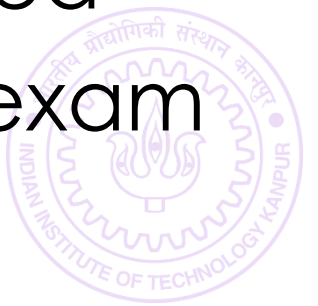
Mr C stands in a line

ESC101: Fundamentals of Computing

Purushottam Kar

Lab Exam (Sun, 09 Sep)

- Come only to your assigned session at least 15 minutes before start of exam – assigned seating will be there!
- **BRING YOUR INSTITUTE ID CARD, HANDWRITTEN NOTES**
- **DO NOT BRING YOUR MOBILE PHONES/IPADS TO EXAM**
- Syllabus – till loops (no arrays)
- **NO** printouts, photocopies, slides, websites allowed
- Prutor CodeBook will be unavailable during lab exam
- **TA HELP WILL NOT BE AVAILABLE DURING EXAM**



Lab Exam (Sun, 09 Sep)

- Morning exam (Wed, Thu batches)
  - 10:30 AM - 1:30 PM – starts 10:30 AM sharp
  - **CC-01**: B9, {B14 even roll numbers}
  - **CC-02**: B7, B10, B11
  - **CC-03**: B12
  - **MATH-LINUX**: B8, {B14 odd roll numbers}
- Go see your room during this week's lab
- Be there 15 minutes before your exam 10:15AM
- Cannot switch to afternoon session

Lab Exam (Sun, 09 Sep)

- Afternoon exam (Mon, Tue batches)
  - 2 PM - 5 PM – starts 2 PM sharp
  - **CC-01**: B1, {B2 even roll numbers}
  - **CC-02**: B4, B5, B6
  - **CC-03**: B3
  - **MATH-LINUX**: B13, {B2 odd roll numbers}
- Go see your room during this week's lab
- Be there 15 minutes before your exam 1:45 PM
- Cannot switch to morning session

Some Tips on Correcting Errors

Some Tips on Correcting Errors

Print non-output variables as well – it is boring ☺ but useful

Some Tips on Correcting Errors

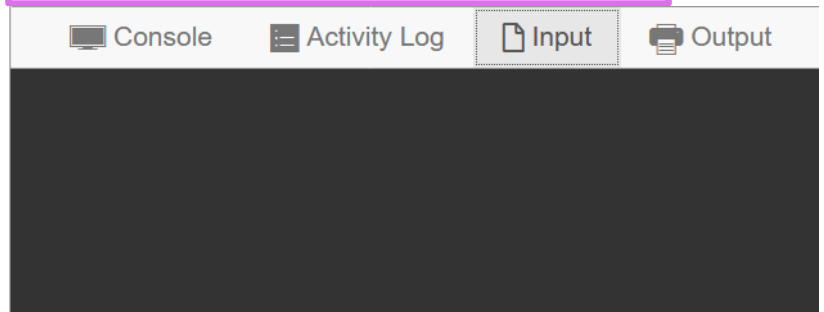
Print non-output variables as well – it is boring ☺ but useful

```
int sum = 0, i, num;
for(i = 1; i <= 5; i++){
    scanf("%d", &num);
    if (num > 0)
        continue;
    sum += num;
}
```

Some Tips on Correcting Errors

Print non-output variables as well – it is boring ☺ but useful

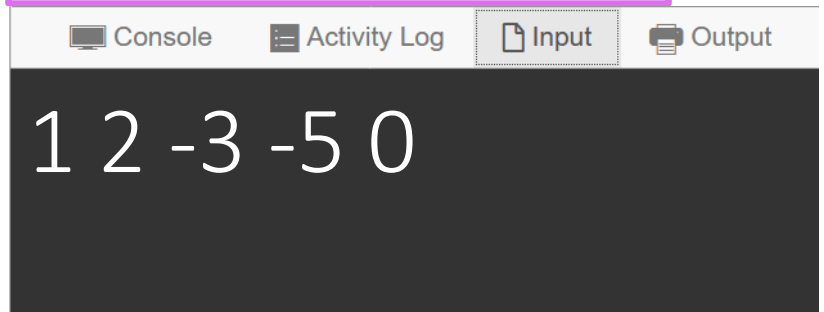
```
int sum = 0, i, num;
for(i = 1; i <= 5; i++){
    scanf("%d", &num);
    if (num > 0)
        continue;
    sum += num;
}
```



