Fundamentals of Object Oriented Programming

CSN- 103

Dr. R. Balasubramanian

Associate Professor

Department of Computer Science and Engineering

Indian Institute of Technology Roorkee

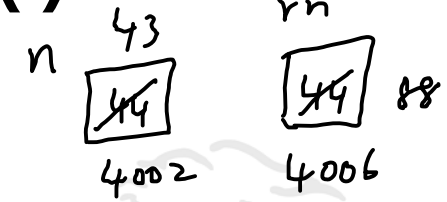
Roorkee 247 667

balarfcs@iitr.ac.in

https://sites.google.com/site/balaiitr/

Reference Operator in C++

```
void main()
{
int n=44;
int rn;
rn=n;
cout<<n<<rn<<endl;
n--;
cout<<n<<rn<<endl;
rn*=2;
cout<<n<<rn<<endl;
}
```

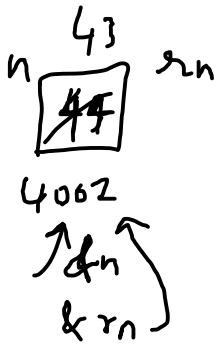


44 44

43 44

43 88

```
void main()
{
int n=44;
int& rn=n;
cout<<n<<rn<<endl;
n--;
cout<<n<<rn<<endl;
rn*=2;
cout<<n<<rn<<endl;
}
```



44 44

43 43

86 86

Dereference Operator (*) in C++

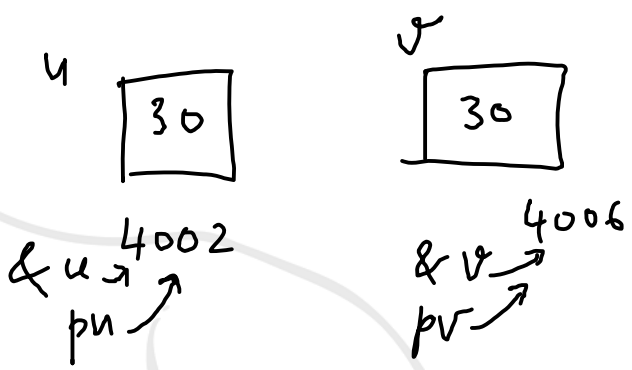
A pointer is a variable that points to an address of an another variable.

```
void main()
 {
int u=30;
int v;
int *pu, *pv;
pu=&u;
v=*pu;
pv=&v;
cout<<u<<&u<<pu<<*pu;
cout<<v<<&v<<pv<<*pv;
}
```

int* pu; // int *pu;

int& r=u;

int* pv;



v → *pu → *(&u) → u → 30

*pv → *(&v) → v → 30

30	4002	4002	30
30	4006	4006	30

```cpp
#include <iostream>

using namespace std;

int main()
{
    int u=30;
    int v;
    int *pu, *pv;
    pu=&u;
    v=*pu;
    pv=&v;
    cout<<" " <<u<<" "<<&u<<" "<<pu<<" "<<*pu<<endl;
    cout<<" " <<v<<" "<<&v<<" "<<pv<<" "<<*pv<<endl;

   return 0;
}
```

Terminal

```
sh-4.3$ g++ pointcheck0.cpp
sh-4.3$ a.out
 30 0x7fffa3a4ec8c 0x7fffa3a4ec8c 30
 30 0x7fffa3a4ec88 0x7fffa3a4ec88 30
sh-4.3$
```

```cpp
#include <iostream>

using namespace std;

int main()
{
    int v=3;
    int* pv;
    pv=&v;
    cout<<" "<<*pv<<" "<<v<<endl;
    *pv=5;
    cout<<" "<<*pv<<" "<<v<<endl;
    return 0;
}
```

```
Terminal
sh-4.3$ g++ pointcheck2.cpp
sh-4.3$ a.out
 3 3
 5 5
sh-4.3$
```

Null Pointer in C++

```
void main()
 {
int *pv;
int pv=0;//pv=NULL;
cout<<*pv;
}
```

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int* pv;
8      pv=NULL;
9      cout<<*pv;
10
11     return 0;
12 }
13
```

Terminal

```
sh-4.3$ g++ pointcheck1.cpp
sh-4.3$ a.out
Segmentation fault (core dumped)
sh-4.3$
```

int *pv = NULL;
cout<<*pv;

Dangling Pointer in C++

```
void main()
 {
int u1,u2;
int v=3;
int *pv;
u1=2*(v+5);
u2=2*(*pv+5);
cout<<u1<<u2;
}
```

Arrays and Pointers in C++


```
int a[10];
for (int i=0; i<10; i++)
  {cin>>a[i];
     cout<<a[i];
     cout<<a+i;
     cout<<*(a+i);
     cout<<i[a];
}
```

Pass by Value and Pass by Reference

- We have seen Java is pass by Value
- Now we will take an example in C++ and see how it is pass by reference!

Pass by Value

```
void swapVariables(float var1, float var2)
{  float temp;
temp = var1;
var1 = var2;
var2 = temp;
cout<<var1<<var2;
}
int main( )
{  float float1 = 3.5,  float2 = 5.6;
swapVariables(float1, float2);
cout<<float1<<float2;
return 0;
}
```

```
1  #include <iostream>
2
3  using namespace std;
4
5  void swapVariables(float var1, float var2)
6  {  float temp;
7  temp = var1;
8  var1 = var2;
9  var2 = temp;
10 cout<<"Inside Function"<<endl;
11 cout<<var1<<" "<<var2<<endl;
12 }
13
14 int main( )
15 {  float float1 = 3.5,  float2 = 5.6;
16 swapVariables(float1, float2);
17 cout<<float1<<" "<<float2<<endl;
18 return 0;
19 }
```

Terminal

```
sh-4.3$ g++ swapval.cpp
sh-4.3$ a.out
Inside Function
5.6 3.5
3.5 5.6
sh-4.3$
```

Pass by Reference

