C Notes

0 stars 0 forks 1 watching Activity

Public repository

main

Branches Tags

aknsubbu ... on Jul 14

View code

C Notes

Programs

Program to print hello world

```c
#include <stdio.h>


int main() {
  printf("Hello World");
  return 0;
}
```

Program and to take runtime input of two numbers and print it

```c
#include <stdio.h>
void main(){
    int x,y;
    printf(">>");
    scanf("%d",&x);
    printf(">>");
    scanf("%d",&y);
    printf("the numbers are %d and %d",x,y);
}
```

Take a number as input and apply postfix increment to it

```
int main(){
   int a=1,b=5,c;
   c=b++;
}
```

readme.md

Take a charecter as an input and print its value until 13 is entered

```
int main(){
   char a;
   a=getchar();
   while(a!=13){
       printf("%c",a);
       a=getchar();
   }
}
```

Write a program to give three numbers as input

```
#include <stdio.h>


int main() {
  int b1,b2,b3,product;
  printf("Enter number 1:");
  scanf("%d",&b1);
  printf("Enter number 2:");
  scanf("%d",&b2);
  printf("Enter number 3:");
  scanf("%d",&b3);
  product=b1*b2*b3;
  printf("/nThe product is:%d",product);
}
```

Program to convert celsius to farenheit using float datatype

```
int main(){
    float cel,faren;
    printf("Enter the celsius value of the number to be converted:");
    scanf("%g", &cel);
    faren=(1.8*cel)+32;
    printf("The converted value is %g",faren);
}
```

Program to convert celsius to fahrenheit using double data type

```
int main(){
  double cel,faren;
    printf("Enter the celsius value of the number to be converted:");
    scanf("%lf", &cel);
    faren=((double)1.8*cel)+32;
```

```
    printf("The converted value is %.20lf",faren);

}
```

Size of Data Types

```
#include <stdio.h>

int main() {
  float x;
  double y;
  long double z;
  char a;

  x = sizeof(float);
  y = sizeof(double);
  z = sizeof(long double);
  a = sizeof(char);

  printf("Size of float: %lu\n", sizeof(x));
  printf("Size of double: %lu\n", sizeof(y));
  printf("Size of long double: %lu\n", sizeof(z));
  printf("Size of char: %lu\n", sizeof(a));

  return 0;
}
```

Print Odd Integers upto 20

```
int main(){
  int i;
  for(i=0;i<=20;i=i+3){
    printf("%d\n",i);
  }
}
```

Data type required to store 2^100 completely

```
int main(){
  //datatype to store 2^100
  double i,x;
  for(i=1;i<=100;i=i+1){
    x=pow(2,i);
    printf("%lf\n",x);
  }
}
```

Number Pyramid

```
void numberPyramid(){
   int num;
   printf("Enter the number of rows you want: ");
   scanf("%d",&num);
   for(int i=1;i<=n;i++){
       for(int j=1;j=<num-i;j++){
```

```
            printf(" ");
        }
        for(int j=1;j<=i;j++){
            printf('* ');
        }
        printf("\n");
    }
}
```

write a program to print numbers from 1 to 10 and repeat from form till end number is given

```
int main(){
    int n,num_count;
    printf("Enter the number of digits to be printed: ");
    scanf("%d",&n);
    num_count=0;
    while (num_count<n){
        for (int i=0;i<5;i++){
            if (num_count<n){
                num_count++;
            } else {
                break;
            }
            printf("%d",i);
        }
    }
}
```

```
int main(){
    for (int i=1;i<=100;i++){
        if (i%5!=0){
            printf("%d",i);
        }
    }
}
```

```
int main(){
    int i=1;
    while(i<50){
        if (i%5==0){
            i++;
            continue;
        }

        printf("%d\n",i);
        i++;
    }
}
```

multiplication table where specific number is to be ommitted

```
int main(){
    int base,omit;
    printf("Enter the base value: ");
    scanf("%d",&base);
```

```
    printf("Enter the number whose multiples are to be omitted: ");
    scanf("%d",$omit);
    for (int i=1;i<=50;i++){
        if (i%omit==0){
            continue;
        }
        printf("%d\n",base*i);
    }
}
```