Some Tips on Correcting Errors

Print non-output variables as well – it is boring ☺ but useful

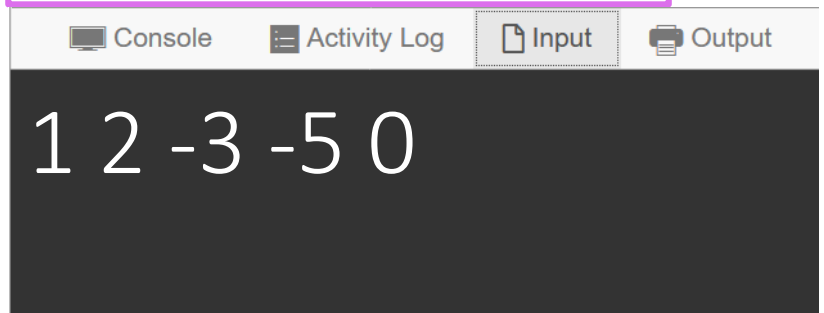
```
int sum = 0, i, num;
for(i = 1; i <= 5; i++){
    scanf("%d", &num);
    if (num > 0)
        continue;
    sum += num;
}
```

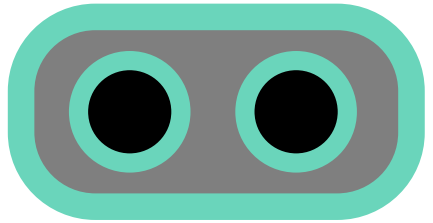


Some Tips on Correcting Errors

Print non-output variables as well – it is boring ☺ but useful

```
int sum = 0, i, num;
for(i = 1; i <= 5; i++){
    scanf("%d", &num);
    if (num > 0)
        continue;
    sum += num;
}
```

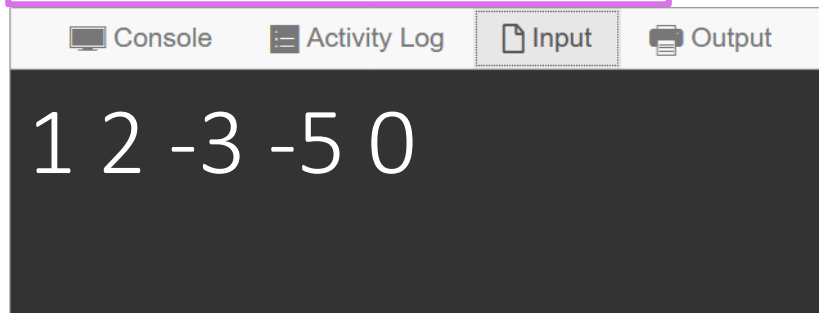


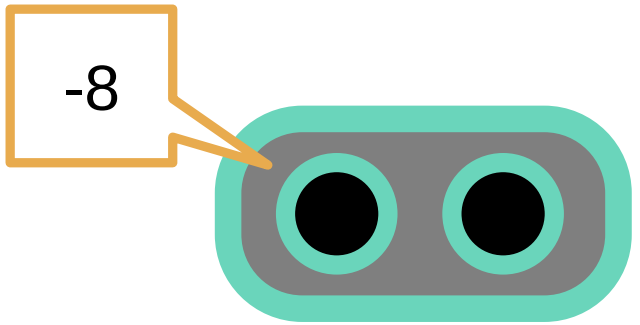


Some Tips on Correcting Errors

Print non-output variables as well – it is boring ☺ but useful

```
int sum = 0, i, num;
for(i = 1; i <= 5; i++){
    scanf("%d", &num);
    if (num > 0)
        continue;
    sum += num;
}
```