```
void swapVariables(float& var1, float& var2)
{  float temp;
temp = var1;
var1 = var2;
var2 = temp;
}
int main( )
{  float float1 = 3.5,  float2 = 5.6;
swapVariables(float1, float2);
cout<<float1<<float2;
return 0;
}
```

```
1  #include <iostream>
2
3  using namespace std;
4
5  void swapVariables(float& var1, float& var2)
6  {  float temp;
7  temp = var1;
8  var1 = var2;
9  var2 = temp;
10 cout<<"Inside Function"<<endl;
11 cout<<var1<<" "<<var2<<endl;
12 }
13
14 int main( )
15 {  float float1 = 3.5,  float2 = 5.6;
16 swapVariables(float1, float2);
17 cout<<"In Main"<<endl;
18 cout<<float1<<" "<<float2<<endl;
19 return 0;
20 }
```

Terminal

```
sh-4.3$ g++ swapvar.cpp
sh-4.3$ a.out
Inside Function
5.6 3.5
In Main
5.6 3.5
sh-4.3$
```

Using pointers

```
void swapVariables(float* var1, float* var2)
{  float temp;
temp = *var1;
*var1 = *var2;
*var2 =  temp;
}
int main( )
{  float float1 = 3.5,  float2 = 5.6;
swapVariables(&float1, &float2);
cout<<float1<<float2;
return 0;
}
```

```cpp
#include <iostream>

using namespace std;

void swapVariables(float* var1, float* var2)
{  float temp;
temp = *var1;
*var1 = *var2;
*var2 = temp;
cout<<"Inside Function"<<endl;
cout<<*var1<<" "<<*var2<<endl;
}

int main( )
{  float float1 = 31.5,  float2 = 51.6;
swapVariables(&float1, &float2);
cout<<"In Main"<<endl;
cout<<float1<<" "<<float2<<endl;
return 0;
}
```

Terminal

```
sh-4.3$ g++ swappoint.cpp
sh-4.3$ a.out
Inside Function
51.6 31.5
In Main
51.6 31.5
sh-4.3$
```

```
void g(int x, int& y);
void main()
{ int a,b;
a=20;
b=15;
g(a,b);
cout<<a<<b<<endl;
g(5*a-2,b);
cout<<a<<b<<endl;
g(a,5*b-3);
cout<<a<<b<<endl;
}
```

```
void g(int x, int& y)
  { x=50;
    y=80;
  }
```

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int v=3;
8      int* pv;
9      pv=&v;
10     cout<<" "<<*pv<<" "<<v<<endl;
11     *pv=5;
12     cout<<" "<<*pv<<" "<<v<<endl;
13    return 0;
14 }
```

```
Terminal
sh-4.3$ g++ pointcheck2.cpp
sh-4.3$ a.out
 3 3
 5 5
sh-4.3$
```

Pointer and reference variables

```
pv=0x7fffc9af9b88;
```

```
pv=0;
```

```
pv=NULL;
```

```
int v1=2*(*pv+v);
```

```
&w=2*(&u+&v);
```

```
int v2=2*(&u+&v);
```

pu+pv

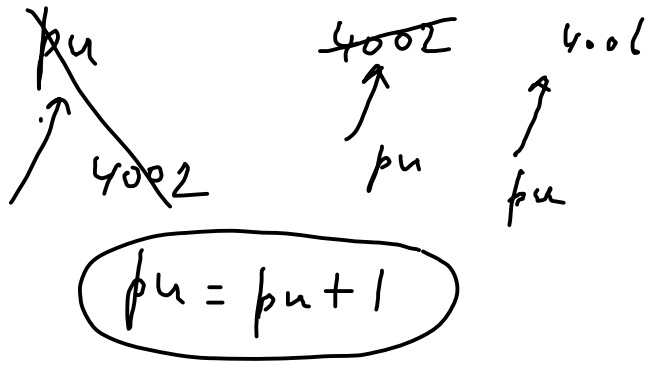
*pu+*pv

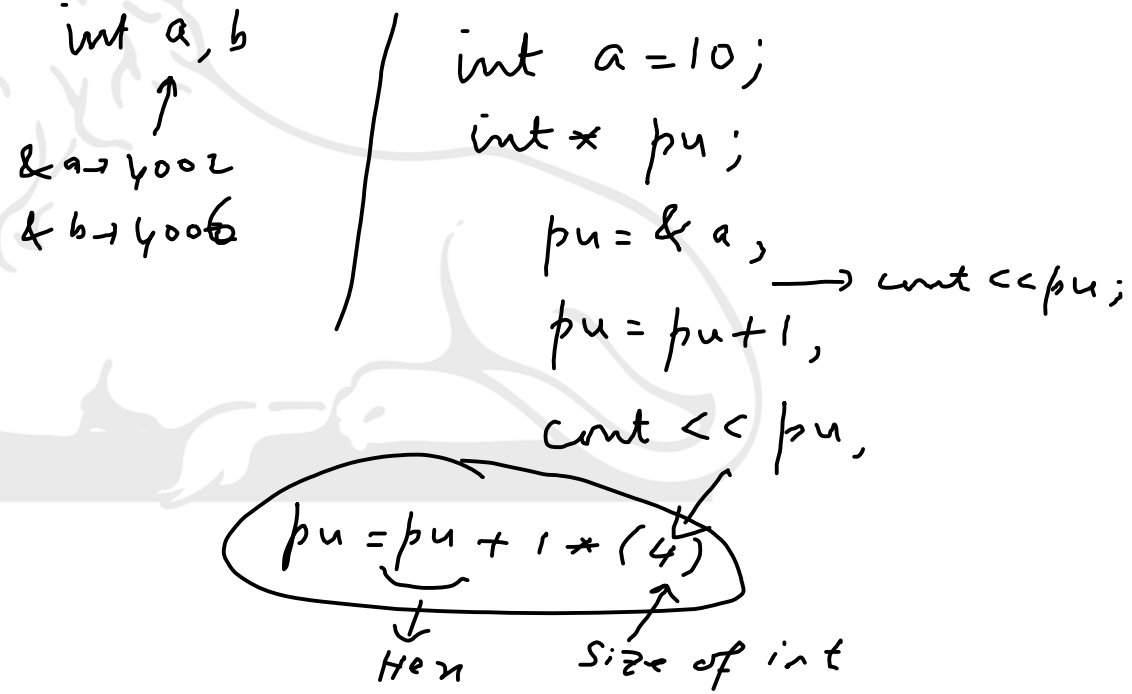
pu=pu+1;

pv=*pu+pv;