multiplication table where only multiples of a specific number are shown

```
int main(){
    int base,inc;
    printf("Enter the base value: ");
    scanf("%d",&base);
    printf("Enter the number whose multiples are to be included: ");
    scanf("%d",&inc);
    for (int i=1;i<=50;i++){
        if (i%inc==0){
            printf("%d\n",base*i);
        }

    }
}
```

scan number digit limit

```
int main(){
    float  a;
    scanf("%5f",&a);
    printf("%f",a);
}
```

```
array creating
int main(){
    int arr[100];
    for(int i=0;i<100;i++){
        printf("Enter element: ");
        int ele;
        scanf("%d",&ele);
        arr[i]=ele;
    }
    for(int i=0;i<100;i++){
        printf("%d",arr[i]);
    }
}

3x4 matrix
int main(){
    float matrix[3][4];


}
```

Consider an array with base address B. The elements amy be stored in row or column major.
We have to find the address of the element / index mentioned Number of bytes=n.
Total number of rows=r Total number of columns=c

```
int main() {
   int B;  // base address
   int n;  // number of bytes
   int r;  // number of rows
   int c;  // number of columns
   int i;  // row index
   int j;  // column index
   int addr;  // address of element

   // input the values of B, n, r, c, i, and j
   printf("Enter the base address B: ");
   scanf("%d", &B);
   printf("Enter the number of bytes per element n: ");
   scanf("%d", &n);
   printf("Enter the number of rows r: ");
   scanf("%d", &r);
   printf("Enter the number of columns c: ");
   scanf("%d", &c);
   printf("Enter the row index i: ");
   scanf("%d", &i);
   printf("Enter the column index j: ");
   scanf("%d", &j);

   // calculate the address of ele
   addr = B + (i * c + j) * n;
   // output the address
   printf("The address of the element at row %d and column %d is %d\n", i, j, addr);

}
```

Program to find the avg of an array of 100 elements

```
int main(){
   int arr[100];
   for (int i=0;i<100;i++){
       printf("Enter the %d element: ",i);
       scanf("%d",&arr[i]);
   }
   int sum=0;
   for (int x=0;x<100;x++){
       sum=sum+arr[x];
   }
   int avg=sum/100;
   printf("The average of the 100 numbers is %d",avg);

}
```

Program to swap adjacent elements in an array

```
int main(){
   int size;
   printf("Enter the size of the array: ");
```

```
    scanf("%d",&size);
    int arr[size];
    // filling the arr
    for(int i=0;i<size;i++){
        int x;
        printf("Enter the element: ");
        scanf("%d",&x);
        arr[i]=x;
    }
    //swapping
    for (int j=0;j<size;j=j+2){
        int temp1=arr[j];
        int temp2=arr[j+1];
        int temp3=arr[size-1];
        arr[j]=temp2;
        arr[j+1]=temp1;
        arr[size-1]=temp3;
    }
    //printing the arr
    for (int s=0;s<size;s++){
        printf("%d",arr[s]);
    }
}
```

Declare array of 3x2 format and get input, print and sum

```
int main(){
    int arr1[3][2];
    int sum=0;
    for (int i=0;i<3;i++){
        for (int j=0;j<2;j++){
            printf("Enter the element: ");
            scanf("%d",&arr1[i][j]);
        }
    }
    for (int i=0;i<3;i++){
        for (int j=0;j<2;j++){
            printf("%d  ",arr1[i][j]);
        }
        printf("\n");
    }
    for (int i=0;i<3;i++){
        for (int j=0;j<2;j++){
            sum=sum+arr1[i][j];
        }
    }
    printf("%d",sum);
}
```

Print the address of an array and prove if it uses row major or column major

```
int main(){
        int arr1[3][2];
        for (int i=0;i<3;i++){
            for (int j=0;j<2;j++){
                printf("Enter the element: ");
                scanf("%d",&arr1[i][j]);
            }
```

```
        }
        for (int i=0;i<3;i++){
            for (int j=0;j<2;j++){
                printf("%d  ",&arr1[i][j]);
            }
            printf("\n");

        int pos1=&arr1[0][0],pos2=&arr1[0][1];
        if (pos1==pos2+4){
            printf("row major");
        }
        }
}
```