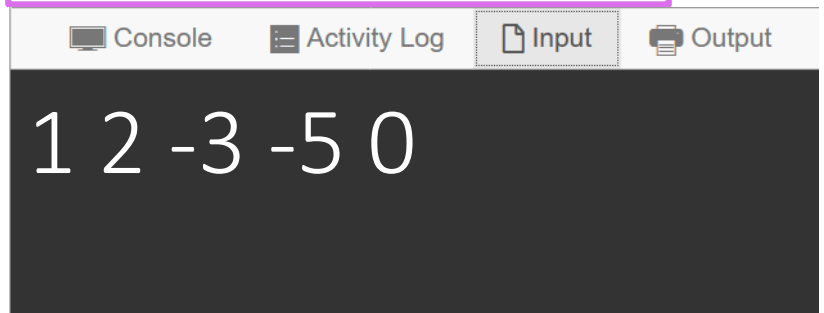


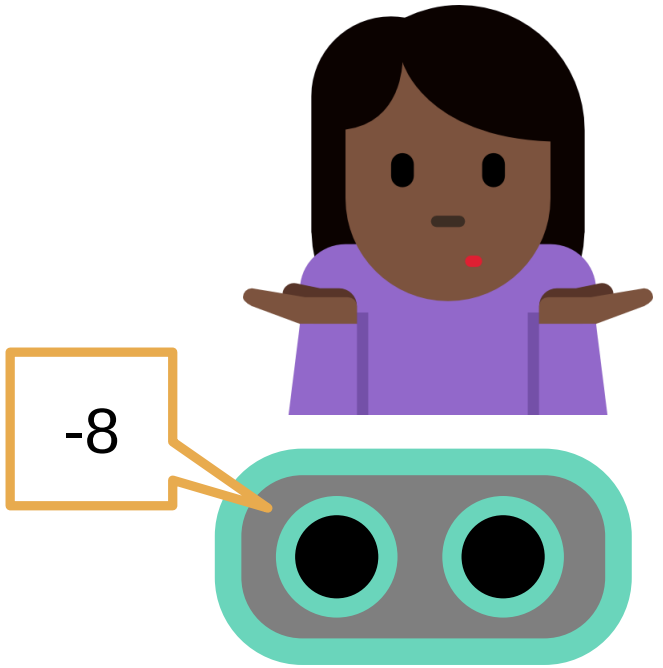


Some Tips on Correcting Errors

Print non-output variables as well – it is boring ☺ but useful

```
int sum = 0, i, num;
for(i = 1; i <= 5; i++){
    scanf("%d", &num);
    if (num > 0)
        continue;
    sum += num;
}
```





