Pointer + Fun^{n} + Recursion

DAY-1

STL

Date / /
Page No.

DSA

Standard Library

DSA

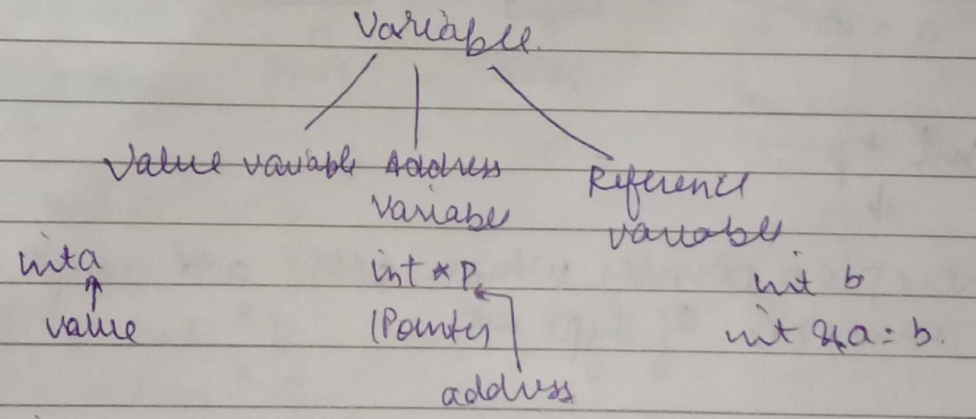
- Array
- Linkedlist
- Tree
- Graph
- Stack
- Queue

Dynamic Programm
Divide & Concer
Greedy Approach

1) Array

Pointer:-

Variable - memory location name h jl a value store krate h



Pointer:

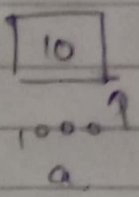
- It is an address variable which is used to store the address of another variable of same type.
- Pointer always store an address of a variable
- In main memory is byte addressable & every byte have an address of integer type

Integers → Decimal
→ Hexadecimal
→ Octal.

⇒ int a = 10; } Initialization

value

declaration



printf("%d", a); 10.

printf("%u", &a)

1000

⇒ int a = 10;

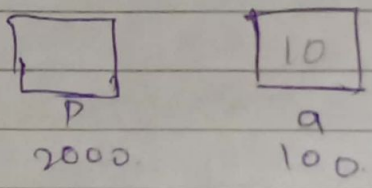
int b b;

b = &a ; X

because b is value variable not address variable

⇒ int a

a = 10.



int *p;

↓

we call here p is a pointer which store address of a variable of type int

int main()

{

int a = 100;

int *p

p = &a

printf("%d\t%d\n", a, p);

printf("%d\t%d\n", &a, *p);

*p = 200.

printf("%d\t%d\n", a, *p);

}

Date / /
Page No.

# int main ()
{
int *ip; → here ip is pointer which store address of integer type variable
char *cp;
float *fp;
double *dp;

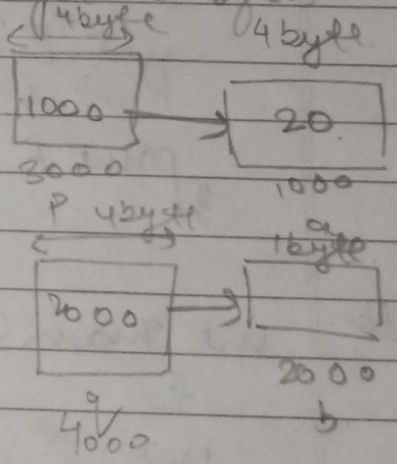
Print f("%d \n", size of (ip));
" " " (cp));
" " " (fp));
" " " (dp));

size of pointer is depend system bytes

# int main ()
{
int a = 20
char b = 'A';
int *p;
char *q;

p = &a; if p = &b then error
int* char*

q = &b;
}



Date
Page No.