Print an upward triangular matrix and a lower triangular matrix

```
int main(){
   int arr1[3][3];
   int arr2[3][3];
   //upward triangle
   for (int i=0;i<3;i++){
       for (int j=i;j<3;j++){
           printf("Enter element");
           scanf("%d  ",&arr1[i][j]);
       }
       printf("\n");
   }
   //lower triangle
   for (int j=0;j<3;j++){
       for (int i=0;i<j=;i++){
           printf("Enter element");
           scanf("%d  ",&arr2[i][j]);
       }
       printf("\n");
}}
```

swtich case statement create menu ask for arthemetic operations

```
void calcMenu (){
   char c;
   int result;
   int a,b;
   do{
   printf("Add\n");
   printf("Subtract\n");
   printf("Multiply\n");
   printf("Divide\n");
   printf("End\n");
   printf("Enter the starting letter of your chosen operation in lowercase: ");
   scanf("%c",&c);
   printf("Enter number 1: ");
   scanf("%d",&a);
   printf("Enter number 2: ");
   scanf("%d",&b);


   switch (c){
       case 'a':
```

```
            printf("The sum of the two numbers are: %d",a+b);
            break;
        case 's':
            printf("The difference between the two numbers are: %d",a-b);
            break;
        case 'm':
            printf("The product of the two numbers are: %d",a*b);
            break;
        case 'd':
            printf("The quotient of the two number after divison is: %d \n",a/b);
            break;
    }
    }while(c!='e');
}
```

get charecter as input using scanf

```
void getCharUsingScanf(){
  char c;
  printf("Enter your charecter: ");
  scanf("%c",&c);
}
```

getchar function prototype

```
void getCharUsinggetchar(){
   char c;
   printf("Enter your charecter: ");
   c=getchar();
   printf("%c",c);
}
```

getch function prototype

```
void getcharUsinggetch(){
   char c;
   printf("Enter your charecter: ");
   c=getch();
   printf("%c",c);
}
```

getch function prototype

```
void getcharUsinggetche(){
   char c;
   printf("Enter your charecter: ");
   c=getche();
   printf("%c",c);
}
```

Get the input for one string without spaces using gets

```
void getstringUsinggets(){
   char c[10];
   printf("Enter your string: ");
   gets(c);
   printf("%s",c);
}
```

Get one input of a string with spaces

```
void getstrWithSpaces(){
char str[100];
printf("Enter a string with spaces: ");
scanf("%[^\n]s", str);
printf("You entered: %s\n", str);
}
```

Get paragraph as input

```
void getpara(){
   int max_length;
   printf("Enter the max length of the paragraph: ");
   scanf("%d",&max_length);
   char para[max_length];
   char line[100];
   int length=0;
   printf("Enter paragraph: ");
   while(fgets(line,sizeof(line),stdin)!=NULL){
       strcat(para,line);
       length+=strlen(line);
       if(length>=max_length-1){
           break;
       }
   }
   printf("The para: %s",para);
}
```

get input until "d" is pressed

```
   char input;
   printf("Enter input (press 'd' to exit):\n");
   while ((input = getchar()) != 'd') {
       printf("You entered: %c\n", input);
   }
}
```

Ternary Operators variable= (conditon)?var1:var2 works as checks against condition having var 1 and var 2 for example and returns one var based on the true or false eval

```
void TernaryOperation(){
   //example
   int max =(1<2)?1:2;
   // check for odd or even
   int a;
   printf("Enter the number 'a' for odd or even check: ");
```

```
    scanf("%d",&a);
    (a%2==0)?printf("Even"):printf("Odd");
    //find the greatest of  three numbers suing only ternary operator
    printf("Greatest between three numbers computation");
    int x,y,z;
    printf("Enter the number 'x': ");
    scanf("%d",&x);
    printf("Enter the number 'y': ");
    scanf("%d",&y);
    printf("Enter the number 'z': ");
    scanf("%d",&z);
    (x>y && x>z)?printf("x is maximum"):(y>x && y>z)?printf("b is maximum"):(z>x && z>y)?printf("z is
}
```

bitwise operations

```
    char a=5,b=1,zero=0;
    printf("%d\n",a&b);
    printf("%d\n",a&&b);
    printf("%d\n",a|b);
    printf("%d\n",a^b);
    printf("%d\n",~b);
    printf("%d\n",a<<b);
    printf("%d",a>>b);
    printf("%d",~zero);
}
```

```
void formatSpecifier(){
    char x[100];
    scanf("%[a-zA-z ]",x);
    scanf("%[^+-/*]",x);
}
```

