preprocessor

gcc
general c - compiler

Doc.c → oos.i → Doc.ass → Doc.obj
Hard disk.

07-01-2020 linke → linking

②

Text
Datat
Symbol Table

Text → progrr actual code
bss not initi ly
Data ← global not bss (Data) initia comm
Symbol Table ← local var

Compiler — line to line read whole program

③

Primary header
Data Text
Data
Symbol tl

Primary header ← word ① O.S of magic number
W → MZ - mark Zukirk
→ P.E = portable Executable

linux → elf executable linkebal format

② Entry print funt"
③ Time state stock.

java → cafe Babe
mac → feed face

mikek
Bcells combine programin Languag (BCPL)

06-01-2020

1965 → ET & T Bell → Brain kernighan
general electronics ken thomson B
MIT → - Denis Ritchi

uniprocessing company hoti

3 mem. multics → PI/I 90% Assempl 10%. ATMJ

name of company

BCPL basic compile programming language
Denis Ritchi 70-72

1970
Chics -
Denis ritch
Brian karnighan
ken thompson

C11
C18

ANSI = American Nation Standart Institud (C88)

ISO = Internation organization of Standart C89

1967 band pallia

stand input / out file

\# include< stdio.h > ← declaration

call
defination
declaration

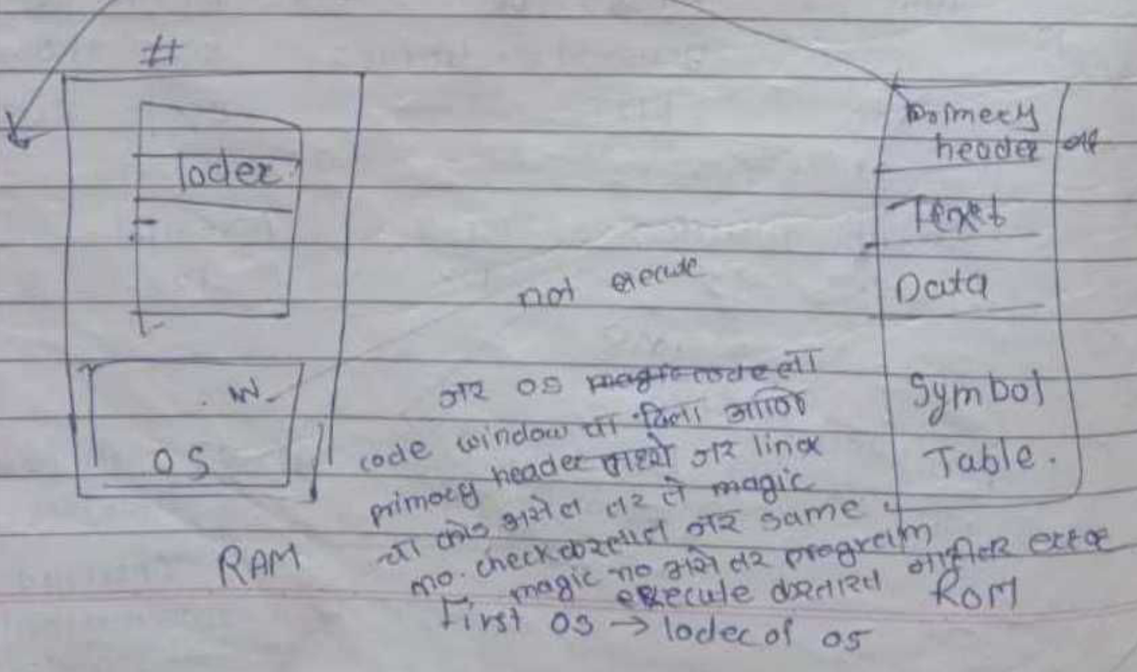
declrahre
kiran (10)

kiran (int. Rs) ← call function.