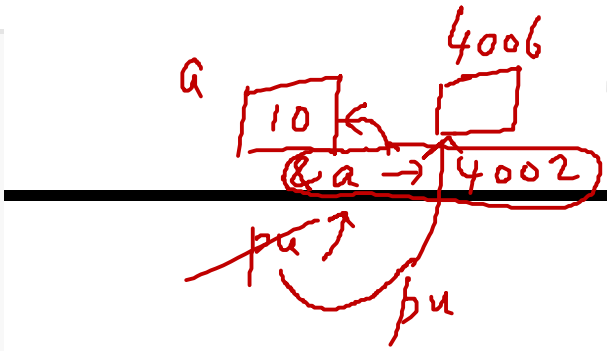
pu=*pu+pv;

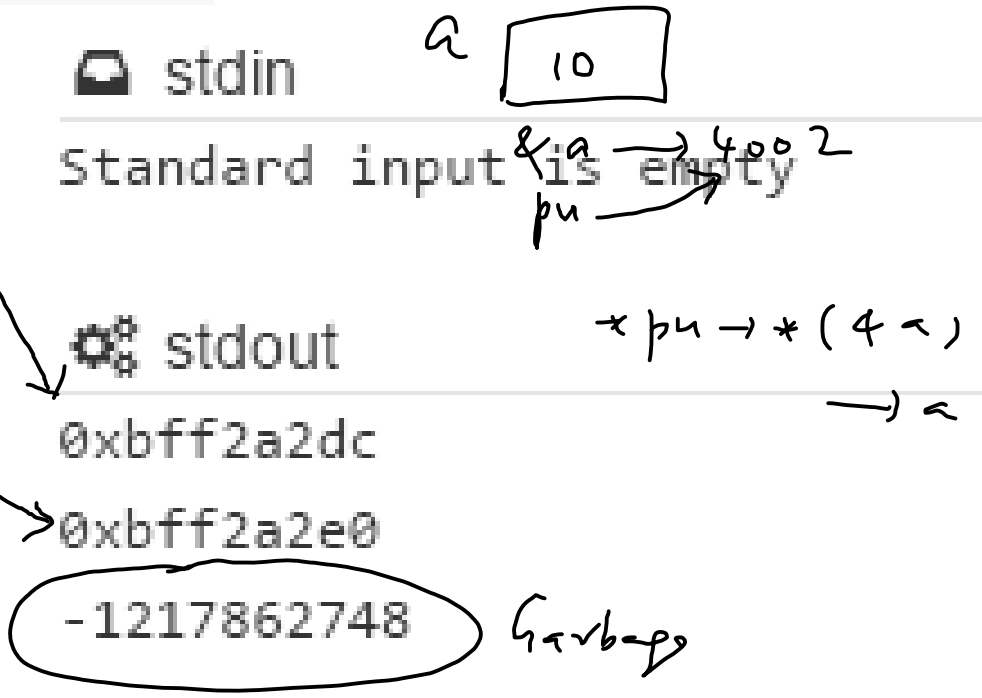
*pu=*pu+pu





```
1.  #include <iostream>
2.  using namespace std;
3.
4.  int main() {
5.      // your code goes here
6.      int a=10;
7.      int* pu;
8.      pu=&a;
9.      cout<<pu<<endl;
10.     pu=pu+1;
11.     cout<<pu<<endl;
12.     cout<<*pu;
13.     return 0;
14. }
```





*pu = &a; ✗

int* pu;
pu = &a;

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      // your code goes here
6      int a=10;
7      int* pu=&a;
8      cout<<pu<<endl;
9      cout<<*pu<<endl;
10     return 0;
11 }
```

Terminal

```
sh-4.3$ g++ point1.cpp
sh-4.3$ a.out
0x7fff2d8d7114
10
sh-4.3$
```

Null Pointer in C++

```
void main()
 {
int *pv;
pv=0;//pv=NULL;
cout<<*pv;
}
```

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int* pv;
8      pv=NULL;
9      cout<<*pv;
10
11     return 0;
12 }
13
```

Terminal

```
sh-4.3$ g++ pointcheck1.cpp
sh-4.3$ a.out
Segmentation fault (core dumped)
sh-4.3$
```

a.out

- "a.out"
  - the default executable target name when one is not specified.

- For more commands
  - http://www.cs.fsu.edu/~jestes/howto/g++compiling.txt

Dangling Pointer in C++

```
void main()
 {
int u1,u2;
int v=3;
int *pv;
u1=2*(v+5);
u2=2*(*pv+5);
cout<<u1<<u2;
}
```

Arrays and Pointers in C++

```
int a[10];
for (int i=0; i<10; i++)
  {cin>>a[i];
     cout<<a[i];
     cout<<a+i;
     cout<<*(a+i);
     cout<<i[a];
}
```

a[0] a[1]

10 20 30 40 50 ... 100

&a → 4002

a+0 → 4006

i = 1

10 20 20

a+0 → base address → 4002 4006

10 20

10 20

a+0
↳ &a[0]
a+1
↳ &a[1]

*(a+0)
↳ *(&a[0])
↓
a[0]

pu = &b
*pu
*(&b)
↳ b

*(a+i)

a[i] → *(a+i)