Write a program to find the sum of the digits of a given number using a while loop.

```
    int n,q,r,sum=0;
    printf("enter the number: ");
    scanf("%d",&n);
    while(n>0){
        r=n%10;
        q=n/10;
        n=q;
        sum=sum+r;
    }
    printf("%d",sum);
}
```

Write a program that takes a character as input and uses the ternary operator to print whether the character is a vowel or a consonant.

```
void vowelTernary(){
    char x;
    printf("Enter the charecter: ");
```

```
    x=getche();
    char vowel[5]={'a','e','i','o','u'};
    for(int i=0;i<5;i++){
        if(x==vowel[i]){
            printf("%c is a vowel",x);
        } else{
            printf("%c is a consonant",x);
        }
    }
}
```

Write a program that takes an integer as input, which represents a month number (1-12), and uses the switch statement to print the corresponding month name.

```
    int x;
    printf("Enter the number of the month: ");
    scanf("%d",&x);
    switch(x){
        case 1:
            printf("January");
            break;
        case 2:
            printf("February");
            break;
        case 3:
            printf("March");
            break;
        case 4:
            printf("April");
            break;
        case 5:
            printf("May");
            break;
        case 6:
            printf("June");
            break;
        case 7:
            printf("July");
            break;
        case 8:
            printf("August");
            break;
        case 9:
            printf("September");
            break;
        case 10:
            printf("October");
            break;
        case 11:
            printf("November");
            break;
        case 12:
            printf("December");
            break;
```

```
    }
}
```

Write program to print pattern

```
    *
   * *
  * * *
   * *
    *
```

```
void Pattern(){
    int n;
    printf("Enter the value of n: ");
    scanf("%d", &n);

    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n-i; j++) {
            printf(" ");
        }
        for (int j = 1; j <= i; j++) {
            printf("* ");
        }
        printf("\n");
    }

    for (int i = n-1; i >= 1; i--) {
        // Print the spaces
        for (int j = 1; j <= n-i; j++) {
            printf(" ");
        }
        for (int j = 1; j <= i; j++) {
            printf("* ");
        }
        printf("\n");
    }
}
```

Write a program to check if it is a palindrome

```
void program() {
    int arr[10]={2,3,4};
    int index=2;
    int size=3;
    int elements[]={5,6,7};
    int elesize= sizeof(elements)/sizeof(elements[0]);
    for(int i=size-1;i>=index;i--){
        arr[i+elesize]=arr[i];
    }
    for(int i=0;i<elesize;i++){
        arr[index+i]=elements[i];
    }


}
```

Find if a number is a factor of another number

```
int isFactor(int a,int b){
   if (a%b==0){
       return(1);
   } else{
       return(0);
   }
}
```

Find if a number is prime or not

```
int isPrime(int x){
   scanf("%d",&x);
   for (int i=0;i<((x/2)+1);i++){
       if(isFactor(x,i)){
           continue;
       } else{
           return(0);
       }
   }
   return(1);
}
```

Find the prime numbers in a range

```
int PrimeRange(int s,int e){
   for (int i=s;i<(e+1);i++){
       if(isPrime(s)){
           printf("%d is a prime",i);
       }else{continue;}
   }
}
```

Sort the numbers in an array

```
int sort(int x[]){
   for (int i=0;i<3;i++){
       for(int j=0;j<(4);j++){
       printf("%d",x);}
   }
   return 0;
}

int main(){
       int num[][4]={1,2,3,4,5,6,7,8,9,10};
    sort(num);
}
```

Stack

- A stack is built using last in first out logic
- It has push and pop operations where we can control only the top-most element of the stack