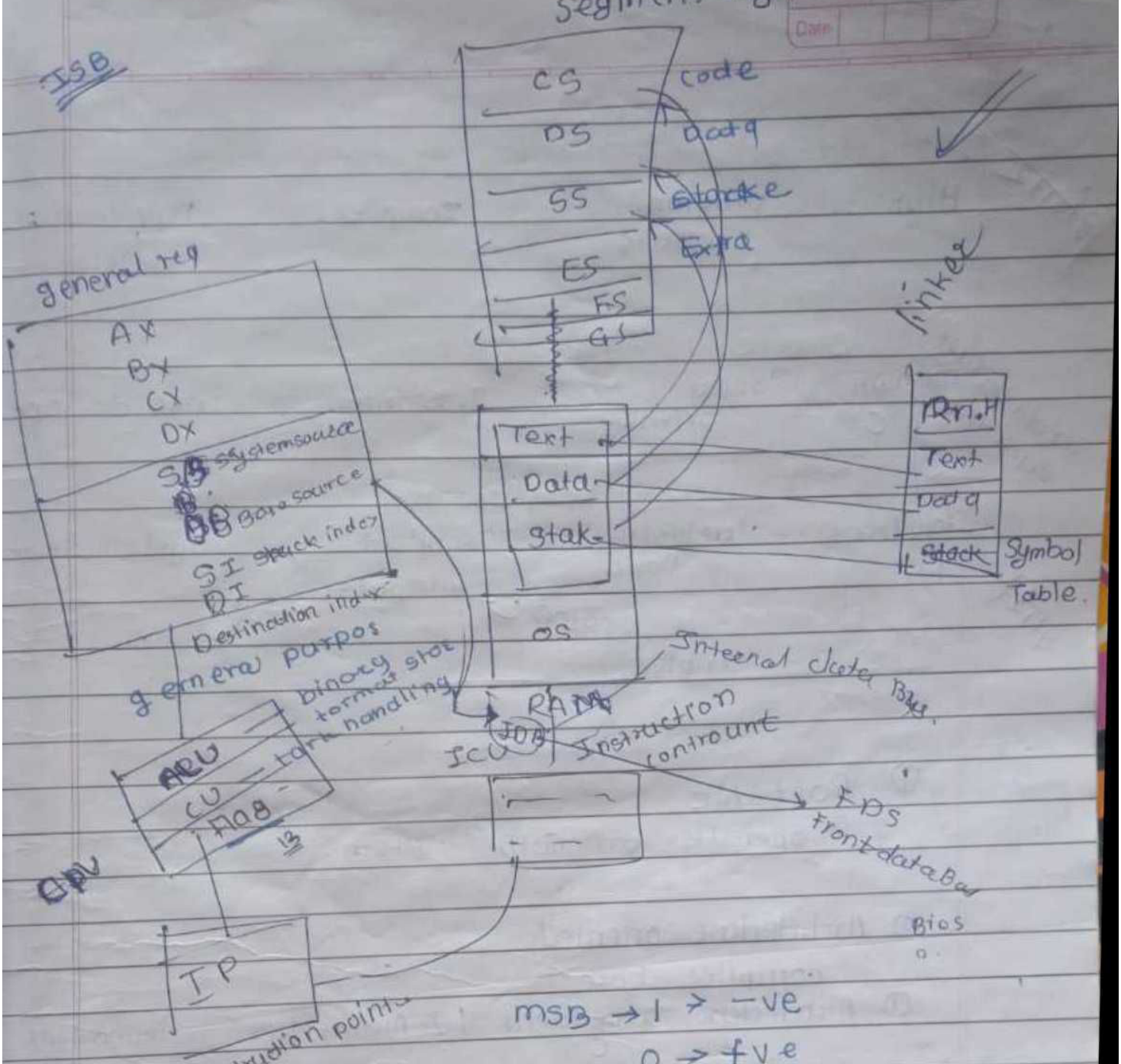
08-01-2020

compiler → read line to line whole program then convert ~~it into the~~ the Human understandeble code to machine dependent code