# int main()
{
int *p = 1000;
int * ≠ int value

int *p = (int *) 1000;
printf("%d\t", p) → 1000
printf("%d\t", *p); → runtime error (*p doesn't denotes to particular value)
printf("Hello class\n");
}
Pro terminate

# int main()
{
int *a;
printf("%d\n", a); runtime error because p can't to have value
printf("%d\n", *p);
}

# int main()
{
int a = 357;
int *p;
char *q;

p = &a;
q = &a

}
q = (char *) &a;
printf("%d\t%d", p, q) o/p 1000, 1000

printf("%d %d", *p, *q);

O/P 357 191

(Undefined for system)

# int main()
{
const int a
a = 100;
printf("%d", a);
}

Error coz a have garbage & const can't be modify

# int main()
{
const int a = 100;
const int *p;
p = &a

*p = 200;

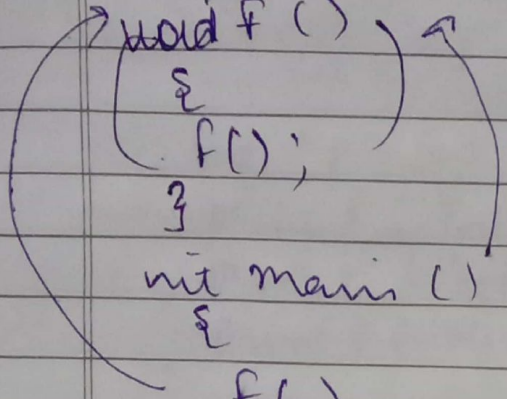
printf("%d %d\n", a, *p);

# int main()
{
int a = 100;
int b = 200;

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

Date / /
Page No.

Recursion → It is a process defining a fun^n in self similar way when we call



```
void f ()
 {
  f();
 }
int main ()
 {
  f()
 }
```

function ko fun^n ki defn & call kia

#
~~#include~~ ~~int main~~

```
void f ()
 {
  printf (" Hello \t ");
  f ()
 }
int main ()
 {
    f();
 }
```

O/P Hello ∞times

#

```
void f (int n)
 {
  if (n==0)
   return;
  printf ("Hello \t");
    f (n-1);
 }
```

```
int main ()
{
   f(5)
}
```

O/P hello (5 time)

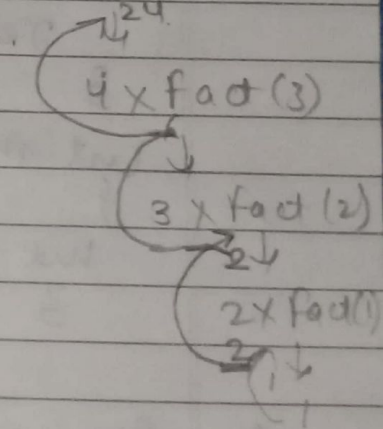
have

* Recursion make less code of line but complexity is higher.

** Factorial s recursive code

```
void int fact (int n)
{
   if (n == 1)
      return 1;

   return n * fact (n-1)
}
int main ()
{
 int f = fact (5);
}
```

f = fact 5
= 5 x fact(4)
= 120.