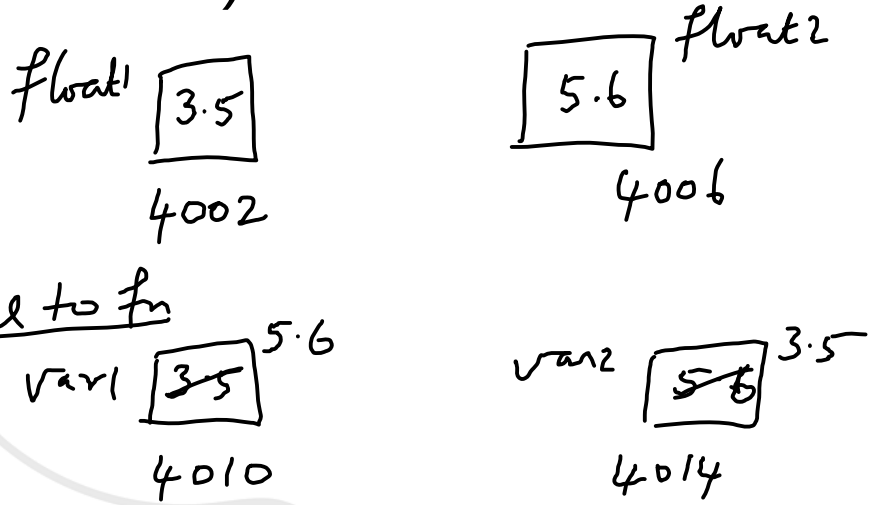
*(a+i) → *(i+a) → ~~i[a]~~ i[a]

Pass by Value and Pass by Reference

- We have seen Java is pass by Value
- Now we will take an example in C++ and see how it is pass by reference!

Pass by Value

```
void swapVariables(float var1, float var2)
{  float temp;
temp = var1;
var1 = var2;
var2 = temp;
cout<<var1<<var2;
}
int main( )
{  float float1 = 3.5,  float2 = 5.6;
swapVariables(float1, float2);
cout<<float1<<float2;
return 0;
}
```



```
1  #include <iostream>
2
3  using namespace std;
4
5  void swapVariables(float var1, float var2)
6  {  float temp;
7  temp = var1;
8  var1 = var2;
9  var2 = temp;
10 cout<<"Inside Function"<<endl;
11 cout<<var1<<" "<<var2<<endl;
12 }
13
14 int main( )
15 {  float float1 = 3.5,  float2 = 5.6;
16 swapVariables(float1, float2);
17 cout<<float1<<" "<<float2<<endl;
18 return 0;
19 }
```

Terminal

```
sh-4.3$ g++ swapval.cpp
sh-4.3$ a.out
Inside Function
5.6 3.5
3.5 5.6
sh-4.3$
```

Pass by Reference

```
void swapVariables(float& var1, float& var2)
{  float temp;
temp = var1;
var1 = var2;
var2 = temp;
}
int main( )
{  float float1 = 3.5,  float2 = 5.6;
swapVariables(float1, float2);
cout<<float1<<float2;
return 0;
}
```

```cpp
#include <iostream>

using namespace std;

void swapVariables(float& var1, float& var2)
{  float temp;
temp = var1;
var1 = var2;
var2 = temp;
cout<<"Inside Function"<<endl;
cout<<var1<<" "<<var2<<endl;
}

int main( )
{  float float1 = 3.5,  float2 = 5.6;
swapVariables(float1, float2);
cout<<"In Main"<<endl;
cout<<float1<<" "<<float2<<endl;
return 0;
}
```

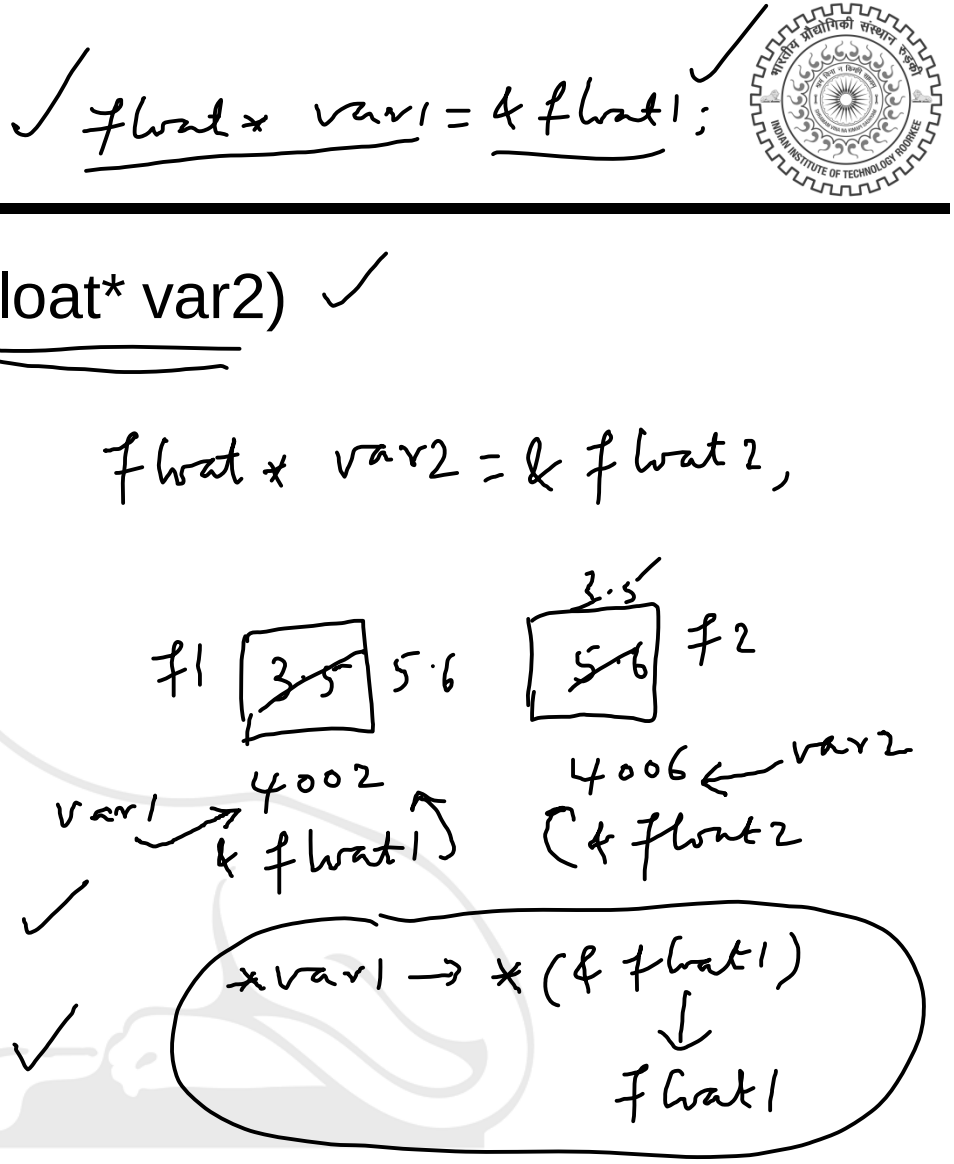
float& var1 = float1;

