Fiver user name REMYNISAM

Password Remy681471

// ternary conditional operator in c

X==1?Printf(“number positive”): Printf(“number negative”);

variable = Expression1 ? Expression2 : Expression3

@\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*@

Int x=1;

Int y=1;

Printf(“%d\n”,(x&&y));

it will check value in both x and y if both are 1 it will print the value 1.

@\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*@

Functions advantage🡺multiple time it can be used

* It can be reusable
* Function we should declare first void printHello();
* Function should be called in main program

int main()

{

printHello();

return 0;

}

* Function (definition)are written after the main program

Void printHello

{

Printf(“Hello world!”);

}

Function property=> Function returns only one value ..

Changes to parameters in a function will not change values in main function

Because calling function will be passing a copy of value in main function..

@\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*@

Function can be declared as return type or without return type

Int printable (int a,int b); this function can be passed with values a and b to a function

printable(int a,int b)

#include <stdio.h>

#include <math.h>

void printTable(int num);

int main()

{

  int num;

  printf("Enter the number");

  scanf("%d",&num);

  printTable(num);

  return 0;

}

void printTable(int num)

{

for (int i = 1; i <= 10; i++)

{

  printf("%d\n",i\*num);

}

}

This is an example of passing a variable in a function and returning a value back

#include <stdio.h>

#include <math.h>

int sum(int num1, int num2);

int main()

{

  int num1, num2;

  printf("Enter the first number");

  scanf("%d", &num1);

  printf("Enter the second number");

  scanf("%d", &num2);

  printf("Sum of two numbers are %d", sum(num1, num2));

  return 0;

}

int sum(int num1, int num2)

{

  int sum = num1 + num2;

  return sum;

}

Arguments versus parameters

++++++++++++++++++++++++++++

Arguments are values that are passed in function call, used to send values , they are actual parameters

Parameters are values in function declaration and definition, used to receive values , formal parameters.

**Recursion-function calling it self repeatedly**

* Recursion uses **selection structure**.
* Any thing that can be done with iteration (loop) can be done with recursion
* Recursion terminates when a **base case** is recognized.
* Recursion makes the **code smaller**.
  + Iteration has infinite loop but recursion has stack overflow( stack overflow is **a runtime error that happens when a program runs out of memory in the call stack**. )

**Important example of recursion is Fibonacci series**

#include <stdio.h>

#include <math.h>

int fibinacci(int num);

int main()

{

  int num;

  printf("Enter a number");

  scanf("%d", &num);

  printf("fibinacci is %d", fibinacci(num));

  return 0;

}

int fibinacci(int num)

{

  if (num == 1)

  {

    return 1;

  }

  if (num == 0)

  {

    return 0;

  }

  int fib = fibinacci(num - 1) + fibinacci(num - 2);

  return fib;

}

**Using iteration or by loop we write as given below**

#include <stdio.h>

#include <math.h>

int fibinacci(int num);

int main()

{

  int num;

  printf("Enter a number");

  scanf("%d", &num);

  printf("fibinacci is %d", fibinacci(num));

  return 0;

}

int fibinacci(int num)

{

  int fib1=0, fib2=1, fib;

  for(int i=1;i<=num;i++)

  {

    fib1=fib2;

    fib2=fib;

    fib =fib1+fib2;

  }

  return fib;

}

**Pointers**

#include <stdio.h>

#include <math.h>

// pointer

int main()

{

  int x;

  int \*ptr;

  ptr=&x;

  printf("%d\n",x);

  printf("%d\n",\*ptr);

  x=x+5;

  printf("%d\n",x);

  printf("%d\n",\*ptr);

   (\*ptr)++;

   printf("%d\n",x);

    printf("%d\n",\*ptr);

  return 0;

}

**Ptr stores address of x[ptr=&x;] where as \*ptr gives its value**

**Thus if we print \*ptr it will give value of x=5 or 0+5=5 or (\*ptr)++**

**It will increment the value as 6…**

**Difference of call by value and call by reference are given below**

#include <stdio.h>

#include <math.h>

int swap(int\* a,int\* b);

int \_swap(int a, int b);

int main()

{

  int a=3,b=5;

  \_swap(a,b);

  printf("a=%d and b=%d ",a,b);

  swap(&a,&b);

 printf("a=%d and b=%d ",a,b);

  return 0;