- The top pointer shows the top-most element of the stack

factorial using recursion

```
int factorial(int n){
   if(n==0){
       return 1;
   } else{
       return (n*factorial(n-1));
   }
}
```

fibonacci using recursion

```
int fibonacci( int n) {
   if (n <= 1){
       return n;}
   else
       return (fibonacci(n - 1) + fibonacci(n - 2));
}
```

tower of hanoi

```
void towerOfHanoi(int n, char source, char auxiliary, char destination) {
   if (n == 1) {
       printf("Move disc 1 from %c to %c\n", source, destination);
       return;
   }

   towerOfHanoi(n - 1, source, destination, auxiliary);
   printf("Move disc %d from %c to %c\n", n, source, destination);
   towerOfHanoi(n - 1, auxiliary, source, destination);
}
```

Structures

--------EXAMPLE--------

```
struc students{
char name[50],
rollno[7],
int age,
float marks[6]};
```

Each element like name , age etc... are called stucture members Finish the declaration with a semicolon A structure member can be accessed only through a structure variable using the dot operator

```
struct students{
        char name[50];
       char rollno[7];
        int age ;
       float marks[6];    };
```

```
int mainimplement(){
   struct students s;
   gets(s.name);
   scanf("%d",&s.age);
   gets(s.rollno);
   for (int i=0;i<6;i++){
       scanf("%d",&s.marks[i]);
   }

}
```

auto and global variable illustration

```
void test(){
   extern int i;
   i++;
   printf("%d",i);
}
int main(){
   test();
   printf("%d",i);
   test();
}
int i=10;
```

copy contents of a string

```
void copy(char org[], char new[]){
   strcpy(new,org);
}
```

create a struc and print its contents

```
struct Student{
   char name[50],rollno[6],phno[10];
   int age;
   float marks[5];
}e,s = {"Anand","22Z209","9384870740",18,{100,100,100,100,100}};

int main(){
   //struct Student s;
   printf("Enter name, roll no, phone number, age and marks in 5 subjects: ");
   printf("Name: %s\nRoll No: %s\nPhone Number: %s\nAge: %d\nMarks in 5 subjects: %f %f %f %f %f",s.
   printf("\n%s",s.rollno);
}
void structProgram3(){
   struct Student e={.name='Anand',.rollno='22Z209',.phno='9384870740',.age=18,.marks={100,100,100,1
   printf("Enter name, roll no, phone number, age and marks in 5 subjects: ");
   printf("Name: %s\nRoll No: %s\nPhone Number: %s\nAge: %d\nMarks in 5 subjects: %f %f %f %f %f",e.
   printf("\n%s",e.rollno);
}
```

Structures Practice

```c
struct Person{
   char name[50];
   float height;
   int age;
}p[3] = {{"guy1",185,85},{"guy2",175,75},{"guy3",165,65}};

int main(){
   printf("%s\n",p[0].name);
   printf("%f\n",p[0].height);
   printf("%d\n",p[0].age);
   printf("%s\n",p[1].name);
   printf("%f\n",p[1].height);
   printf("%d\n",p[1].age);
   printf("%s\n",p[2].name);
   printf("%f\n",p[2].height);
   printf("%d\n",p[2].age);

}


struct Student {
   char name[50], rollno[6], phno[10];
   int age;
   float marks[5];
};

void calculateAveragePerStudent(struct Student student) {
   float sum = 0;
   for (int i = 0; i < 5; i++) {
       sum += student.marks[i];
   }
   float average = sum / 5;
   printf("Average marks for student %s: %.2f\n", student.name, average);
}

void calculateAverageForArray(struct Student students[], int size) {
   float sum = 0;
   int totalMarks = 0;

   for (int i = 0; i < size; i++) {
       for (int j = 0; j < 5; j++) {
           sum += students[i].marks[j];
           totalMarks++;
       }
   }

   float average = sum / totalMarks;
   printf("Average marks for the entire array: %.2f\n", average);
}

int main(){
   struct Student students[3]={{"Anand","22Z209","9384870740",18,{100,100,100,100,100,}},
                               {"Anand","22Z209","9384870740",18,{100,100,100,100,100,}},
                               {"Anand","22Z209","9384870740",18,{100,100,100,100,100,}}};
   for (int i=0;i<3;i++){
       calculateAveragePerStudent(students[i]);
   }
   calculateAverageForArray(students,3)
}
```