Terminal

```
sh-4.3$ g++ swapvar.cpp
sh-4.3$ a.out
Inside Function
5.6 3.5
In Main
5.6 3.5
sh-4.3$
```

Using pointers

```
void swapVariables(float* var1, float* var2)
{  float temp;
temp = *var1;
*var1 = *var2;
*var2 =  temp;
}
int main( )
{  float float1 = 3.5,  float2 = 5.6;
swapVariables(&float1, &float2);
cout<<float1<<float2;
return 0;
}
```



```cpp
#include <iostream>

using namespace std;

void swapVariables(float* var1, float* var2)
{  float temp;
temp = *var1;
*var1 = *var2;
*var2 = temp;
cout<<"Inside Function"<<endl;
cout<<*var1<<" "<<*var2<<endl;
}

int main( )
{  float float1 = 31.5,  float2 = 51.6;
swapVariables(&float1, &float2);
cout<<"In Main"<<endl;
cout<<float1<<" "<<float2<<endl;
return 0;
}
```

Terminal

```
sh-4.3$ g++ swappoint.cpp
sh-4.3$ a.out
Inside Function
51.6 31.5
In Main
51.6 31.5
sh-4.3$
```

```
void g(int x, int& y);
void main()
{ int a,b;
a=20;
b=15;
g(a,b);
cout<<a<<b<<endl;
g(5*a-2,b);
cout<<a<<b<<endl;
g(a,5*b-3);
cout<<a<<b<<endl;
}
```

```
void g(int x, int& y)
{ x=50;
  y=80;
}
```

x = 20

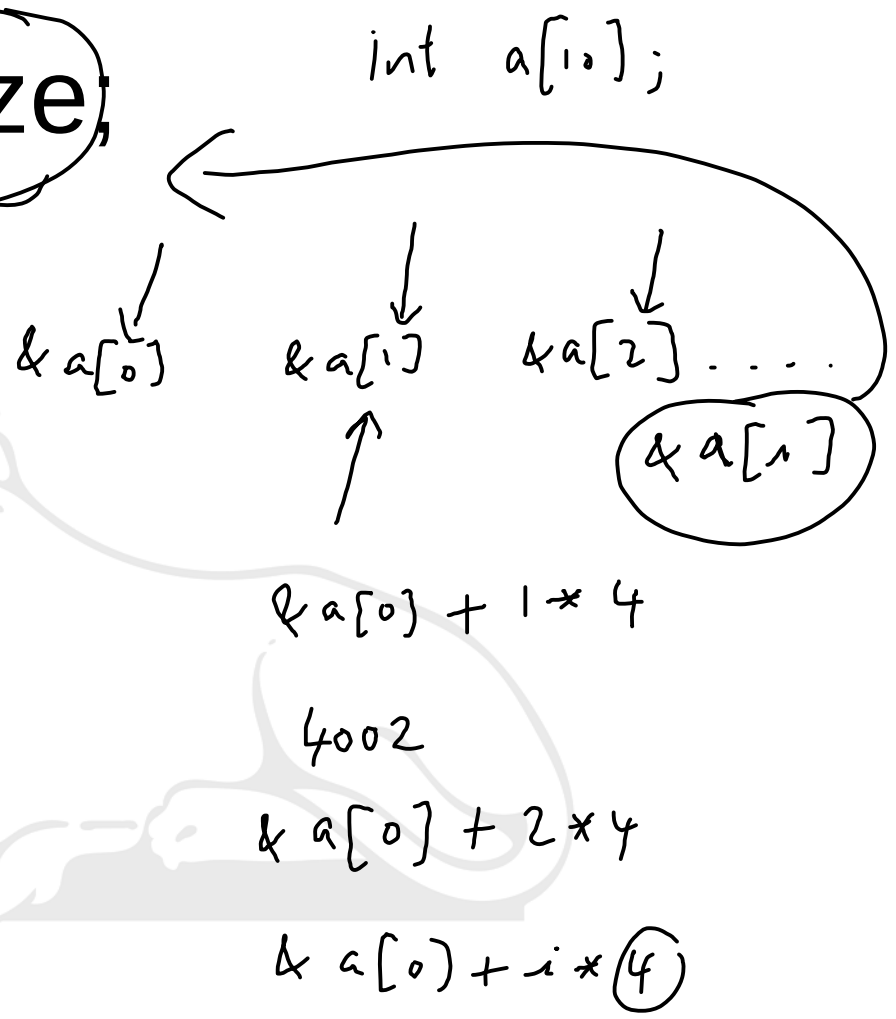
int & y = b

a: 20 — &a → 4002

b: ~~15~~ 80 — &b → 4006 (&y)

Address of i^{th} location element in an Array

&a[i]=&a[0]+i*e_size;