Interpriter- read a line convert it to machine dependent coden and then manchin independent code



ISB



MSB → 1 → −ve

0 → +ve

32 bit	64 bit	16 bit general register	8 bit	8 bit
DAx	RDx	EAx	AX	AH AL
DBx	RBx	EBx	BX	BH BL — 8 Bit
DCx	RCX	ECx	CX	CH CL
DDx	RDx	EDX	DX	DH DL

64 bit 32 Bit

16 16

9-1-2019

1st gen	High	program oriente	Compiler	Application developer
architecture dependent	medium	C (and C++)	object oriented	Interpriter	web develope
2nd gen	low level	virtual Basic	compilert interprit	System devel
	bes			
	assembly level	scripted	python ↗	

① Portable
operate on both system.

② Architecture oriented
compiler base
① Architecture dependent / ② Architecture Independent
C Java

③ Platform oriented - os - base
platform depented windo C
platform independent → java

progm
JVM (compax.
intel JVM

01-202

Stack / storage classes / variable. Iteratoe.

auto storage
static
extern
register.

(A) auto — (1) memoey allocation
(1) D-ds stack, dss, data, heap, RODATA, Register

void fun() → stack
↳ into → auto 3
o out memoey

• default storage class

(2) Default memoey
(Garbage, zero.)
into g

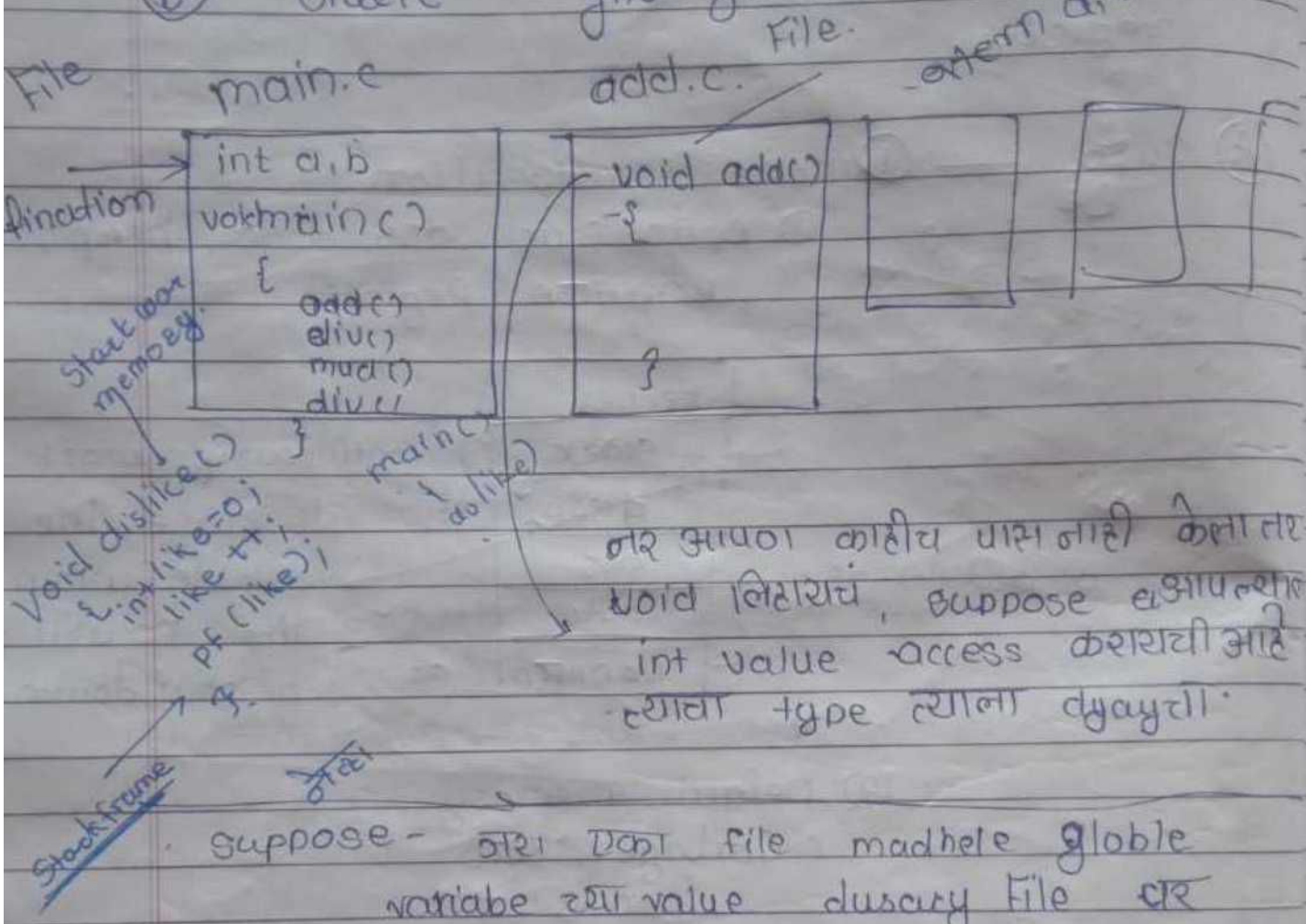
(3) lifetime (local, global)
(4) Scope. (global, local)