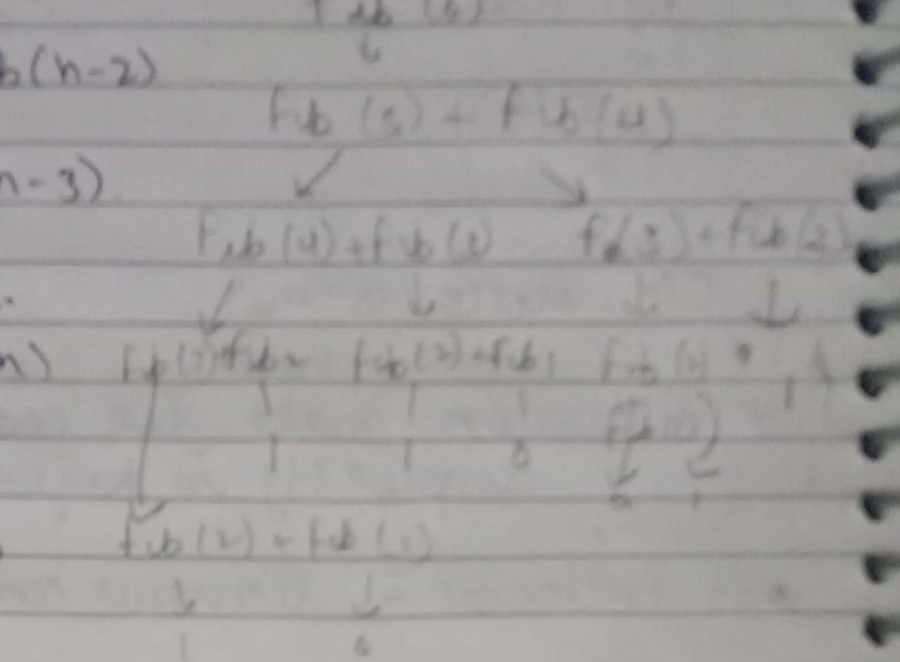


Fibonnaci Series

0 1 → 1 2 → 3 5 → 8 13 → 21 34 → 55 ... or

$n^{th} = (n-1)^{th} + (n-2)^{th}$

f(n) →
fib(n-1) + fib(n-2)
↓
fib(n-2) + fib(n-3)




```
int fib(int n)
{
  if (n==1)
    return 0;

  if (n==2)
    return 1;

  return fib(n-1) + fib(n-2);
}
int main()
{
  int f = fib(5);
}
```

* Recursion is slower than looping due to high complexity

* void f(int n)
{
  if (n==0)
    return;

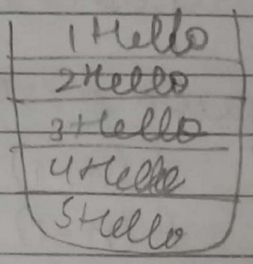
  f(n-1) → Recursive function
  printf("%d Hello\n", n);

Date / /
Page No.

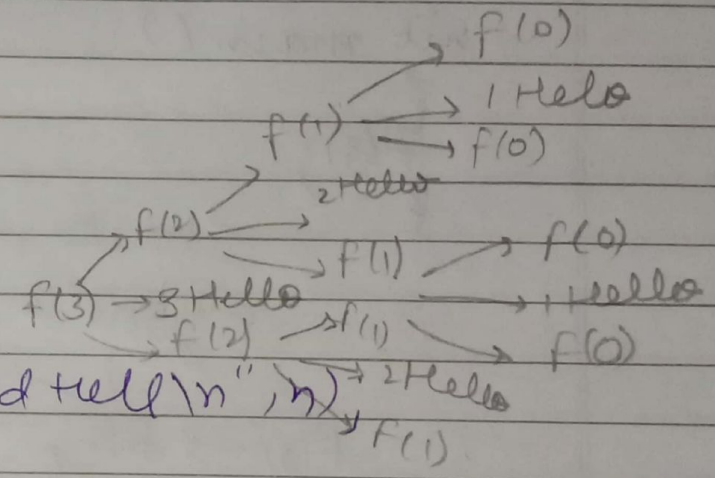
Recursive Statement flow using stack

```
}
int main()
{
  f(5);
}
```

f(5) → f(4) → f(3) → f(2) → f(1) → f(0)



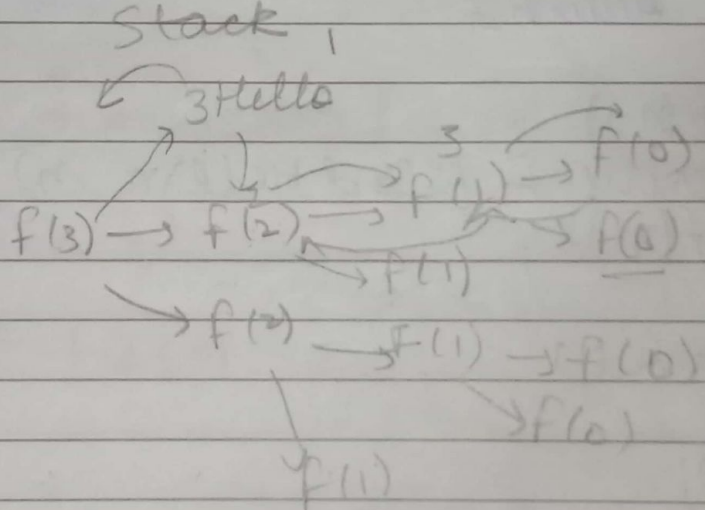

```
void f(int n)
{
  if (n==0)
    return;
  f(n-1);
  printf("%d Hello\n", n);
  f(n-1);
}
int main()
{
  f(3);
}
```