}

int \_swap(int x,int y)

{

 int temp=x;

 x=y;

 y=temp;

 printf("a=%d and b=%d ",x,y);

}

 int swap(int\* x,int\* y)

 {

 int temp=\*x;

 \*x=\*y;

 \*y=temp;

 printf(" swappped values are a=%d and b=%d ",\*x,\*y);

 }

**\_swap is a function by call by value**

**Swap is a function by call by reference….**

**Call by reference will change value of variables in main function**

**but call by value will not…**

**pointers are used for returning multiple values through a function**

**if we pass address of a variable, we can store values in address and return multiple values from a function…**

**example**

#include <stdio.h>

#include <math.h>

int doWork(int a, int b,int\* sum,int\* product,int\* average);

//sum ,production and average function

int main()

{

  int x=2,y=5;

  int sum,product,average;

  doWork(x,y,&sum,&product,&average);

  printf("sum=%d\n product=%d\n average=%d\n",sum,product,average);

  return 0;

}

int doWork(int a, int b,int\* sum,int\* product,int\* average)

{

  \*sum= a+b;

  \*product=a\*b;

  \*average=(a+b)/2;

  return 1;

}

**Array is also a pointer**

#include <stdio.h>

#include <math.h>

int main()

{

int aadhar[5];

int \*ptr=&aadhar[0];

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

{

 printf("%d index:",i);

 scanf("%d",ptr+i);

}

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

{printf("values are%d\n",\*(ptr+i));

}

  return 0;

}

**This is an example program to show an array works like a pointer**

#include <stdio.h>

#include <math.h>

void reverseArray(int arr[], int num);

int main()

{

  int array[] = {1, 2, 3, 4, 5, 6};

  int length = 6;

  reverseArray(array, length);

  for (int i = 0; i < length; i++)

  {

    printf("%d", array[i]);

  }

  return 0;

}

void reverseArray(int arr[], int num)

{

  for (int i = 0; i < num / 2; i++)

  {

    int temp = arr[i];

    arr[i] = arr[num - i - 1];

    arr[num - i - 1] = temp;

  }

}

**In this function we are reversing an array using a function where it will reverse array and store values reverse in the same address.**

**We are printing the value in main function then also the value is printed in the reverse order..so this an example to show array as a pointer..**

**Two dimensional array**

**This is an example to store multiplication table of two numbers**

**In a loop we store multiplication tables of two numbers for example 2 and 3… using help of user defined functions…Then printing it in main function using loop {*note : here I got an error doing this program Was arr[][10] should be declared as arr[][10] instead of arr[][] else it shows errors like not compatable when we use single array it will act like a pointer else we should declare size*…}**

#include <stdio.h>

#include <math.h>

void multTable(int arr[][10], int num1, int num2);

int main()

{

  int array[2][10], num1, num2;

  multTable(array, 0, 1);

  for (int i = 1; i < 10; i++)

  {

    printf("%d\t", array[0][i]);

  }

  printf("\n");

  for (int i = 1; i <10; i++)

  {

    printf("%d\t", array[1][i]);

  }

  return 0;

}

void multTable(int arr[][10], int num1, int num2)

{

  for (int i = 0; i<10; i++)

  {

    arr[num1][i] = i \* 2;

    arr[num2][i] = i \* 3;

  }

}

**String in c programming**

**String are used to store words(i.e sentences) in memory. They are declared as**

**Char str[]={‘R’,’E’,’M’,’Y’,’\0’};**

**Or**

**Char str[]=”REMY”;**

#include <stdio.h>

#include <math.h>

void printName(char arr[]);

int main()

{

  char firstName[]={'R','E','M','Y','\0'};

  char lastName[]=" M Ali";

  printName(firstName);

  printName(lastName);

  return 0;

}

void printName(char arr[])

{

  for (int  i = 0; arr[i]!='\0'; i++)

  {

   printf("%c",arr[i]);

  }

}

**Scanf() has a drawback it cant read a full sentence but it read first word of a sentence.**

**So will see new function for string gets() puts()**

**But this gets() is out dated in c language so we use a function fgets(stringName,size,stdin)=> stdin here means standard input**

#include <stdio.h>

#include <math.h>

void printName(char arr[]);

int main()

{

  char fullName[100];

  printf("Enter your full name.");

  fgets(fullName,100,stdin);

  printf("Your full name is.");

  puts(fullName);

  return 0;