when lifetime is life globle the scope is also glable the canbe start at the same ~~os~~ or may be end at the same.

(5) linkage
(no, external, internal)

```
void fan()
{
   int a;
}
```

(B) Static चे global.



जर आपण काहीच पास नाही केलात तर void लिहायचं, suppose a आपल्याला int value access करायची आहे त्याचा type त्याला dyaych.

suppose - जर एका file madhele globle variabe चा value dusarya File वर declared करायचा असल्यास

e extern int a,b जर दुसऱ्या file वर लिहायचं.

suppose - आपल्याला त्या File madhale variable val protect करायचा असल्यास - पहिल्या या file चा variable la static as

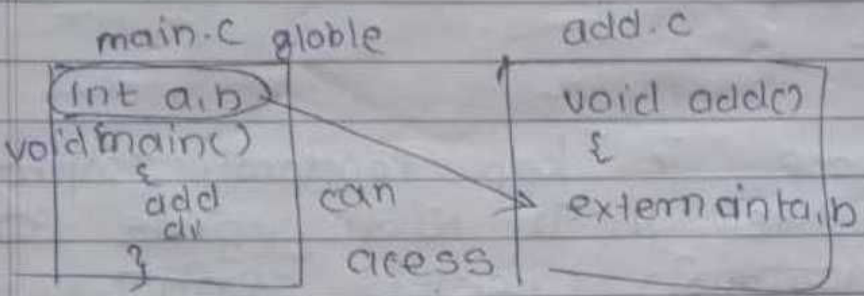
• static int a,b as declared करायचं

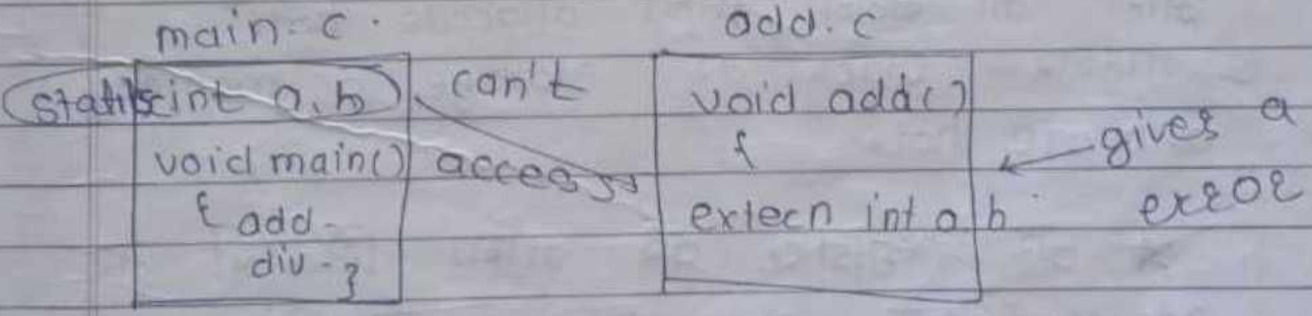
(c) extern

आपण एका file वरती var दुसऱ्या file वर नाही use करून शकत पण using extern आपण variable ~~assign~~ access करू शकतो

