PF

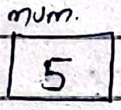
Types of pointers :-

1- Constant Pointer
2- Pointer to constant
3- Constant Pointer to constant.

• constant Pointer:

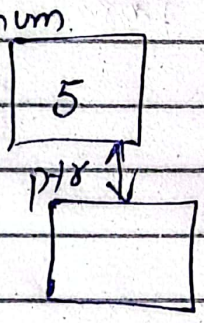
```
const int num = 5;
num = 6; → wrong
```

give error because constant means fix so value does not change.




```
int num = 5;
const
int *ptr = &num;
ptr = &num; → wrong → error.
```

→ pointer ki ek variable ke sath asterik aaraha hai.



is constant pointer ma address change nei kia jata.

• Pointer to Constant :-

```
int num = 5;
const int * ptr = &num;
*ptr = 6; → wrong
```

error.

pointer jis ko point kara hai yo change nai hota.

- **Constant pointer to constant.**

```
int num = 5;
const int * const ptr = &num;
    *ptr = 6;  → wrong
    ptr = num2;  → wrong.
```

give error.

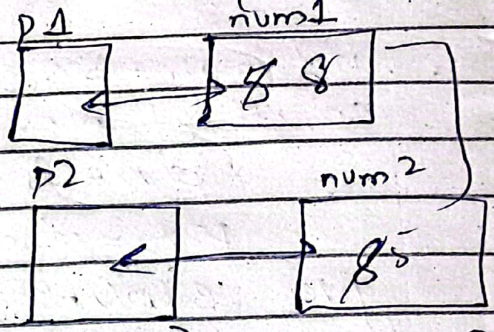
```
cout << ptr;  → address appoint
cout << *ptr;  → memory ma store
                 hota hai us ko point
                 karta hai
```

```
int main(){
 int num1 = 5;
 int num2 = 8;  //pointer pe (&) sign
 swap(&num1, &num2);
cout << num1 << " " << num2 << endl;

}
void swap(int &var1, int &var2){
 int temp = var1;
  var1 = var2;
  var2 = temp;
 cout << var1 << " " << var2 << endl;
}
```

Now we do same work with pointers.

```
void swap(int *p1, int *p2){
    5
int temp = *p1;
*p1 = *p2;
*p2 = temp;
```



(convert By reference to pointer) after swap

```
int main(){

int num = 5;
(void) increment(&num);
cout << num << endl;


void increment(int *ptr){

(num++) *ptr++;

}
```

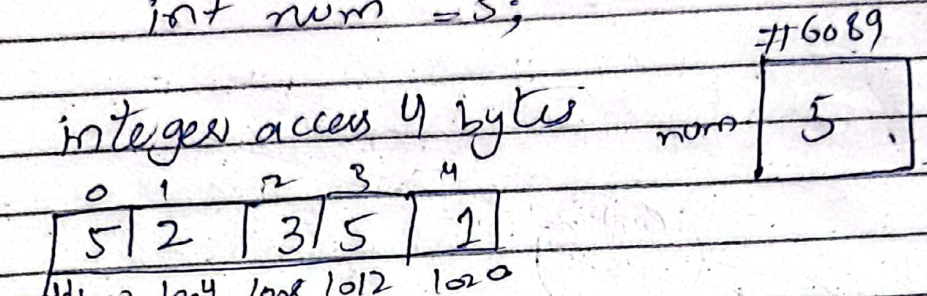
Level 3

1.2.1

PF

int num = 5;

integers access 4 bytes

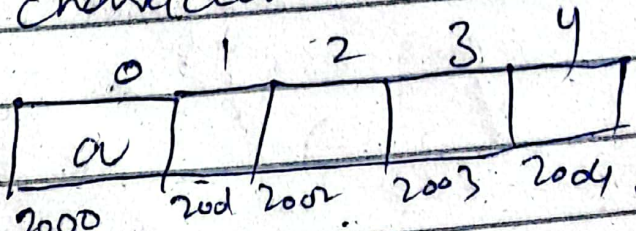


nums[index] → show index store value.