Pointers in C

```
int type1(){
   int a;

   int *ptr=&a;
   printf("%d\n",a);
   printf("%d",&a);}

int type2(){
   float f;
   float* fptr=&f;
   printf("Enter f:");
   scanf("%f",&f);
   printf("%f\n",f);
   printf("%p\n",&f);
   printf("%f",*fptr);}


int type3(){
int n1[4]={1,2,3,4},n2=6;
int *p1=&n1;
int *p2=&n2;
printf("%d \n %d \n %d \n %d \n %d  \n %p\n %p \n",n1[0],n1[1],n1[2],n1[3],n2,p1,p2);
   printf("%d \n %d \n %d \n %d \n %d  \n %p\n %p \n",&n1[0],&n1[1],&n1[2],&n1[3],&n2,&p1,*p2);
   for(int i=0;i<4;i++){
       printf("%d",*(p1++));
   }
}
```

Storage Classes in C

1. Automatic Storage Classes in C : (auto) Every variable defined in a function or block belongs to automatic storage class by default. The variable of the block belong to the block and are declared with the auto specifier. The variables are destroyed when the control exits the block.
2. Register Storage Classes in C : (register) They are equivalent to auto class in c but are stored in the CPU registers and not the storage. These varibale can be quickly accessed as they are in the CPU register. The are usually reserved for frequently used variables. Variables of this class are local to the block inwhich they are defined. They are destroyed as soon as the control exits the block. We cannot get the address of a register varibale and they are not given a initial value unless specified.
3. Static Storage Classes in C : (static) Static functions are within a function or a file depended on whether it is a local or global variable. They are not destroyed when the control exits the block. They are initialized to 0 if not initialized. They are initialized only once. They are stored in the data segment of the memory. Local Varibale : - They are contained inside a specific block or function, a permanent storage space is created in the compiler. The stati local variable is visible to the function or the block it is a part of. They retain their old values and donot change their values on reentering the function or block. Global Variable: - They are contained in a file, a permanent storage space is created in the compiler. The static global variable is visible to all the functions in the file. They retain their old values and donot change their values on reentering the function or block.
4. External Storage Classes in C : (extern) External storage class allows us to define a variable that is specified but not initialized there. It siginifies to the complier that it hass already been initialized somewhere else. This allows us to assign/ initialize the value of the variable even after using it and it will give a uniform answer throughtout the program.It can be used to share data between multiple files. It is used to declare a global variable or function in another file.

Multi-Dimensional Pointers: -

- Here

Create a simple array of pointers for a matrix of 4x4 integers.

```
    int array[4];
    int *p[4];
    int a=4,b=5,c=4,d=6;
    p[0]=&array[0];
    p[1]=&array[1];
    p[2]=&array[2];
    p[3]=&array[3];
    for (int i = 0; i < 4; i++)
        printf("%p\n", p[i]);


}
```

Add pointers to independent variables into an array of pointers

```
int main(){
    int array[4];
    int *p[4];
    int a=4,b=5,c=4,d=6;
    p[0]=&a;
    p[1]=&b;
    p[2]=&c;
    p[3]=&d;
    for (int i = 0; i < 4; i++){
        printf("%p\n", p[i]);
        printf("%d\n",*p[i]);
    }
}
```

Create a pointer to a two dimensional array : This creates and iterates through the first elements of the subarray and prints the values of the first elements

```
int main(){
    int arr[3][3]={
            {1,2,3},
            {4,5,6},
            {7,8,9}
    };
    int (*p)[3]=arr;
    for (int i = 0; i < 3; i++){
        printf("%p\n", p[i]);
        printf("%d\n",*p[i]);

    }
}
```

Create a pointer to a two dimensional array and print the values of all the elements

```c
int main(){
    int arr[3][3]={
            {1,2,3},
            {4,5,6},
            {7,8,9}
    };
    int (*p)[3]=arr;
    for (int i = 0; i < 3; i++){
        //printf("%p\n", p[i]);
        //printf("%d\n",*p[i]);
        for (int j=0;j<3;j++){
            //**IMPORTANT**
            printf("%d\n",*(*(p+i)+j));
        }
    }
}
```

Create an array of pointers which point to float values

```c
int main(){
    float *ptr[5];
    float a=1.2,b=2.4,c=3.6,d=4.8,e=6.0;
    ptr[0]=&a;
    ptr[1]=&b;
    ptr[2]=&c;
    ptr[3]=&d;
    ptr[4]=&e;
    //print the 4th element
    printf("%f\n",*ptr[3]);
    //print the address at which 2.4 is located
    printf("%p\n",ptr[3]);
}
```