in a, b — defination

int a=10 → declaration / initiation.





Static → used for maintain the scope only for that file

1-19

bss - blog
stat symbol

- Static local

main.c

```
void main()
{
static int a,b
 like
}
```

— gives the random value

sap
1
2
3
4

suppose jar static nasel tr te
1 1 1 1 1 ashich value del.

② Register -

suppose je loop constantly reapeate hot asel te H तर त्याला register वर जागा देण्यासाठी register वापरतात

only local use

तो आधी o.s ला विचारलं जातं जागा आहे का मगच जागा allocate hoti नाहीतर stack वर storage or memory allocate hoti

compiler request करतो

★ जर register वर जागा मिळाली तर address नाही store करु शकत किंवा access नाही करु शकत.

जर H local memory असेल तर variable ला जागा stack var milate

जर global variable असेल तर त्याला data वर memmory मिळते.

Sometimes Variables ला can get the value 0. but the the Variable gets the memory in stack

Local → Stack
globle → data

29-01-2019

Basic

int → 2 short — 2 bit
int long → 4 bit
Long log long int → 8 bit.

Float - Float - 4 bit
double - 8 bit

char - 1 byte.

void -

Data can chang the size of of varible

signed → + aswell as negative
Unsigned → only positive.

derived *
function →
array →
Pointer →

User defined
Structure
union
Enum.

```
# include <stdio.h>

int main()
{
    const int a = 10
```

30-1-2020

Operators.

Depends upon the no of operands.

1) uniary → 1 operands
2) Binary → 2 operands
3) Ternary → 3 operands

-a +a
a a+b

- more than 2.

Perform the operation

1) Arithmatic
2) Relational
3) Bitwise
4) logical
5) Increment decrement
6) others
7) conditional
8) shortened
9) assignment

20 = a ⟶ not applicable

① Arithmatic operator
+, −, x, /, %, % ← remainder.

② Logical.
! - not
|| - an or ..one of them is true.
&& = and both condition true
if first is false the need not to check other condition

③ Bitwise

Not	~		and *		or +
or	|	0 0	0	0 0	0
and	&	0 1	0	0 1	1
XOR	^	1 0	0	1 0	1
shift.	>>	1 1	1	1 1	1
	<<				

xor

0 0	0
0 1	1
1 0	1
1 1	0

④ Relation
>
<
>= } equality
<=
!=
==

⑤ assignment
=
a = 20
a+ = 1
a = a+1

Shortened

a=+1

a+=1

a=a+1 / a=1+a

a+/1

a=a/1

others →

① , and sizeof().

in ( ) , is act as operator
so in ( ) when we put the
operat condition use the (;)

out of ( ) it act as seperator

Suppose

int a=(10,20,30,40) ← operator
o/p a=40

ina= 10,20,30,40 ← seperator

when

pf ("%d%d%d\n", a++, ++a, a++)
execute from
right → left ←

without pf execute from
→
left → right.

Preincrement and ~~po~~ decrement

03-02-2019

★ control statement

①
② selecting statement. if
③ Iterative statement
④ jumping statement

① if — if ( ) ← condition.

if ( )
↑ 1 non-zero value the execute the if statement.

always condition is not require

② if-else

if ( )→1 conditrue execute
{
}
else
{
}

otherwise execute false statement.

③ if — nexted if.

