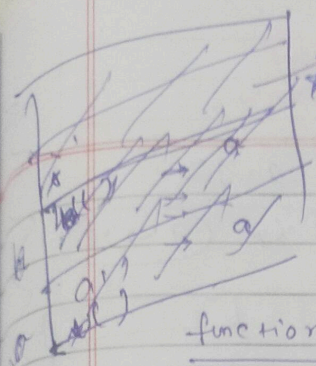
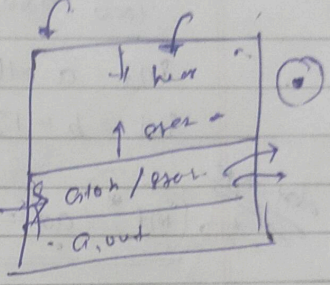


Sodeep

Segmentation error

function
- predefine
- user define

function
- returning type
- Argument type



function and declare —

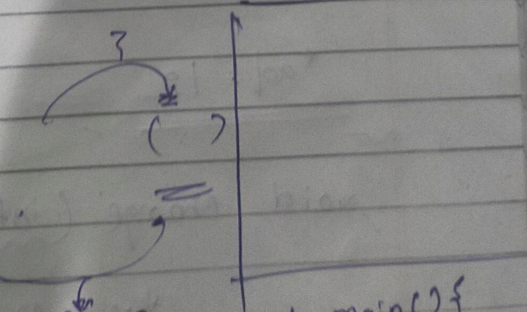
→ fun. header declaration

```
void b(){
    printf("b");
}
void main(){
    a();
    b();
}
void a(){
    printf("a");
    b();
}
```

```
#include<stdio.h>
void a();
void b();
void add(int a, int b);
void add(int, int);
void add(int a, int b);
```

```
void add(int a, int b)
{
    printf(a+b);
}
```

pass by value
pass by reference.




```
void change(int a){
    a = 0;
}
```

a — local

```
void main(){
    int a = 12;
    change(a);
    printf(a);
}
```

12

int void add

Pointer

Date
Page No.

(*)

a	b	ad
[rad]	[12]	[1000]
1000	2000	3000

int a = 10
int b = 12

data type → int* ad = &a

printf(ad)

int a = 12
↳ (a → 12)

pointer
↳ int*
↳ float*
↳ char*
↳ double*

(&) &var → address of var
(*) var → value at that addr

int* ad = &a

*(&a) = a

X int *ad = &a

*(&a)
*ad = a

*ad = 12

void change (int* x){
*x = 0
}
void main(){
int a = 12
change(&a)
}

x
[1000]
2000

[12 0]
1000

Print(a) → 0

Swaping

```
void swap(int a, int b){
    int temp = a;
    a = b;
    b = temp;
}
```

a ↔ *p
b ↔ *N

```
void swap(int* p, int* q1){
    int temp = *p;
    *p = *q1;
    *q1 = temp;
}
```

passing (&)

* add

	a	&a	&add
	10	1000	2000
	a 1000	add 2000	int * add
		int*	

swap(&a, &b)

int* sum

retn &a

Arrays

↳ continuous Memory allocation

4 4 4 (4 x 10)

1	2	3	0	0	0	0	0

1000 1004 1008 1012 1016

int arr[10];

int arr[10] = {1, 2, 3, 4, 5};

arr → 1000 (40)

base address

00 - 51

{ arr[0] = 1;
arr[1] = 2;

~~arr = {1, 2};~~

arr[0] = 0[arr]

*(arr+0) = *(0+arr)

arr[1]

*(arr+1)

Size

int s = sizeof(arr)

~~sizeof(int)~~

sizeof(int)

int s = sizeof arr / sizeof(arr[0])