Static Array

</> source code

```
1  // Example program
2  #include <iostream>
3  using namespace std;
4
5  int main() {
6      // your code goes here
7      int a[5];
8      for (int i=0; i<5; i++)
9       cin>>a[i];
10     for (int i=0; i<5; i++)
11      cout<<" "<<a[i];
12     return 0;
13 }
```

Success time: 0 memory: 3460 signal:0

```
20 30 40 50 60
```

Static Array

- Memory will be allocated during compile time

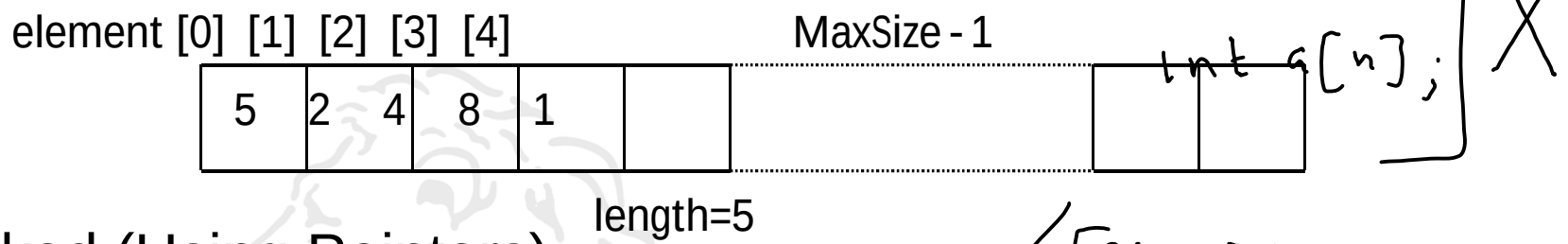
</> source code

```
1   // Example program
2   #include <iostream>
3   using namespace std;
4
5   int main() {
6       // your code goes here
7       int a[5];
8       for (int i=0; i<5; i++)
9        cin>>a[i];
10      for (int i=0; i<5; i++)
11       cout<<a[i];
12      delete [ ] a;
13       /*for (int i=0; i<5; i++)
14       cout<<a[i];*/
15      return 0;
16  }
```

Runtime error time: 0 memory: 3416 signal:11

One Dimensional Arrays

- Contiguous: `int a[10];`




```
cin >> n;
int a[n];
```

X

- Linked (Using Pointers)

```
int* a;
a=new int[10];
```

n

✓

```
cin >> n;
int *a;
a = new int[n];
```

✓

```
int *a = new int[10];
```

✓ C++

```
int [] a = new int[10];
```

JAVA

- Array is a collection of objects in which all are of same data type.

(1)

```
float a[8]={20.0,30.0,40.0};
for (int i=0; i<8; i++)
 cout <<a[i]<<endl;
```

Output:

20.0, 30.0, 40.0, 0.0, 0.0, 0.0,0.0,0.0

(2)

```
float a[4]={10.0,20.0,30.0,40.0};
for (int i=0; i<8; i++)
 cout <<a[i]<<endl;
```

Output:

10.0, 20.0, 30.0, 40.0, G1, G2, G3,G4

(3)

```
float a[4]={10.0,20.0,30.0};
for (int i=0; i<8; i++)
 cout <<a[i]<<endl;
```

Output:

10.0, 20.0, 30.0, 0.0, G1, G2, G3, G4

(4)

```
float a[]={10.0,20.0,30.0};
for (int i=0; i<8; i++)
 cout <<a[i]<<endl;
```

Output:

10.0, 20.0, 30.0, G1, G2, G3, G4, G5

(5)

```
int a[6]={10,20,30,40,50,60};
cout <<a<<endl;
cout <<a[0]<<endl;
cout <<a+4<<endl;
cout <<a[4]<<endl;
```

Output:

Base address

10

Address of a[4] (or) &a[4]

50

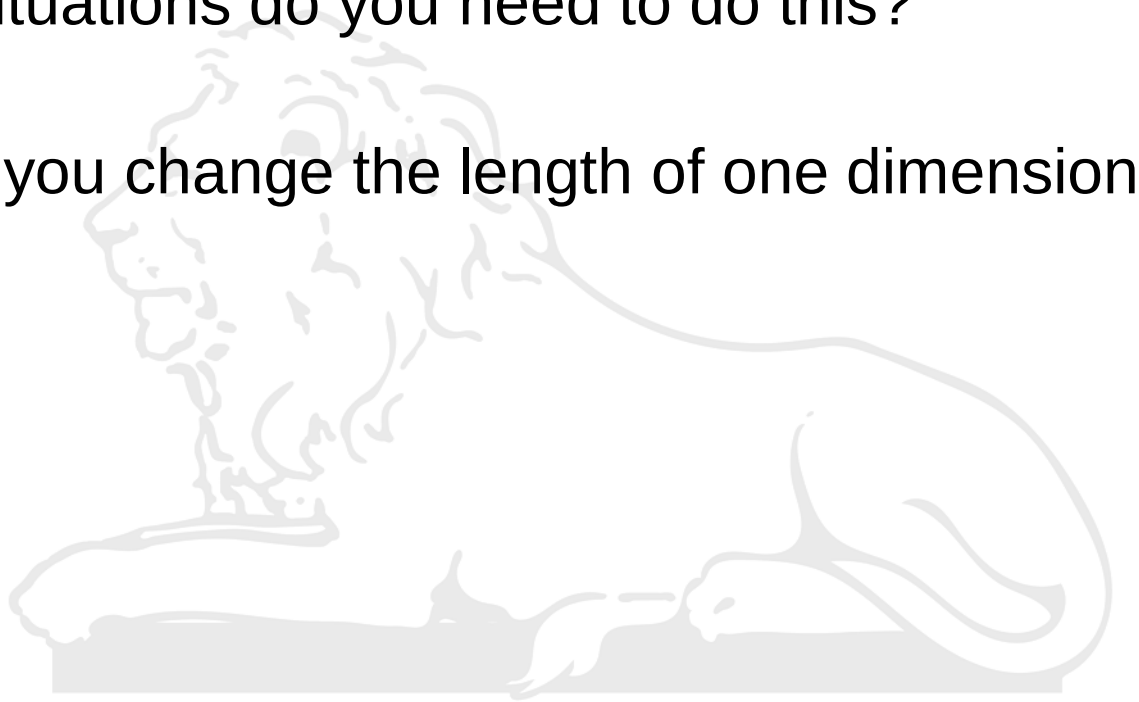
- Write a C++ program to generate 1000 random numbers between 0 to 999 and find max, min and average of these numbers.

rand()%1000

Rand()

Changing the Length of 1D Array

- What does it mean to change the length of an array?
- In what situations do you need to do this?
- How can you change the length of one dimensional array?