{ }
else if ( )
{ }
else if ( )
{ }
else

; ⇒ NULL statement

in the switch case we can not performed the all conditions in the as compared to the if-else()
but at
the time required to execution of ie if-else is more than the switch case
* If ii

(2) Switch case

```
Switch ( )
{
case 1 : -
    break
case2 : -
    break;
case3 :
    break;
default (    )
}
```

(3) for
if we know the start and end of the loop then we use the for loop

e.g Suppose आजीबांना शाळ जणायचीय तर त्यांना तर त्यात 108 मणी असतात प्रत्येक मणी एकदा तरी repete झाले पाहिजे त्यासाठी use for loop.

for (i<10 ; i++ ; i=50)

हे चालत

```
for (i= 1 ; i<=108 ; i++)
{
}
```

curly brack is ~~here~~ not necessary

Suppose आपल्याला 10 वेळा repeate माल जपायची आहे, परत 10 साठी for loop.

```
for (j=1 ; j<=10 ; j++)
{
    for (i=1 ; i<=108 ; i++)
    {
    }
}
```

* Suppose आपल्याला 3 time ला (~~sak~~ (morning, afternoon, ~~ani~~ evening) माल जपायचीआहे

```
for (k=0 ; k=3 ; k++)
{
    for(j=1 ; j<=10 ; i++) ;
    {
    } for (i=1 ; i<=10 ; i++) ;
    {
    }
}
}
```

④ while

when dont know the end of the condition where to end but the condition is Finite then use the while.

```
while( )   <- condition
  {


  }
```

⑤ do-while

```
do
 {
 write what we want to display
  prf( )
  scan( )
  swich (ch)
   {
    case 1:
    case 2
   }default
while(ch>3)
```

C

jumping - ① exit → out of program

② break → switch, for, while & do-while.
out of loop
↓
use in only loop

③ goto → use without any loop.
using lable

~~for~~

④ return –

int main
{
return(0);
}

return type & function type should be same

continue →
without perform the statement continue the loop.

④ continue
↑ fakt use only for zero

termination *

o/p of c compiler → An assembly
illegal instruction → no of memory
assembly → machine → assembly
(void) 0 → null
CPP → predefined

int man()
{ int i = 1024;
for(;i;i>>=1)
{ int x = 0;
int * ptr = &x
printf("%d", *ptr) }

18-05-2020

dma dynamic memoey allocation

★ Static memoey ⇒ compiler ला माहिती असतं आपण की किती memoey dy द्यायची आहे compiler ला compile time ला माहिती असतं किती memoey milnar आहे memoey runtimelach milte

e.g int a[10] ⇒ 10×4 = 40 byte memoey पाहिजे

हे compiler compile time la predict करतो

★ dynamic memoey ⇒ compiler ला माहिती नसतं compile time ला किती memoey milnar आहे तर runtime ला माहिती असतं तेव्हा मी memoey allocation

- predefined function predict करतं किती memoey द्यायची

① Static memoey allocation
② dynamic memoey =1—
③ automatic memoey allocati

- suppose function च्या आत एखादा variable define kela तर automatic memory milte आहे runtime

dynamic ⇒ memoey किती मीळणार हे नंतर ~~runtime~~ ला ठरवतो.

int a[3][4]

a is a 2-D array which contains the 3 1-D array each 1-D array contains the 4 element element of int type