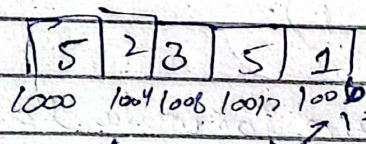
nums; → shows address of first value means 1000.

in address we used operation like:
(+) addition.
(-) subtraction.
(++) increment
(--) decrement.

in character:



4
integer
nums + 1.
1000 + 1 = 1004.
4 multiple 4 becuz of integer contain 4 byte first multiple the n add.
1000+3 = 1012.
2000-2 = 1992.
for double contain 8 byte.
for character contain 1 byte.



*(num+i) → dereferen. then 1000.
dereferen means jo address pai value hai yo access karna hai.

if 1000+3
1000 + 3 = 1012
give value of address 1012 is 5.

	0	1	2	3	4
num[i]	5	2	3	5	1
&num[i]	1000	1004	1008	1012	1016
num+i address	1000	1004	1008	1012	1016
*(num+i)	5	2	3	5	1

```
const int size = 5;
int nums[size] = {5,2,3,5,1};
// array ke name ma & nai lagta
int *ptr
ptr = nums;
for(int i=0;i<size;i++){
cout<<ptr[i]; → aisay bhi kar
sakta aisay C++ khud
calculate karta hai
agar
cout<<*(ptr+i);
↓
is tarah hum formula laga
sakay hain or dereference kar
sakty hain in first case jo
khud laga kar kar sakta hai.
int
sum
void display(int *ptr, int size){
for(int i=0;i<size;i++){ int sum;
cout<<ptr[i]<<endl;
*(ptr+i)
sum = sum + ptr[i]; } return sum;
sum = sum + (ptr[i]);
const int size = 5;
int nums[size] = {5,2,3,5,1};
display(nums, size);
sum (int nums, size);
```

Find max double type array.

```
(double max(int*)

double max(double *ptr, int size){
  for(int i=0; i< size; i++) double max;
  {
    if (ptr[i]<0)
    {
      ptr[i]=maximum;
    }
  else
    {
      cout<<" minimum "<<endl;}.
  return max;
int main(){
 const int size=5;
double num

    int maximum = max(ptr, size)
```

```
double max(int *ptr, int size)
{
  int max=0;
for(int i=0; i<size; i++)
  {
    if(ptr[i]>max)
    {
    (cout<<x)
max = ptr[i];
  }
  return max;
  }
  int main(){
  const int size=5;
  int num[5]={5,3,5,2,1};
  cout<<Max(nums, size);
  return 0;
  }
```

compiler compile time

free

runtime.

Saray variable memory jahan compiletime

Jabtak program chaltay hai yo fix rehti hai.

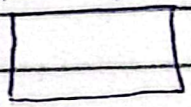
program machaltay chaltay hum free Hi karwa saktay hain storage benefit (g Dynamic memory allocate

Dynamic Memory (Run-time allocation)

→ memory ko koi naam nai de saktay assign nai ho sakti bus address int. hoga run-time.

→ main reason to read pointer is Dynamic memory.

→ jab free ho jati hai tu hum kaun delay hain: jab chahaye.



Keyword :-

new → dynamic memory allocation heap ma:

→ data type;

[new int;] → this line change by memory address.

we have to make pointer to store. in pointer ← (new int)

```
int * ptr = new int;
  *ptr=5;
```

pointer ke pass jo address hai us ma 5 store karwa.

if double then put double like / float thing every

```
double * ptr = new double;
```

delete ptr; → (not delete pointer means memory deallocate karwao jis ka address pointer ke pass hai)

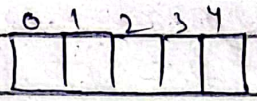
deallocate ← ↵

means free karwanay ke liye take naya use kia ja sakay).