}

void printName(char arr[])

{

  printf("%s",arr);

}

**String also a pointer like Char \*ptrStr; ( Its value can be re initialise )**

**Example initially we are giving a value “Hello world ” in str(**

**char str[10];) its value cannot be re initialised with any other value or word like”Hello” will be showing error..**

#include <stdio.h>

#include <math.h>

int main()

{

  char \*ptrStr = "Hello world";

  puts(ptrStr);

  ptrStr = "world";

  puts(ptrStr);

  char str[] = "Hello world";

  puts(str);

  // str="world"; Here it will show error since it is a string not a pointer

  return 0;

}

**Different build in function by including header file**

**#include<math.h>**

**#include<string.h>**

**To count length of a word we can use a build in function**

1. **Int length=strlen(str);**

**We will get count of a word in a string array.**

1. **strcpy(str1,str2); to copy str2 to str1.**
2. **strcat(str1,str2); It will concatenate(join) the value of str1 and str2.**

**Here came to learn new thing like initialising a new variable in loop**

**Like => for(int i=0, int j=0 ; i<6 ; i++,j++ )h**

**By using a single for loop I can iterate through two different string array…**

#include <stdio.h>

#include <math.h>

void slice(char str[],int n,int m);

int main()

{

  char str[100];

  printf("Enter a sentence\t");

  fgets(str,100,stdin);

  slice(str,3,6);

  return 0;

}

 void slice(char str[],int n,int m)

 {

  char newStr[100];

  int j=0;

 for (int i = n ; i <= m; i++,j++)

 {

  newStr[j]=str[i];

 }

 newStr[j]='\0';

 printf("sliced portion %s",newStr);

 }

**Here I got one error while doing this program was like**

**I didn’t add null value(‘\0’) to newString after for loop in slice function**

* **It was because I was using %s in printf after that instead of %c**

**(format specifier was not able to identify it as string since it doesn’t had a null at end ,so if we are using %c we should iterate newString in a for loop for printing the string.It indicate advantage of %s(format specifier))….**

**Important \*\*\*\*\* please read carefully above program;;;;**

**Structures**

**Structures are used to store data of values with different data type,,**

**We can store int, char ,float in a continuous memory location by initialising at one time it can be used accessing the structure name example: struct student s1 and use each variable by using s1….**

#include <stdio.h>

#include <math.h>

#include <string.h>

struct student

{

  int rollNo;

  char name[100];

  float weight;

};

int main()

{

  struct student s1;

  strcpy(s1.name, "Remy");

  s1.rollNo = 13;

  s1.weight = 68;

  printf("Name %s\n", s1.name);

  printf("Roll No:%d\n", s1.rollNo);

  printf("Weight%f\n", s1.weight);

  return 0;

}

Struct student s1;

Is used to store single student data..We can use an array to declare n number of students like struct student s[100].

We Can access student 1 data by using s[0].name, s[0].rollNo, s[0].weight

Structures can be initialised directly => struct student s[0]={“Remy”,13,68};

Pointer can used with structures

#include <stdio.h>

#include <math.h>

struct student

{

  int rollNo;

  char name[100];

  float weight;

};

int main()

{

  struct student s1 = {13, "Remy", 68};

  struct student \*ptr;

  ptr = &s1;

  printf("Rollno=%d", (\*ptr).rollNo);

  return 0;

}

(\*ptr).rollNo can be used with arrow operators

like ptr-> rollNo it will be easier to use…..

Structures in function

We can pass structure defined variables through a functions

#include <stdio.h>

#include <math.h>

struct student

{

  int rollNo;

  char name[100];

  float weight;

};

void printStudentDetail(struct student s1);

int main()

{

  struct student s1 = {13, "Remy", 68};

  /\* struct student \*ptr;

  ptr = &s1; \*/

  printStudentDetail(s1);

  return 0;

}

void printStudentDetail(struct student S1 )

{

  printf("RollNo:%d\n",S1.rollNo);

  printf("Name:%s\n",S1.name);

  printf("Name:%f\n",S1.weight);

}

Void printStudentDetail(struct student s1); function declaration should be done after defining structure (struct student{char Name[100]};) as done above