arr+3

10	20	15	25	30
0	1	2	3	4

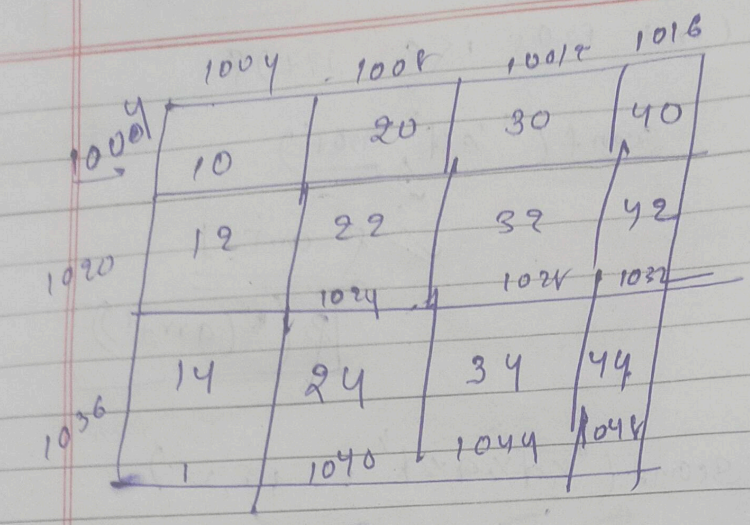
arr[3]+3

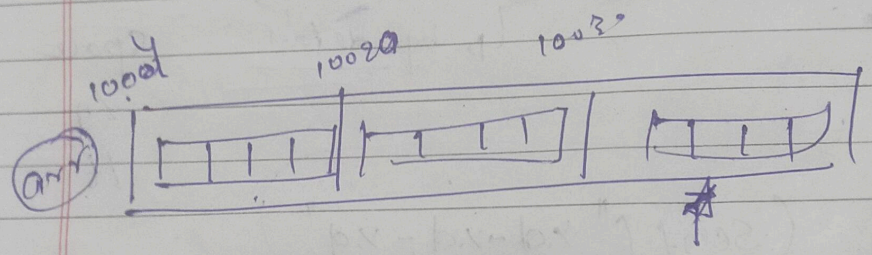
↳ 28

Date
Page No.

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

    scanf("%d", arr+i)
}
                // & arr[i]
                & (*(arr+i))


scanf("%d%d%d", ...)
    -> by default   space
                    \n
                    \t

scanf("%d,%d,%d"

                 int* arr
void reverse (int arr [ ] ) {

    int lb =0
    int ub = sizeof (arr) / sizeof(arr[0]) -1 ;

    while (lb < ub) {                      int temp = *(arr+lb)
        int temp = arr [lb]
        arr [lb] = a [ub]
}

void main () {

    int arr [5] = { 1, 2, 3, 4, 5 }

    ~~void re~~
    reverse ( arr )


*arr + 3
```

*arr di
arr[0]





(arr[0])[2] = 30

((arr+0)+2)

((arr+2)+3)

0	1	2	3	4	5	6	7	8	9
A	y	u	s	h	i	\0	\0	\0	\0

String

↳ Array of Character.

\0

char name[10];

char name[10] = {'A', 'y', 'u', 's', 'h', 'i'};

name[ ]

A	y	u	s	h	i	\0

Ayushi Pani <3

Date
Page No.

char name[] = "Ayushi";

char name[10];

(X) name = "Ayushi";

→ n

%.s

printf("%.s", name);

puts(s)

user input

Ayushi_pani

scanf("%s", arr);

scanf("%[^\n]s"

gets(arr);

fgets(arr, 10, stdin);

char name[]

char*

function

char* name

(I) strcat(arr, brr)

(II) strcpy(crr, arr)

strcpy(name, "Ayushi");

char crr[10] = "Ayu"

= "Ayushi"

name1 = Ayushi Sarkar
name2 = Ayushi pani

Date
Page No.

strlen(a)

name < name2

strncpy(arr, 10, bin)

-ve

0 → strcmp(name1, name2)

+ve

name1 > name.

+ve

Sudeepa Tinku

-2

+1 Shon Ka Sath Aya

+10

char* = strtok(name/adr, " ")

st = "The sky is Blue"

char* tok = strtok(st, " ")

print("%s", tok)

tok = strtok(NULL, " ")

→ sky

tok = strtok(NULL, " ")

→ is

tok = strtok(NULL, " ")

→ Blue

tok = str(Null, " ")

10

Date
Page No.

```
char* tok = strtok(str, " ");
while (tok != NULL){
    printf("%s", tok);
    tok = strtok(NULL, " ");
}
```

Structure

```
typedef struct person{
    int age;
    char name[10];
} pr;
```

pr a;

struct person a;

```
a.age = 56;
a.name = "
strcpy(a.name, "Rohit");
```

```
fgets(a.name, 10, stdin);
scanf("%d", &(a.age));
```

struct person* ad = &a;

```
(*ad).age
ad->age
```

```
&(ad->age)
fgets(ad->name, 10, a)
```

Date
Page No.

SMA
static mem all.

int a;

int* p;

DMA
dynamic mem all

malloc()
calloc()

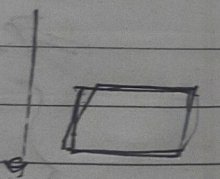
void
void*

if can be another

malloc( )
size

calloc(,)
no. size

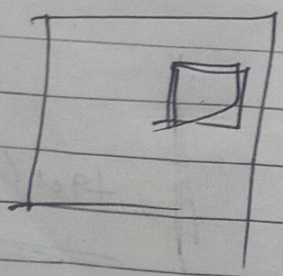
→ int* a = (int*) malloc(3x4);



3x sizeof(int)

int* a = (int*) calloc(3, 4)

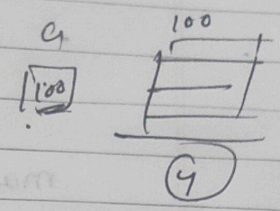
(3, sizeof(int))



Memory leak

Date
Page No.

(i) return return a
(ii) deallocate
↳ free(a)




