1. #include<stdio.h>

#define n 255;

int my\_strncpy();

int main()

{

int my\_strncpy();

return 0;

}

int my\_strncpy()

{

char source[n];

char i;

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

{

printf(“Enter the source string”, source[i]);

scanf(“%s”,&source[i]);

}

d0

{

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

{

If(source[i]<=n)

{

char \*my\_strncpy(char \*destination, const char \*source, int n);

printf(“The Destination string after copy from source string%s”,source[i]);

}

}

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

{

If(strln(source[i])<n)

{

printf(“Destination string is padded with NULL characters”, source[i]);

char \*ptr=NULL;

source[i]=+ptr;

}

}

}while(source[i]=n);

return \*destination;

}

1. #include<stdio.h>

#define n 255;

int my\_strncpy();

int main()

{

int my\_strncpy();

return 0;

}

int my\_strncpy()

{

char \*source;

char i;

int n;

printf(“Enter the source string”, source);

scanf(“%s”,\*source);

source++;

d0

{

If(\*source<=n)

{

char \*my\_strncpy(char \*destination, const char \*source, int n);

printf(“The Destination string after copy from source string%s”,source);

source++;

}

If(strln(source)<n)

{

printf(“Destination string is padded with NULL characters”, source[i]);

char \*ptr=NULL;

\*source=+ptr;

source++;

}

}while(\*source=n);

return \*destination;

}

1. #include<stdio.h>

int main()

{

char \*input;

d0{

printf(“Enter input string”);

scanf(“%s”,&input);

printf(“Number of characters in the string is %d”, sizeof(input));

string++;

}while(strlen(input==255);

return 0;

}

1. #include<stdio.h>

int insertPosN(struct node \*\*pHead, int n>0, char \*newData);

Int main()

{

struct node{

char data[100];

struct node \*pNext;

};

int insert();

return 0;

}

int insertPosN(struct node \*\*pHead, int n>0, char \*newData);

{

printf(“Singly Linked List:”);

char node;

int n>0;

d0{

printf(“Insert node into singly linked list”);

scanf(“%c”,&newData->node);

node=malloc(n\*sizeof(char));

int m=0;

printf(“The node %c inserted at the position %d”,node, m);

return 1;

m++;

}while(m<n);

If(node==0)

{

return 0;

}

else

{

return -1;

}

}

1. #include<stdio.h>

int findPos(struct node \*pHead, char fData);

int main()

{

struct node{

char data;

struct node \*pNext;

};

int findPos();

return 0;

}

int findPos(struct node \*pHead, char fData)

{

printf(“Singly Linked List:”);

char node;

int n>0;

d0{

printf(“Insert node into singly linked list”);

scanf(“%c”,&fData->node);

int m=0;

printf(“The node %c inserted at the position %d”,node, m);

return 1;

m++;

}while(fData!=NULL);

return -1;

}

1. Constructor and Destructor both are the special member function in c++ programming language. Difference is constructor involved in memory allocation of variable or function. Destructor involved in memory deallocation of variable or function.
2. Class is used to bind data together in c++ programming language and object is the variable of class. Memory doesn’t allocated when class is created, but Memory is allocated when the object is created.
3. I have added the following programs on my github account
4. **To add two integer numbers:**

#include<stdio.h>

int main()

{

int a=0,b=0,c=0;

printf(“Enter the values of a and b”);

scanf(“%d,%d”,&a,&b);

c=a+b;

printf(“Sum of two this %d %d integers is %d”,a, b, c);

return 0;

}

**Summary:**

\* #include is a pre-processor directive which includes the both two header files i.e System header file(<headerfile.h>) and User defined header file(“headerfile.h”)

\*<stdio.h> is a system header file . It helps to standard IO stream.

\*int main() is a main function which returns 0. Because the return 0 indicates function termination.

\*int a=0; b=0;c=0; is a variable definition with initialization. It is good program practice to initialize the variable. Without initialization may give the garbage value. Initialization of variable with various data types as follows: int 0, float 0, double 0,char ‘\’, string”\”, pointer NULL. Here, variable definition defines the size, type as well as the space reservation

\*printf() function used to display

\*scanf() function used to get the return and display on output screen

\* ; is a statement terminator

\* if we enter the character or string instead of integer in values of a and b, we can get the garbage value

\* we can create this program function call like int add (int a, int b) but here, we must declare this function before main function as pre-processor directive or define this function before main function definition. Otherwise it will produce the warning as this function is undefined.

1. **To print odd or even numbers** :

#include<stdio.h>

int main()

{

int a=0;

printf(“Enter the integer value”);

scanf(“%d”, &a);

if(a%2==0)

{

printf(“The Entered Number is Even Number”);

}

else()

{

printf(“The Entered Number is Odd Number”);

}

return 0;

}

**Summary:**

\* #include is a pre-processor directive which includes the both two header files i.e System header file(<headerfile.h>) and User defined header file(“headerfile.h”)

\*<stdio.h> is a system header file . It helps to standard IO stream.

\*int main() is a main function which returns 0. Because the return 0 indicates function termination. If main function is void, there is no need to write return 0.

\*int a=0; is a variable definition with initialization. It is good program practice to initialize the variable. Without initialization may give the garbage value. Initialization of variable with various data types as follows: int 0, float 0, double 0,char ‘\’, string”\”, pointer NULL. Here, variable definition defines the size, type as well as the space reservation. In a C program. we can declare a variable or function many time but can define variable or function only one time.