*Typedefintion for structures*

**Structures key feature is putting niche name for defined name of structure ….**

#include <stdio.h>

#include <math.h>

// house no,block ,city ,state

typedef struct adress

{

  int houseNo;

  int block;

  char city[100];

  char state[100];

} ad;

void enterDtlAdress(ad \*adrs);

void PrintDetails(ad adr);

int main()

{

  ad owner[5];

  enterDtlAdress(owner);

  PrintDetails(owner[0]);

  PrintDetails(owner[1]);

  return 0;

}

void enterDtlAdress(ad \*adrs)

{

  for (int i = 0; i < 2; i++)

  {

    printf("Enter the house number:");

    scanf("%d", &adrs[i].houseNo);

    printf("Enter the block of 1:");

    scanf("%d", &adrs[i].block);

    printf("Enter the city of 1:");

    scanf("%s", &adrs[i].city);

    printf("Enter the state of 1:");

    scanf("%s", &adrs[i].state);

  }

}

void PrintDetails(ad adr)

{

  printf("Adress details \n");

  printf("House Number-%d\n", adr.houseNo);

  printf("Block Number-%d\n", adr.block);

  printf("City-%s\n", adr.city);

  printf("state-%s\n", adr.state);

}

**Struct address defined above is given a nick name as “ad”**

**By using typedefined we are using “ad”short name to create a new struct .In the above example we are entering details of a group of people by defining a struct called address and calling two function to enter details and print details..**

**Using example of struct we can understand more about structures**

**Go through**

#include <stdio.h>

#include <math.h>

struct vectors{

int x;

int y;

};

void calculateSum(struct vectors \*v1,struct vectors \*v2,struct vectors \*sum);

int main()

{

  struct vectors v1={ 2,3};

  struct vectors v2={ 5,8};

  struct vectors sum;

  calculateSum(&v1,&v2,&sum);

  printf("Sum of x vector =%d",sum.x);

  printf("sum of y vector=%d",sum.y);

  return 0;

}

void calculateSum(struct vectors \*v1,struct vectors \*v2,struct vectors \*sum)

{

  sum->x=v1->x+v2->x;

  sum->y=v1->y+v2->y;

}

**Volatile memory RAM🡺 memory get erased when power fails**

**Non volatile memory hard disk🡺it always stores if we save.**

**File**

**We can store files in system and we will access it with code as given below**

**We will store 1 2 3 4 5 in a txt file ….**

**Then read it with our code 🡺 here fopen is used to open the file test.txt(file name, read ‘r’ or write ‘w’ mode) and fscanf is used to read the file and store data in &num memory location .we print details using printf .**

**At end we should close the file we have opened.. using fclose(fptr);**

#include <stdio.h>

#include <math.h>

int main()

{

FILE \*fptr;

fptr=fopen("test.txt","r");

  int num;

fscanf(fptr,"%d",&num);

printf("number =%d\n",num);

fscanf(fptr,"%d",&num);

printf("number =%d\n",num);

fscanf(fptr,"%d",&num);

printf("number =%d\n",num);

fscanf(fptr,"%d",&num);

printf("number =%d\n",num);

fscanf(fptr,"%d",&num);

printf("number =%d\n",num);

fclose(fptr);

  return 0;

}

**In the same way create a new text file by entering code in write mode in this file is created automatically…**

#include <stdio.h>

#include <math.h>

int main()

{

FILE \*fptr;

fptr=fopen("student.txt","w");

 char name[100];

 int  age;

 float markPrcnt;

printf("Enter the student name");

scanf("%s",&name);

printf("Enter the student age");

scanf("%d",&age);

printf("Enter the student mark percentage");

scanf("%f",&markPrcnt);

fprintf(fptr,"Student name=%s\n",name);

fprintf(fptr,"Student age=%d\n",age);

fprintf(fptr,"Student mark prct=%f\n",markPrcnt);

fclose(fptr);

  return 0;

}

**Dynamic memory allocation**

**+++++++++++++++++++++++++++**

**It is allocating memory during runtime of program ,Eg: if are entering details of school student details in an array and array is full and in between the academic year we need to enrols 3 more children then memory size should be changed.. so we use dynamic memory….**

**We use some code like malloc() calloc() free ()realloc()**

**Reference:** [**https://www.youtube.com/watch?v=irqbmMNs2Bo**](https://www.youtube.com/watch?v=irqbmMNs2Bo)

**(Appna college youtube)**