(with double:-)

```
double * ptr = new (int) double;
  * ptr = 5;
```

```
delete ptr;
```




```
int * ptr = new int [size];
    for(int i=0; i<size; i++){
        cin >> ptr [i];

delete [] ptr;
```

(in array empty square bracket)
In Dynamic ma const int size
no need to put so we can take
size from user in dynamic not compulsory)

```
#include<iostream>
using namespace std;

int main(){
    int size;
    cin >> size;
int * ptr = new int [size];
for (int i=0; i<size; i++) {
    cin >> ptr[i];
}
cout << "display" << endl;
```

```
for(int i=0;i<size;i++){
    cout<<ptr[i]<<endl;
}
return 0;
}
```

output

4 → size

1

2

3

4

Display

1

2

3

4

$$= 8x^{1/8} + c.$$

size, array, array pass to function, min, max, average.

```
#include<iostream>
using namespace std;
double array(int *num[], int size)
{
    int min = ptr[0];
    int max = ptr[0];
for(int i=0; i<size; i++)
    {
    if((max >) num[i] > max)
    {
max= num[i];
    }
    else if (num[i] < min)
    {
    min=num[i];
    }
else{
    cout<<"invalid"<<endl;
    }
    int sum = 0;
sum= sum + num[i];
```

show first index store first value if {-1, -2, -3} show 0 but not show min so store 1st index.

```
double avg = 0;
avg = sum/2;

return avg(min, max, sum);


int main(){
 int size;
cout << "enter the size" << endl;
 cin >> size;
 int * ptr = new int[size]; for(int i=0; i<size; i++)
    { cin >> ptr[i]; }

array double(num, size);
 cout << "avg" << endl; delete[]ptr;
 return 0;
```

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
 ifstream fin;
 ofstream fout;
 fin.open("input.txt");
 fout.open("output.txt");
if(!fin.is_open()||!fout.is_open())
 {
  cout<<"error opening file"<<endl;
 }
  int size=0;
 int num;
 while(fin>>num)
  {
    size++;
  }
     fin.clear();
     fin.seekg(0);
 int *ptr = new int[size];
for(int i=0;i<size;i++)
 {
   fin>>ptr[i];
}
 for(int i=0;i<size;i++)
{
    fout<<ptr[i]; }.
```

```
delete[]ptr;
fin.close();
fout.close();
return 0;
```

• Distruction message

|PF|

* Return static Array } from function.
and
dynamic (Point) Array
* Dangling Pointer
* Memory leakage } pointer Hazards.
* Shallow Copy
* Deep Copy

Dangling pointer:-

→ Return Static Array:
in function ma array banatey hein tu array ki starting address likhtey hein phir int main mia pointer ma call karni hoti hai. destroy ho chuka hogai tu issue resolve nai hoga.

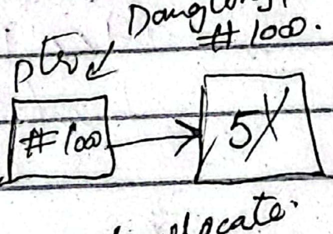
int *ptr = create();

in function

int * create() { return type).

→ use Dynamic array in functions.

Dangling pointer



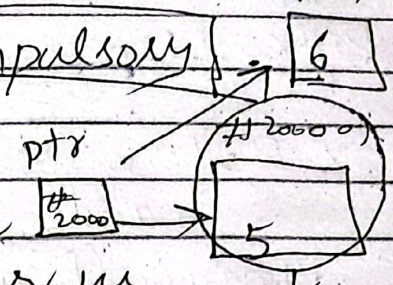
int *ptr = new int;
*ptr = 5;
delete ptr; → dangling pointer
cout << ptr << endl;

wo pointer jis k pass memory ka address hota hei jo deallocate hogati hei — ho chuki]

solution of dangling pointer.

ptr = nullptr; → compulsory

Memory leakage:-



also memory jis ko dyna mical create kia tha or us ko access nai kia ja sakta memory leakage hota hai.

memory leakage.