\*printf() function used to display

\*scanf() function used to get the return and display on output screen

\* ; is a statement terminator

\* if we enter the character or string instead of integer in values of a , we can get the garbage value

\* if(a%2==0) checks the value of a by modulo operator, if it gives the remainder as zero, it returns the entered number is even. Otherwise it returns the entered number is odd.

1. **To add even numbers only**

#include<stdio.h>

int main()

{

int a[6]={19, 4, 2, 21, 26, 5};

int Even\_sum=0;

for(i=0;i<6;i++)

{

printf(“The values in an Array”, &a[i]);

}

for(i=0;i<6;i++)

{

if(a[i]%2==0)

{

Even\_sum=Even\_sum+a[i];

printf(“The sum of Even Numbers%d”, Even\_sum);

}

else

{

exit 0;

}

}

return 0;

}

**Summary:**

\* #include is a pre-processor directive which includes the both two header files i.e System header file(<headerfile.h>) and User defined header file(“headerfile.h”)

\*<stdio.h> is a system header file . It helps to standard IO stream.

\*int main() is a main function which returns 0. Because the return 0 indicates function termination. If main function is void, there is no need to write return 0.

\* int a[6] is definition of an array. Array is used to store set of variables in similar data type. Here, 6 represents the size of array

\*int Even\_sum=0 is a variable definition with zero initialization.

\*for(i=0;i<6;i++) is a loop statement. It have initialization, condition and incrementation. Here, instead of 6, we can’t put “a”. Because array provides garbage value in this for loop condition.

\*if(a%2==0) helps to find the even numbers

\*Even\_sum=Even\_sum+a[i] provides the sum of even numbers

\*else function retruns zero for odd numbers

\*; is a statement terminator

**4. Swapping of two numbers**

#include<stdio.h>

#include<math.h>

int main()

{

int a=0,b=0;

printf(“Enter the values of a and b”);

scanf(“%d,%d”,&a,&b);

a=a+b;

b=a-b;

a=a-b;

printf(“The values of a and b after swapping%d,%d”, a, b);

return 0;

}

**Summary**

\* #include is a pre-processor directive which includes the both two header files i.e System header file(<headerfile.h>) and User defined header file(“headerfile.h”)

\*<stdio.h> is a system header file . It helps to standard IO stream.

\*<math.h> defines the mathematical operation

\*int main() is a main function which returns 0. Because the return 0 indicates function termination. If main function is void, there is no need to write return 0.

\*int a=0; b=0;c=0; is a variable definition with initialization. It is good program practice to initialize the variable. Without initialization may give the garbage value. Initialization of variable with various data types as follows: int 0, float 0, double 0,char ‘\’, string”\”, pointer NULL. Here, variable definition defines the size, type as well as the space reservation

\*printf() function used to display

\*scanf() function used to get the return and display on output screen

\* ; is a statement terminator

\* if we enter the character or string instead of integer in values of a and b, we can get the garbage value

\*a=a+b;b=a-b;a=a-b; s the mathematical expression to swap two variables.

\* we can create this program function call like int add (int a, int b) but here, we must declare this function before main function as pre-processor directive or define this function before main function definition. Otherwise it will produce the warning as this function is undefined.

**5. To display the address and value of variable using pointer**

#include<stdio.h>

int main()

{

int \*ptr, var;

printf(“Enter the value of ptr”);

scanf(“&d”, \*ptr);

ptr=&var;

printf(“The value of ptr is %d”, \*ptr);

printf(“The address of variable is %d”, ptr);

return 0;

}

**Summary**

\* #include is a pre-processor directive which includes the both two header files i.e System header file(<headerfile.h>) and User defined header file(“headerfile.h”)

\*<stdio.h> is a system header file . It helps to standard IO stream.

\*int main() is a main function which returns 0. Because the return 0 indicates function termination. If main function is void, there is no need to write return 0.

\* int \*ptr, var is a variable definition with pointer. Here, pointer(\*) holds the address of another variable and returns the actual value for variable. Compared to array , pointer is a effective way of defining variable, because it doesn’t preallocate memory .Thus, it instantly allocate memory for each variable .So, there no extra space is needed.

\*Here, \*ptr returns the actual value of variable and ptr returns the address of variable.

9. First I click on + simple on the right corner of github webpage. Then I click on create new repository icon and I give a name to my new repository the summit or confirm icon to create new repository

10. \* Initially I browse” git create branch” on vs editor

\* Then I write new branch name (Ex: swapping algorithm)

\* Then press enter icon

\* The vs editor automatically change into my new branch then I write a program on vs editor and save it (Ex.swappinga.c)

\* On Terminal I will type git init

\* Then I write git push set—origin--upstream swapping algorithm

\* Then I add this program as git add swapping.c

\* Then I write git commit –m “Added swapping Program”

\* Then I write git push

\* Thus I have my new program on my new branch

\* I can see this on github while refreshing github

\* Then I click on new branch and select new pull request icon and select master branch to receive this pull request

\* After this I refresh again github, then I have a new pull request from my new branch(swapping algorithm) to master branch.

\* Then I click on merge pull request icon

\* So, now I merge changes from one branch to other.

11. \* If we have any changes in program, first change then save this program and then move this program into stage changes by pressing stage changes icon on right corner of vs text editor.

\* Then go to terminal of vs editor

\* Type git init

\*Then type git add program name(Ex: swappingb.c)

\*Then type git commit –m “Added swapping.c Program”

\*Then type git push

\*Now we successfully submit changes to github

12. http is the Hyper Text Transfer Protocol . Application layer contains http . It contains the data to transfer. But, I don’t know the different http status code

13. I don’t know the linux command line