Note: Here p [3] and 3 [p] both will work as will p+3 and 3+p as they are mathematical equivalents to each other

Write a simple program to create a structure of the items in the store which tells us the name of the item, no of items and unit price.

```c
struct Item{
    char name[100];
    int quantity;
    int price;
};
int findQuant(struct Item items[],int size){
    char str[100];
    printf("Enter the name of the element to find the price: ");
    gets(str);
    for(int i=0;i<size;i++){
        if(!(strcmp(str,items[i].name))){
            printf("The price of the item is: %d\n",items[i].price);
            printf("The quantity of the specified items are: %d\n",items[i].quantity);
        }
    }
}
int main(){
```

```
    struct Item soap[5]={
             {"Soap1",100,100},
             {"Soap2",100,100},
             {"Soap3",100,100},
             {"Soap4",100,100},
             {"Soap5",100,100}};
    findQuant(soap,5);

}
```

File Pointer

Opening a File: -

```
file_pointer=fopen("file name","mode");
```

Closing a file: -

```
fclose(file_pointer);
```

Modes: -

1. Read Mode (r): We can use this to read he data from within a file. We can use fgets() to get the data from the file. If we try to read from a file that does not exist the it will return an error. So we add an error handler code snippet

```
if (file_pointer==NULL){
    printf("File doesn't exist");
}
```

```
fgets(data from file, MAX LENGTH of the file, file pointer);
```

2. Write Mode (w): We can use this to write data to a file. If the file does not exist then it will create a new file. If the file already exists then it will overwrite the data in the file. We can use fputs() to write data to a file. It takes the pointer to the beggining of the file.

```
fputs(data to be written, file pointer);
```

3. Append Mode (a): We can use this to write data to a file. If the file does not exist then it will create a new file. If the file already exists then it will append the data to the end of the file. We can use fputs() to write data to a file. It takes the pointer to the end of the file.

Commands and Methods in File Operations: -

1. **fscan** : - This is used to scan the data from the file.

```
fscanf(file_pointer,"%d",&variable);
```

2. **fprint**: - This is used to print the data to the file.

```
fprintf(file_pointer,"%d",variable);
```

3. **fgetc**: - This is used to get a single character from the file.

```
fgetc(file_pointer);
```

4. **fputc**: - This is used to put a single character into the file.

```
fputc('a',file_pointer);
```

5. ***fseek***: - This is used to move the file pointer to a specific location.

```
fseek(file_pointer,0,SEEK_END);
```

6. **ftell**: - This is used to tell the current location of the file pointer.

```
ftell(file_pointer);
```

7. **rewind**: - This is used to move the file pointer to the beginning of the file.

```
rewind(file_pointer);
```

8. **feof**: - This is used to check if the file pointer is at the end of the file.

```
feof(file_pointer);
```

9. **ferror**: - This is used to check if there is an error in the file.

```
ferror(file_pointer);
```

10. **clearerr**: - This is used to clear the error in the file.

```
clearerr(file_pointer);
```

11. **fread**: - This is used to read a block of data from the file.

```
fread(data,sizeof(data),1,file_pointer);
```

12. **fwrite**: - This is used to write a block of data to the file.

```
fwrite(data,sizeof(data),1,file_pointer);
```

Basic Read and Write Operation

```
void main(){
    char str[100],data[100];
    FILE *fptr,*ptr;
    fptr=fopen("test.txt","w");
    gets(str);
    fputs(str,fptr);
    fclose(fptr);
    ptr=fopen("test.txt","r");
    fgets(data,sizeof(data),ptr);
    printf("%s",str);
}
```

Reading a file charecter by charecter

```
void main(){
    char str[100],data[100];
    FILE *fptr,*ptr;
    ptr=fopen("text.txt","r");
//    fgets(data,sizeof(data),ptr);
char x;
do{
    x=fgetc(ptr);
    printf("%c",x);
}while(x!= EOF || NULL);
}
```