```
int num=6;
int *ptr = new int;
*ptr=5;
delete ptr; → solution
ptr=&num;
```

→ dynamic array ma memory leakage ka khatra hoga.

```
int *create(int size);
int (size) main(){
int size;
cout<<"enter size";
cin>>size;
int *ptr create(size);
cout<<"Display outside function"<<endl;
for(int i=0;i<size;i++){
    cout<<ptr[i]<<endl;
}
return 0;
}
```

```
int *create(int size)
{
 int *ptr=new int size[size];
 for(int i=0; i<size; i++)
 {
    p[i]= i+1;
 cout<<" display inside function"<<endl;
 for(int i=0; i<size; i++)
 {
  cout<<ptr[i]<<endl;
 }
 return ptr;
}
```

- Shallow copy:

```
int num=5;
int *ptr1=&num;
int *ptr2= ptr1;
  *ptr1=6;
```

data ko copy to kar lava lakin data same hovava.
issue here is same data copy ho rahva hai jis tavah point karta yo same hai is liye yo affect kar rahva hai ek kaam dusray ko affect kar rahva hai.

except

ptr

ptr = 0.

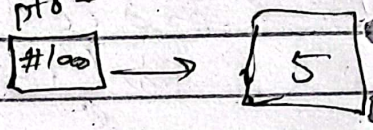
- Deep copy

int num = 5; → int *ptr1 = #

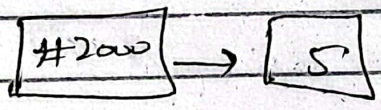
int *ptr2 = new int; → new memory create karwa li

*ptr2 = *ptr1;

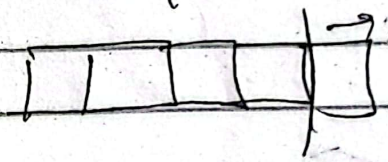
deep copy me new memory leo create karwa li to koi problem nahi hoga.



Dynamic array me size zayada ho jata hai out of → deep copy.



The bound jaraha ho to usse heap overflow kehtey hain



out of boundary = Memory leakage.

```
int *create();
int *reverse(int *ptr, int size) {
    int *newArr = new int[size];
    int j=0;
    for(int i=size-1; i>=0; i--)
    {
        newArr[j] = ptr[i];
        j++;
    }
    return newArr;
```

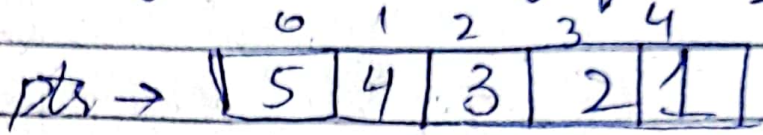
$x+y = \frac{2p}{q}$

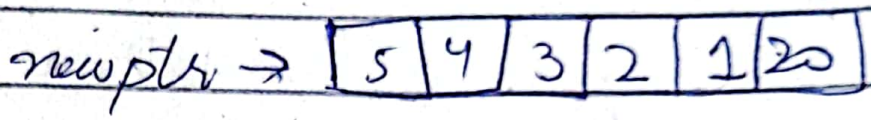
$x+y = \frac{p}{q} + \frac{m}{t}$

$\frac{p+m}{qt}$

Auto Grow / shrink

① Create new dynamic array of size+1. (size - 1)





② copy old array values into new array.

③ Deallocate old array.

④ size++; / size--;

```
int *autoGrow (int *ptr, int &size){
  // ek array or size receive karaye ga, in autograder
  int * newArr = new int[size+1]; → size-1;
  for(int i=0; i<size; i++){ → i<size-1;
    newArr[i] = ptr[i];
  }
  cin >> newArr[size]; → x in auto shrink.
  delete[] ptr; ✓
  ptr = nullptr; ✓
  size++; → size--;
  return newArr;
}
int main()
{
  int size = 5; ✓
  int * arr=new int [size]; ✓
for(int i=0; i<size; i++)
  {
    arr[i]= i; ✓
  }
  arr = autoGrow (arr, size);
}
return 0;
}
```

indouble sizex2
for three, size+3
for half size/2.

shrink.

PF