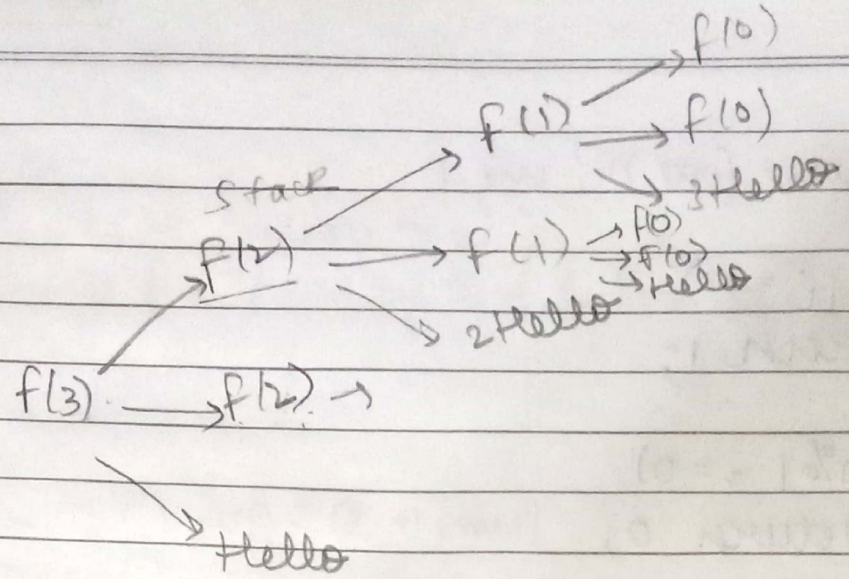
O/P 1 Hello
2 "
1 "
3 "
1 "
2 "
1 "


```
void f(int n)
{
  if (n==0)
  return 0;

  printf("%d Hello \n", n);
    f(n-1)
    f(n-1);
}
int main()
{
 f(3);
}
```

next Ques.



3 Hello
2 Hello
1 Hello
1 Hello
2 Hello
1 Hello
1 Hello



1 hello
1 hello
2 hello
1 hello
1 hello
2 hello
3 hello

Recursion

```
void f (int n)
{
    int i=0;
    if (n==0)
    return;
    printf("%d\n", ++i);
    f(n-1)
}
int main()
{
    f(3)
}
```