Multiple Line read and return data along with position of the pointer

```
void main(){
    int pos;
    char str[100],data[100];
    FILE *ptr;
    ptr=fopen("text.txt","r");
while(fgets(data,sizeof(data),ptr)){
    pos=ftell(ptr);
    printf("%s\n",str);
    printf("Positon of the pointer is: ")
    printf("%d\n",pos);
}

}
```

Writing a structure to a binary file

```
struct Item{
    char name[20];
    int price;
    int quantity;
};
void main(){
    FILE *ptr;
    struct Item x={
        "Apple",
```

```
        100,
        10
    };
    ptr=fopen("items.bin","wb");
    fwrite(&x,sizeof(struct Item),1,ptr);
}
```

Reading a structure from a binary file

```
#include <stdio.h>
struct Item{
    char name[20];
    int price;
    int quantity;
};
void main(){
    FILE *ptr;
    struct Item x;
    ptr=fopen("items.bin","rb");
    fread(&x,sizeof(x),1,ptr);
    printf("Name: %s\nPrice: %d\nQuantity: %d\n",x.name,x.price,x.quantity);
}
```

Writing a number to a file using fprintf

```
#include <stdio.h>
void main(){
    FILE *ptr;
    ptr=fopen("text.txt","w");
    int integer=5;
    fprintf(ptr,"The integer is %d",integer);
    fclose(ptr);
}
```

Reading an integer from a file usding fscanf

```
#include <stdio.h>
int main(){
    FILE *ptr;
    int num;
    ptr=fopen("text.txt","r");
    if (ptr==NULL){
        printf("The file does not exist\n");
    } else{
        printf("The file exists\n");
    };
    fscanf(ptr,"%d",&num);
    printf("The value of a is %d\n",num);
    fclose(ptr);
    return 0;
}
```

Create a program to write the details of students to a file and then read and print the data after which we have to calculate the average marks per subject

```c
#include<stdio.h>
struct Student {
    char name[100];
    char roll_no[100];
    int marks[2];
}
int main(){
    FILE *ptr;
    ptr = fopen("student.txt","w");
    int n;
    printf("Enter the number of students :");
    scanf("%d",&n);
    struct Student names[n];
    for (int i=0;i<n;i++){
        printf("Enter the name of the student: ");
        gets(n[i].name);
        printf("Enter the roll no of the student: ");
        gets(n[i].roll_no);
        for (int j=0;j<2;j++){
            printf("Enter the marks for subject %d: ",j+1);
            scanf("%d",n[i].marks[j]);
        }
    }
    fwrite(names,sizeof(struct Student),n,ptr);
    fclose(ptr);

    // reopen to read the marks
    ptr=fopen("student.txt","r");
    struct Students x[n];
    fread(x,sizeof(Struct Student),n,ptr);
    for (int i=0;i<n;i++){
        printf("Name:%s",x[i].name);
        printf("Roll Number: %s",x[i].roll_no);
        for (int j=0;j<2;j++){
            printf("Marks for subject ",x[i].marks[j]);

        }
    }
    //calculate the average of marks per subject
    for (int j=0;j<2;j++){
        for(inti=0;i<n;i++){
            sum+=x[i].marks[j];
        }
        printf("Average marks for subject %d is %d",j+1,sum/n);
    }
    fclose(ptr);



}
```

enum method in c

```c
#include <stdio.h>
enum ErrorCode {On,Off};
int main(){
    enum ErrorCode status;
    status = On;
```

```
    printf("%d\n",status);
    status = Off;
    printf("%d\n",status);
    return 0;
}
```

Create a union and compare the size with an identical structure

```
#include<stdio.h>
union Student {
    char name[100];
    char roll_no[100];
    int marks[2];
};
struct Students {
    char name[100];
    char roll_no[100];
    int marks[2];
};
int main(){
    union Student s1;
    printf("%d\n",sizeof(s1));
    struct Students s2;
    printf("%d\n",sizeof(s2));
    return 0;
}
```

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

C 66.8% Makefile 23.8% CMake 9.3% TypeScript 0.1%

Suggested Workflows

Based on your tech stack

Actions Importer Set up

Automatically convert CI/CD files to YAML for GitHub Actions.

C/C++ with Make Configure

Build and test a C/C++ project using Make.

CMake based, multi-platform projects

Configure

Build and test a CMake based project on multiple platforms.

More workflows

Dismiss suggestions