```
#include<iostream>
#include<fstream>
using namespace std;
int *autoGrow (int *oldArr, int &size);
int main(){
int size=1; →0
int *arr=new int[size]; →nullptr;
ifstream fin;
fin.open("data.txt",ios::in);
if(fin.is_open()){
int num;
while(fin>>num){
arr[size-1]=num;
arr=autoGrow(arr,size);
}
cout<<"display"<<endl;
for(int i=0;i<size-1;i++)
{
cout<<arr[i]<<endl; garbage value
}                    because autogrow
else{                increase one memory.
cout<<"error";
fin.close();
}
return 0;
}
```

→ deallocate

solution if put si put to null ptr & awk function call first

```
int main(){
 int size=0; // have zero size
 int *arr=nullptr;
    //zero size ki array banai hai

 if stream fin;
    // made the file
 fin.open("data.txt",ios::in);
  if(fin.is_open()){
    // read file
  int num;
    // initialize num.
 while(fin>>num){
    // read file
 arr=autoGrow(arr,size);
 function call so that size increment
```

(PF)

collection of array

Array of 1D-array.

2Dimensional Array (2D-Array)

data type identifier [rowssize] [colsize];

rowssize → rows

colsize → columns size

	columns 0	1	2	3
rows 0	0,0	0,1	0,2	0,3
1	1,0	1,1	1,2	1,3
2	2,0	2,1	2,2	2,3

↗ 2D array.

double numbers [3] [4];

At a time of initialization

double nums[3][4] = {{1,2,3,4},{5,6,7,8},{9,10,11,12}};

↳ without small brackets we do this better use bracket if not then value initialized with 0

if want to access any value then put rows and columns index:

nums[rows_index][columns index] = 5;

nums[2], [3], = 5;

→ if want to access all index value then put nested loop.

```
for(int r=0; r< row_size; r++)
{
  for( int c=0; c< col_size; c++) {
    cin>> nums [r] [c];
  }
}
```

i	j
0	0
	1
	2
1	0
	1
	2
2	0
	1
	2
3	0
	1
	2
4	

	0	1	2	3
	+	+	+	x
0	5	7	8	9
1	4	3	6	3
2	7	8	9	10

2D array

→ array of arrays.
→ In 2D array always two [] if put marks [0] → address first.
[1] → first row start address.

Address.
- marks → starting address
- marks [row-index] → index point

```
for(int i = 0; i<4; i++){
    display(marks[i], 3);
    cout << endl;
}
```

we can make function for display in 1D array to access all rows, columns. but we have to do seperately for 2D array access.

for script → console
base on value
submit in console

```
void display2D[int arr[][3], int rows, int col)
{
    put simple wala code jo pichey kia
}
also pass this function in main()

display(marks, 4, 3);
```

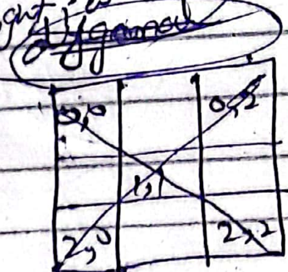
```
int
void 2Dsum(int [][3], int rows, int cols)
{
        int sum=0;
for(int i=0; i<3; i++){   → int sum=0;
   for(int j=0; j<4; j++){
      (sum[i][j] + arr)
   sum = sum + arr[i][j];
   }
}
   cout<< "sum:" << sum << endl;
return sum;
}
```

If do line by line row by row put int sum=0;

do function call.

2D sum(marks, 4, 3);

if(i==j) → (0,0)(1,1)(2,2).



if (i+j == col(size-1).

generic nei hai kaw ke suplit karna hai.

*((arr + row_index * col_size) + col_index)

(1000 + 0 * 3) + 0 = 1000

(1000 + 1 * 3) + 1 = 1004.

(1000 + 2 * 3) + 2 = 1008.

(1000 +