C++

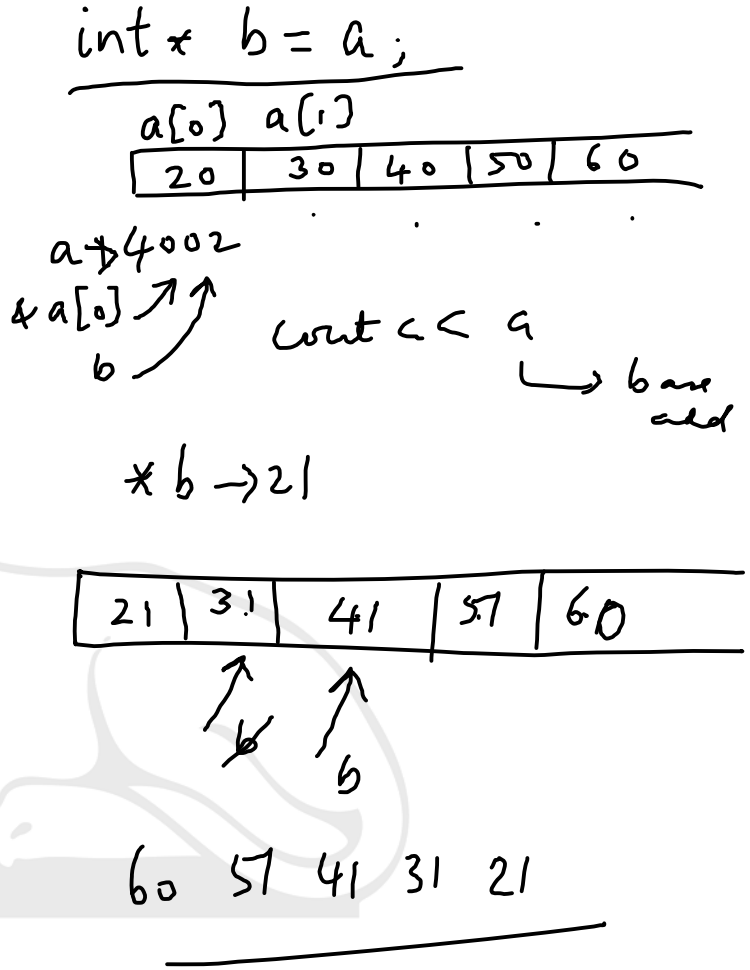
```
int *a;
int n=10;
while (n>0)
 { a=new int[n];
for (int i=0; i<n;i++)
 cin>>a[i];
sort(a,n); //some
           //function
n--;
delelte [ ] a;
}
```

JAVA

```
int [ ] a;
int n=10;
while (n>0)
 { a=new int[n];
for (int i=0; i<n;i++)
 a[i] = input.nextInt();
sort(a,n); //some function
n--;
}
```

Find the output of the following C++ Program

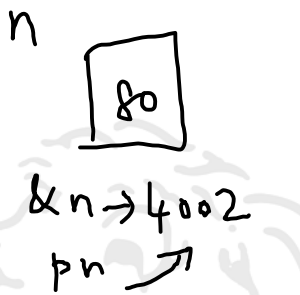
```
void change(int* b);
void main()
 { int a[5]={20,30,40,50,60};
    change(a);
     for (int i=4; i>=0; i--)
    cout<<a[i];
  }
void change(int* b)
{ for (int i=0; i<4; i++)
    { *b=*b+1;
       b++;
     }
}
```



```
void change(int* b)
void main()
 { int a[5]={20,30,40,50,60};
      for (int i=0; i<4; i++)
    { *a=*a+1;
    a++;
      }
for (int i=4; i>=0; i--)
   cout<<a[i];
 }
```

Pointer to Pointer

```
void main()
{ int n=80;
  cout<<n;
  cout<<&n;
  int *pn;
  pn=&n;
  cout<<pn;
  cout<<&pn;
  cout<<*pn;
```



80

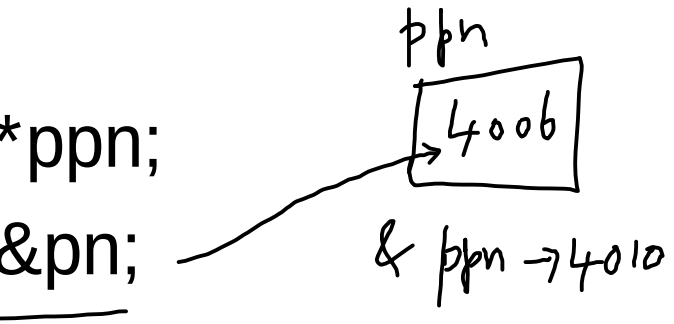
4002

4002

4006

*(&n)→n→80

```
int **ppn;
ppn=&pn;
cout<<ppn;
cout<<&ppn;
cout<<*ppn;
cout<<**ppn;
}
```



