

int main() {

sEMP es;

uEMP eu;

uEMP \*eu1=NULL;

printf("\nSize of struct =%d",sizeof(es));

printf("\nSize of union =%d",sizeof(eu));

eu.id=101;

eu.sal=10000;

strcpy(eu.Name,"PatanMahida");

printf("\nName=%s",eu.Name);

eu1=(uEMP \*)malloc(sizeof(uEMP));

eu1->id=102;

printf("\nID:%d",eu1->id);

return 0;

}

output:Size of struct =32

Size of union =20

Name=PatanMahida

ID:102

ENUMERATION(enum):is a set of named integer constants that specify all the legal values a

variable of that type can have.

\*The keyword enum signals the start of an enumeration type

SYNTAX:

enum enum\_type\_name{enumeration list}variable\_list;

\*enum is staring from starting index(0)

=>/\*

ENUMERATION

\*/

#include <stdio.h>

#include<string.h>

#include<stdlib.h>

enum daysOfWeek

{

Monday,#Monday=10000

Tuesday,

Wednesday,

Thrusday,#Thursday=265

Friday,

Saturday,

Sunday,

}Days;

int main()

{

Days=Friday;

printf("\nDay=%d",Days);

printf("\n\n");

return 0;

}

OUTPUT:Day=4

Day=100004

Day=266

=>/\*

ENUMERATION

\*/

#include <stdio.h>

enum RegClrCodes

{

Black,

Brown,

Red,

Orange,

Yellow,

Green,

Blue,

Voilate,

Gray,

White,

}RCC;

int dispMenu()

{

int cc;

printf("\nPress,");

printf("\n0.Black");

printf("\n1.Brown");

printf("\n2.Red");

printf("\n3.Orange");

printf("\n4.Yellow");

printf("\n5.Green");

printf("\n6.Blue");

printf("\n7.Voilate");

printf("\n8.Gray");

printf("\n9.White");

scanf("%d",&cc);

return cc;

}

int main()

{

switch(dispMenu())

{

case Black:

printf("\nBlack");

break;

case Brown:

printf("\nBrown");

break;

case Red:

printf("\nRed");

break;

case Orange:

printf("\nOrange");

break;

case Yellow:

printf("\nYellow");

break;

case Green:

printf("\nGreen");

break;

case Blue:

printf("\nBlue");

break;

case Voilate:

printf("\nVoilate");

break;

case Gray:

printf("\nGrayk");

break;

case White:

printf("\nWhite");

break;

default:

printf("\nEnter correct choice");

printf("\nProgram Ended");

}

}

OUTPUT:

Press,

0.Black

1.Brown

2.Red

3.Orange

4.Yellow

5.Green

6.Blue

7.Voilate

8.Gray

9.White

10

Enter correct choice

Program Ended

=>#include <stdio.h>

enum RegClrCodes

{

Black=101,

Brown,

Red,

Orange,

Yellow=303,

Green,

Blue,

Voilate,

Gray,

White,

}RCC;

int dispMenu()

{

printf("\nPress,");

printf("\n%d.Black",Black);

printf("\n%d.Brown",Brown);

printf("\n%d.Red",Red);

printf("\n%d.Orange",Orange);

printf("\n%d.Yellow",Yellow);

printf("\n%d.Green",Green);

printf("\n%d.Blue",Blue);

printf("\n%d.Voilate",Voilate);

printf("\n%d.Gray",Gray);

printf("\n%d.White",White);

scanf("%d",&RCC);

return RCC;

}

int main()

{

switch(dispMenu())

{

case Black:

printf("\nBlack");

break;

case Brown:

printf("\nBrown");

break;

case Red:

printf("\nRed");

break;

case Orange:

printf("\nOrange");

break;

case Yellow:

printf("\nYellow");

break;

case Green:

printf("\nGreen");

break;

case Blue:

printf("\nBlue");

break;

case Voilate:

printf("\nVoilate");

break;

case Gray:

printf("\nGrayk");

break;

case White:

printf("\nWhite");

break;

default:

printf("\nEnter correct choice");

printf("\nProgram Ended");

}

}

output:

Press,

101.Black

102.Brown

103.Red

104.Orange

303.Yellow

304.Green

305.Blue

306.Voilate

307.Gray

308.White

303

Yellow

FILES:A collection of logically realted information.

Two Types:

1)sequential file:All records are arranged in a particular order.

2)Random Access file:Files are accessed at random.