```
int main(){
int nums[3][3] = {        // in generic function.
cout<<diagonalSum(nums[0], 3, 3);
return 0;
```

Generic function

```
int diagonalSum(int *ptr, int rows, int cols){
for(int i=0; i<rows; i++){          → int sum=0;
  for(j=0; j<cols; j++){
    cout<<*((ptr + i*cols) + j))<<endl;
    sum = sum +
  }
  return sum;
}
if(i==j)
```

*(arr + i * cols) + j

1000 + 1 * 3 + 1

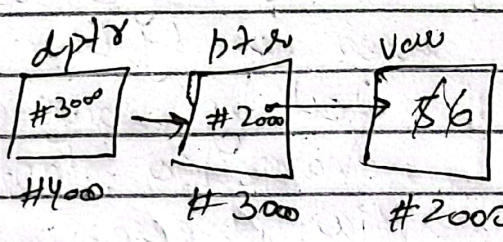
1 → 2, 1 → 3

2D ARRAY DYNAMIC MEMORY.

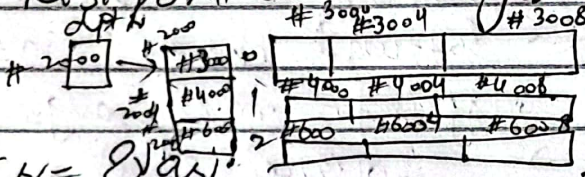
→ 2D ARRAY ek sath dynamic nav bana sakte.
→ har row ko elada elada 1D dynamic array banaye or us ko merge kar de.
→ har row ko point / har array pointer

Double pointer-

int var;



double pointer bhi address store karaye ga lekin wo kisi pointer ka hoga.




```
int var;
int *ptr = &var;
*ptr = 5;
int **dptr = &ptr;
**dptr = 6;
```

2D dynamic banane ke liye double pointer banate hain.
Pointer ki array ke har array ek pointer ho.

```
int **dptr = new int*[rows];
for(int i=0; i<rows; i++)
{
    dptr[i] = new int[cols];
}
```

create 2D dynamic

```
*(*(dptr + r) + c)
```

row_index, col_index

((dptr + r) + c)

*(2000 + 0)

(3000 + 0)

3000.

4byte int so 2 x 4 = 8 ; + c. index

*(2000 + 2) → 2008 → *(6000 + 1) →

6004.

```
int **dptr = new int* [rows];
for (int i=0; i < rows; i++) {
    dptr[i] = new int [cols];
}
for( int r = 0; r = rows; r++) {
    for (int c=0; c=cols; c++) {
        cout<<*(*(dptr + r) + c);
    }
}
```

manipulate karnay ke liye we karty hain.

for deallocate 2D dynamic

```
for (int i=0; i<rows; i++) {
    delete [] dptr[i];
}
delete[] dptr;
dptr = nullptr;
```

deallocate

↓

dangling pointer

first index access so memory leakage.

```
bool equal(int** ptr, int rows, int cols){
    bool check = true;
    for(r=0; r<rows; r++){
        for(c=0; c<cols; c++){
            if((i != j) && *(*(ptr+r)+c) != 1
                cout << *(*(ptr+r)+c) << endl;
            }
            else
            {
                Bool
                cout << " not equal " << endl;
            }
        else if(i != j) && *(*(ptr+r)+c).
            check = false;
            break; }.

    return ptr;
}
    int main(){
    int rows;
    cin >> rows;
    int cols;
    cin >> cols;
    int** equals = equal(ptr, rows, cols);
```

```
int ** ptr = new int * [rows];
for( int i=0; i< cols; i++) {
  ptr[i] = new int [cols];
}

int ** equals = equal( ptr, rows, cols);

cout << " equals " << endl;

for (int j=0; j< rows; j++) {
  delete [] ptr[i];
}
  delete [] ptr;
ptr = nullptr;

return 0;
}
```