compile ला compile time lach memory deto but after compile

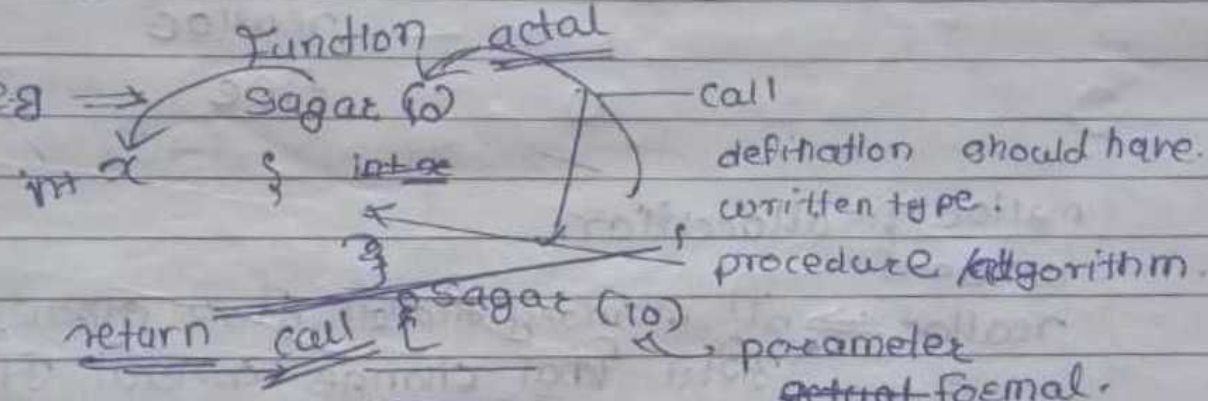
dynamic memory ap आपल्याला function मीळवून देत.

function s ⟶ go get memory calculation

def delation → Predefined : predefine.

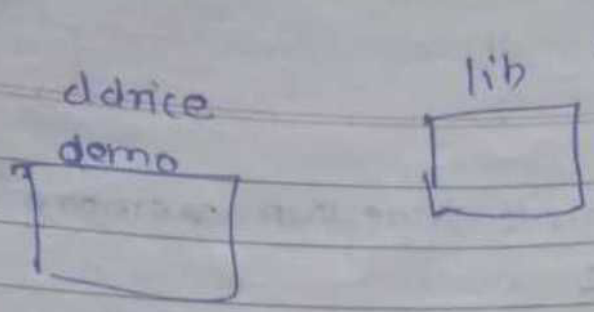
call redefine

① malloc → ( ) ← actual parameter
② calloc ⇒ ( )
③ realloc ⇒ ( )
④ Free



lib folder { malloc ( ) - actual parameter should given
calloc ( )
realloc ( )
free

call { } defination.



~~Predifine~~ function defination शोधण्यासाठी declation ~~is~~ लागतं

declaction
returntype Fun^{n}name ( );
↑ in
in header file
stdio.h
stdlib.h ← foe malloc
calloc
realloc
free

malloc } allocation
calloc }
realloc ⇒ जी memoey मीळाली तीजर वाढवायची असेल किंवा change करायची असेल तर त्यासाठी realloc वापरतात.

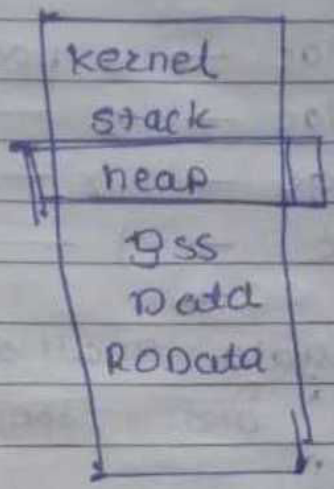
- realloc च्या आधी malloc किंवा calloc वापरलेला पाहिजे.

Fgree ⇒ memoey deallocation
melloc किंवा calloc वापरलेलं पाहिजे.

- dynamically alloca केलेली memoey deallocate करतो.

★ Syntax

Pololype ⇒ parameter, memor



- malloc ⇒ ; no of byte

  malloc (size_t) no

- calloc ⇒

  calloc (size_t ; size of (element));

  ↑ no of element

  ↙ 10 byte size allocate keli

  - malloc (10)

    byte

    calloc (10, size(8))

    8 byte ची 10 element

    10 x 8 ⇒ 80

- realloc ⇒

  realloc (void* ptr, size_t)

  ~~किती पहिजे तेवढं सांग~~

  ~~आधिचं किती करून ठेवायचं कमी जास्त~~

  ~~ते मी ठरवत.~~