\*To Access a file we need a pointer.

The only declaration needed for a file pointer is exemplified by:

FILE \*fp;

FILE \*fopen(char \*name,chat \*mode);

fp=fopen("file name","mode");

\*If the file is open it will give the address of file if not it will give null.

\*fclose():to close a file

\*Once the funtion fopen()returns a FILE type pointer stored in a sppointer of type

FILE,this pointer becomes teh medium through which all subsequent I/O

can be performed.

=>#include <stdio.h>

/\*

FILE =>DS

fd =>file pointer to FILE DS

fopen("NameofFILE","Mode")=>opening a file

Mode=>r,w,a,r+,w+,a+

fclose(fd)=>close the opened file

read/write

fprintf,fscnaf=>formated writing and reading resp

fputs,fgets=>unformated i/o opertaion

fwrite/fread=>reading/writing binary objects

\*/

int main() {

FILE \*fd=NULL;

fd=fopen("Test1.txt","W");

if(fd=NULL);

{

perror("fopen: ");

printf("\n%p\n\n",fd);

exit(1);

}

printf("\nOpened the file successfully\n\n");

fclose(fd);

return 0;

}

=>#include <stdio.h>

int main() {

FILE \*fd=NULL;

char ch;

fd=fopen("Test1.txt","r");

if(fd=NULL);

{

perror("fopen: ");

printf("\n%p\n\n",fd);

exit(1);

}

printf("\nOpened the file successfully %p \n\n",fd);

while((ch=fgetc(fd)!=EOF))

{

putchar(ch);

}

fclose(fd);

return 0;

}=>to display the file data on screen

=>To copy content from one file to another file.

#include <stdio.h>

#include <stdlib.h>

int main() {

FILE \*sourceFile;

FILE \*destFile;

char sourcePath[100];

char destPath[100];

char ch;

sourceFile=fopen("day00.c","r");

destFile=fopen("day01.c","w");

if(sourceFile==NULL || destFile==NULL)

{

printf("Unable to open file");

printf("please check if file exists or not");

exit(1);

}

fclose(sourceFile);

while (ch=fgets(sourceFile)!=EOF)

{

fputc(ch,destFile);

}

}

Assignment:Read the contents from the database the record of the employees

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_NAME\_LEN 50

// Define a structure for a single record

struct Record {

int id;

char name[MAX\_NAME\_LEN];

char gender;

int number1;

int number2;

};

int main() {

FILE \*file = fopen("day02.txt", "r"); // Open the file for reading

if (file == NULL) {

printf("Error opening file\n");

return 1;

}

struct Record records[100]; // Array to hold records (assuming there are 100 records)

int count = 0; // Counter for how many records are read

// Read the file line by line

while (fscanf(file, "%d|%49[^|]|%c|%d|%d\n",

&records[count].id,

records[count].name,

&records[count].gender,

&records[count].number1,

&records[count].number2) == 5) {

count++; // Increment the record count

}

fclose(file); // Close the file

// Print all records to verify

for (int i = 0; i < count; i++) {

printf("ID: %d, Name: %s, Gender: %c, Number1: %d, Number2: %d\n",

records[i].id,

records[i].name,

records[i].gender,

records[i].number1,

records[i].number2);

}

return 0;

}

=>#include<stdio.h>

#include<string.h>

#include<stdlib.h>

typedef struct Employees

{

int id;

int sal;

char Name[20];

char Gender;

int Phno;

FILE \*file=fopen(day02.txt,"r");

if(file==NULL)

{

printf("Error opening file\n");

exist(1);

}

}EMP;

int display(EMP \*);

int main()

{

char Names[10][20];

int row=0,col=0,i;

char \*ptr=NULL;

EMP e1;

EMP \*testEmp=NULL;

testEmp = ((EMP \*)malloc(sizeof(EMP)));

ptr = strtok(file,"|");

do{

strcpy(Names[row],ptr);

ptr = strtok(NULL,"|");

row++;

}while(ptr != NULL);

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

printf("\n%s",Names[i]);

e1.id = atoi(Names[0]);

strcpy(e1.Name,Names[1]);

e1.Gender = Names[2][0];

e1.Phno = atoi(Names[3]);

e1.sal = atoi(Names[4]);

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

if(display(testEmp)==1)

{

printf("\nError in Display\n");

return 1;

}

printf("\n\n");

return 0;

}

=>how to allocate memory based on records

=>by using system we can not exceute system commands

using execl we can