global

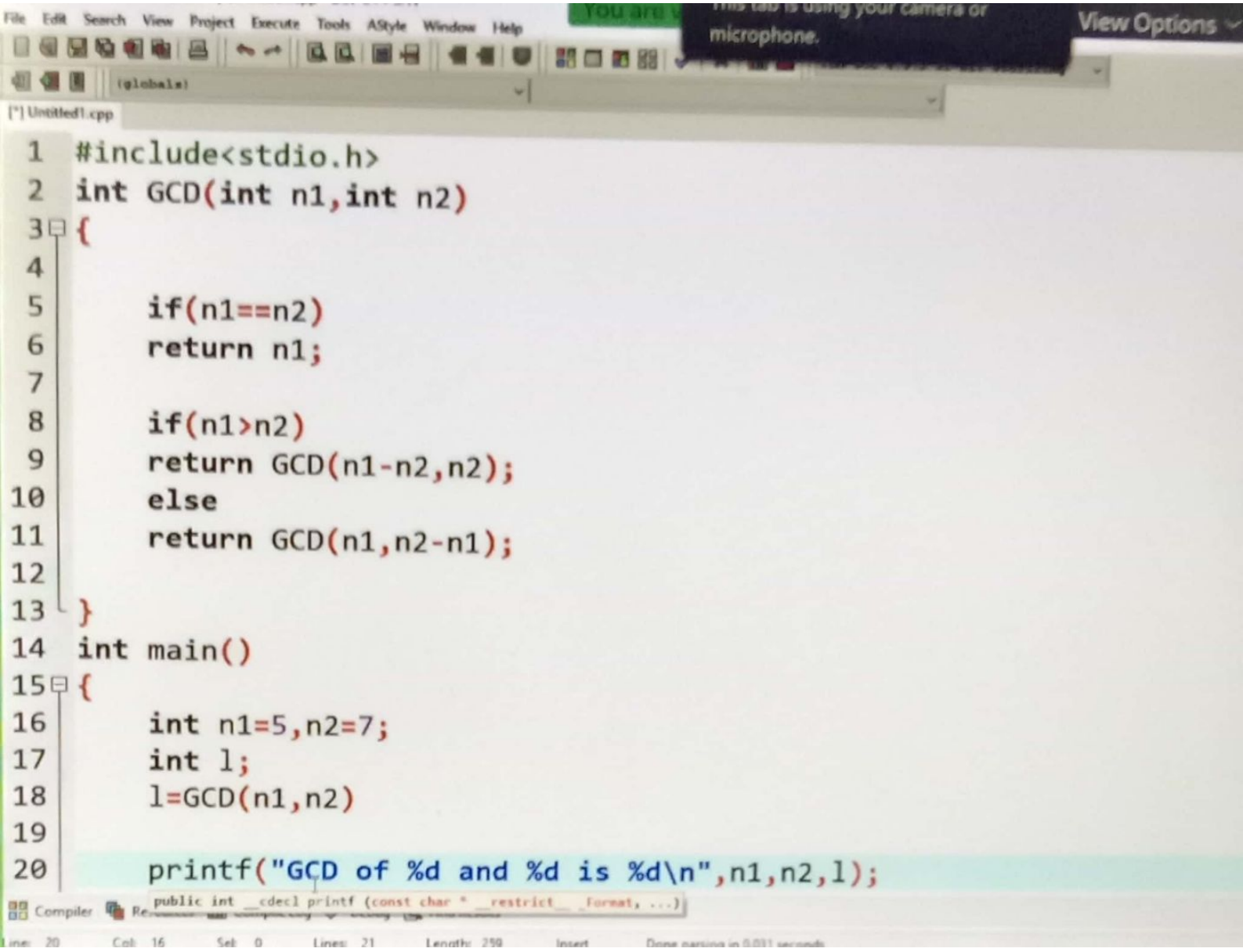
0 | P ⇒ 1 1 1 1

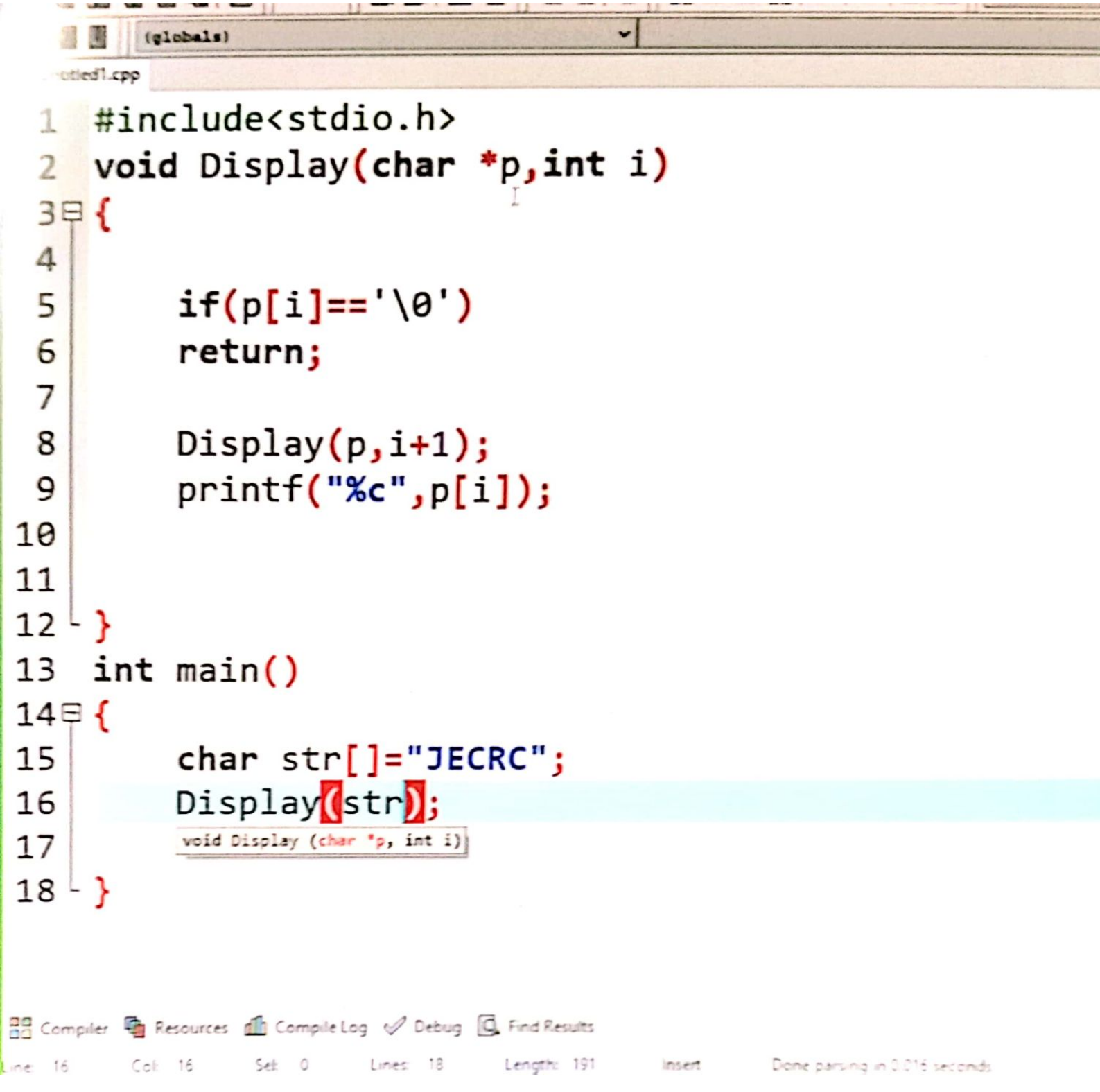
n ⇒ 1 2 3

Date
Page No.

```
int
# void Prime (int n, int i
   {
   if (i==1)
   return 1;

   if (n%i ==0)
      return 0;   Prime(n,5) → Prime(11,4) → Prime
                                (11,3) → Prime
returnPrime (n, i-1);      (11,2) → Prime(11,1)
 }

int main
 {
  int n;
  Prime (n, n/2)
 }

# int LCM (int n1, int n2, int max
  {
   if (max % n1 == 0) && (max % n2 == 0)
  return max;
   return LCM (n1, n2, max+n1);
 }


int main ()
   {
   int n1 = 5, n2 = 7
   int l;
   if (n1 > n2)
   l = LCM (n1, n2, n1)
```

```
else
l = LCM (n2, n1, n2);
printf("LCM of %d & %d is %d\n", n1, n2, l);
```


File Edit Search View Project Execute Tools AStyle Window Help
microphone.
View Options
(globals)
[*] Untitled1.cpp
#include<stdio.h>
int GCD(int n1,int n2)
{
if(n1==n2)
return n1;
if(n1>n2)
return GCD(n1-n2,n2);
else
return GCD(n1,n2-n1);
}
int main()
{
int n1=5,n2=7;
int l;
l=GCD(n1,n2)
printf("GCD of %d and %d is %d\n",n1,n2,l);
public int __cdecl printf (const char * __restrict__ _Format, ...)
Compiler


(globals)
Untitled1.cpp
#include<stdio.h>
void Display(char *p,int i)
{
if(p[i]=='\0')
return;
Display(p,i+1);
printf("%c",p[i]);
}
int main()
{
char str[]="JECRC";
Display(str);
void Display (char *p, int i)
}
Compiler
Resources
Compile Log
Debug
Find Results
Line: 16 Col: 16 Sel: 0 Lines: 18 Length: 191 Insert Done parsing in 0.016 seconds