```
int* create( ){
    int* a = (int*) calloc (4, sizeof(int));
    a[0] = 1
    a[1] = 2
    a[2] = 3
    a[3] = 4
    return a → // free(a)
}
void display (int** arr){

}
void main( )
int *b = create( );
display (b);

free (b)
}
```

Date
Page No.

Argument Array

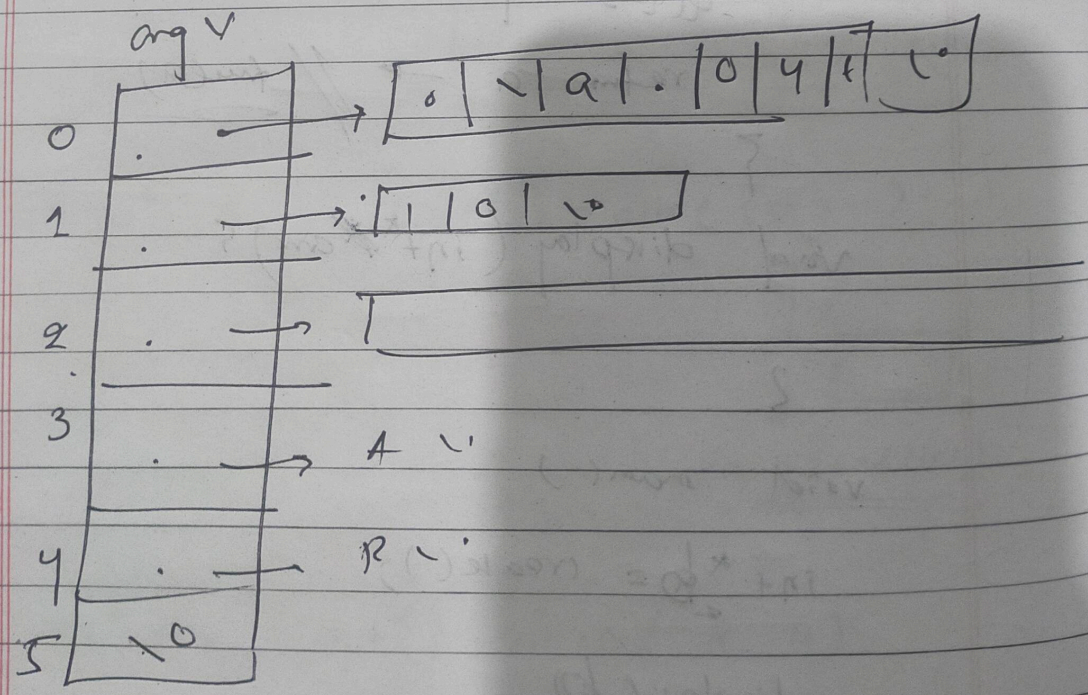
./a.out 10 "Ayush sarar" 4 B

1 pointer

int

char *

```
main( int argc, char* argv)

print (argc) → 5

for (int i=0; i<argc; i++){

    printf ("%s", argv+i);

}
```



atoi

#. Char * trimb (const char* input) {

int size_t len = strlen (ip);

char* st = ip ;

char* end = ip + len - 1;

// trim leading

while (*start == ' ') {

start ++ ;

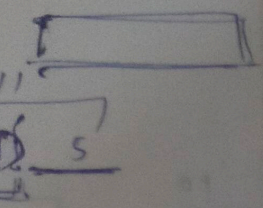
}

// trim trailing

while (end = ' ') {

end -- ;

}

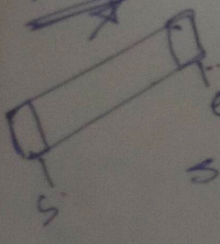


// trimmed :-

int trim_l = end - start + 1.

5-2+1 → 4

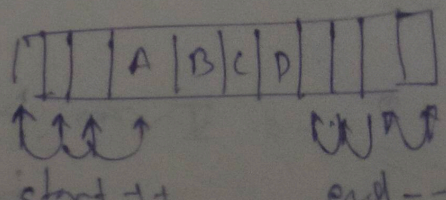
M. Allocation



char* trimmed = (char*) malloc (trim_l + 1);

strcpy (trimmed, start, trimed);

strcpy (x, y, 5)



M. locate

char* trimed = (char*) malloc (trimd+1).

null char

trim_l = size + 1.

strcpy (trimmed, start, trim_l).