4006

4010

*(&pn)→pn
4002

(&pn)
*(pn)
80

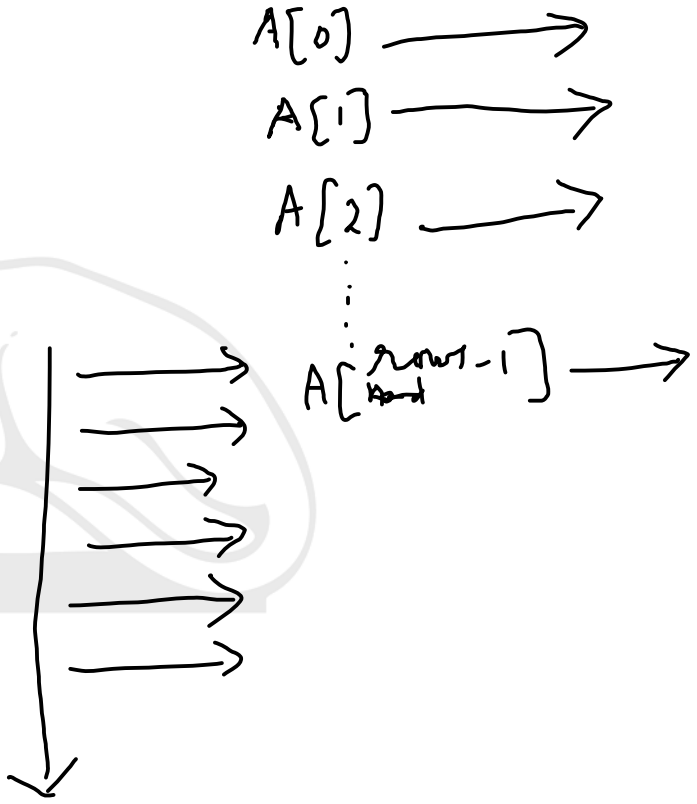
Two Dimensional Arrays

- In C++

Static Array: `int  arr_2d[10][12];`

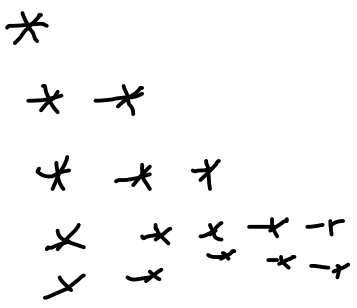
Dynamic Array:

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



Java arrays can be multidimensional. For example, a 2-dimensional array is an array of arrays. Two-dimensional arrays need not be rectangular. Each row can be a different length. Here's an example:

```
int [ ][ ] A;          // A is a two-dimensional array
A = new int[5][ ];     // A now has 5 rows, but no columns yet
A[0] = new int [1];  // A's first row has 1 column
A[1] = new int [2];  // A's second row has 2 columns
A[2] = new int [3];  // A's third row has 3 columns
A[3] = new int [5];  // A's fourth row has 5 columns
A[4] = new int [5];  // A's fifth row also has 5 columns
```



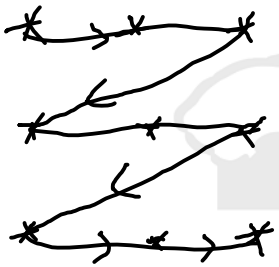
Two Dimensional Arrays

- In Java

```
int [ ] [ ] A;
A= new int[rows][ ];
 for (int i=0; i<rows; i++)
    A[i]=new int [cols];
```

In C++ Example 1

```
int a[3][3]={1,2,3,4,5,6};
  for (int i=0; i<3; i++)
     { for (int j=0; j<3; j++)
          cout<<a[ i ][ j ]<<" ";
          cout<<endl;
     }
```



Output:

1 2 3

4 5 6

0 0 0

In C++,

Example 2

```
int a[3][3]={1,2,3,4,5,6};
  for (int i=0; i<3; i++)
    { for (int j=0; j<7; j++)
        cout<<a[ i ][ j ]<<" ";
        cout<<endl;
    }
```

Output:

1	2	3	G1	G2	G3	G4
4	5	6	G5	G6	G7	G8
0	0	0	G9	G10	G11	G12

In C++,

Example 3

```
int a[ ][3]={1,2,3,4,5,6};
  for (int i=0; i<3; i++)
     { for (int j=0; j<3; j++)
          cout<<a[ i ][ j ]<<" ";
          cout<<endl;
     }
```

Output:

```
1    2    3
4    5    6
G1   G2   G3
```

In C++, Example 4

```
int a[ ][3]={1,2,3,4,5,6};
  for (int i=0; i<3; i++)
    { for (int j=0; j<7; j++)
        cout<<a[ i ][ j ]<<" ";
        cout<<endl;
    }
```

Output:

```
1   2   3   G1  4   G2  G3  G4
4   5   6   G5  G6  G7  G8
G9  G10 G11 G12 G13 G14 G15
```

In C++,

Example 5

```
int a[ ][3]={1,2,3,4,5,6,7};
  for (int i=0; i<3; i++)
    { for (int j=0; j<3; j++)
        cout<<a[ i ][ j ]<<" ";
        cout<<endl;
    }
```

Output:

```
1   2   3
4   5   6
7   0   0
```

In C++,

Example 6

```
int a[ ][3]={1,2,3,4,5,6,7};
    for (int i=0; i<3; i++)
      { for (int j=0; j<3; j++)
            cout<<a[ i ][ j ]<<" ";
            cout<<endl;
      }
```

```
int a[3][ ] = {1, 2, 3, 4, 5, 6};
```

Error

Output:

1	2	3
4	5	6
7	0	0

Address of [*i*1][*i*2]th location element in a 2D-Array

Given a[*r*1][*r*2] array,

*r*1-No. of rows, *r*2-No. of Cols.

0≤*i*1<*r*1 and 0≤*i*2<*r*2, finding the address of a[*i*1][*i*2]

$0 \le i1 < r1$

$0 \le i2 < r2$

&a[*i*1][*i*2]=&a[0][0]+(*i*1*r2+*i*2)*e_size;

3-D

(i1 * r2 * r3 + i2 * r3 + i3) *

&a[i1][i2] = &a[0][0] +