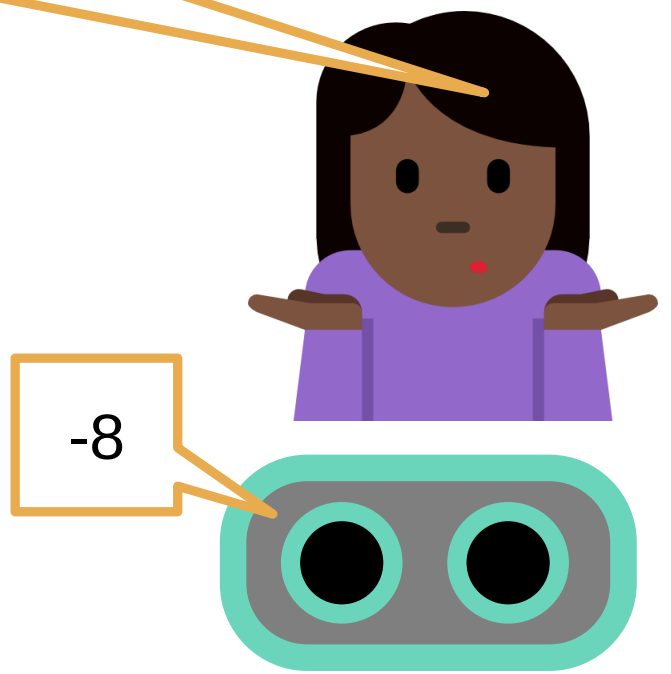
Some Tips on Correcting Errors

Print non-output variables as we[illegible] useful

```
int sum = 0, i, num;
for(i = 1; i <= 5; i++){
    scanf("%d", &num);
    if (num > 0)
        continue;
    sum += num;
}
```

Console | Activity Log | Input | Output

```
1 2 -3 -5 0
```



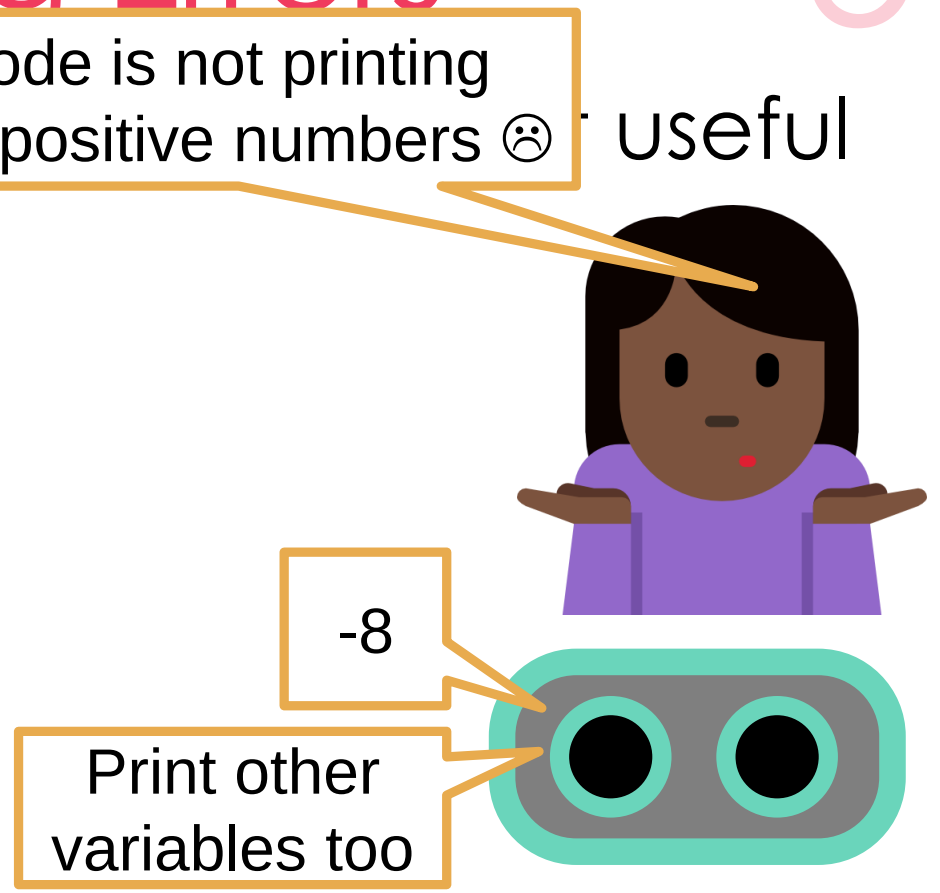
Some Tips on Correcting Errors

Print non-output variables as we[illegible] useful

```
int sum = 0, i, num;
for(i = 1; i <= 5; i++){
    scanf("%d", &num);
    if (num > 0)
        continue;
    sum += num;
}
```

Console | Activity Log | Input | Output

```
1 2 -3 -5 0
```



Some Tips on Correcting Errors

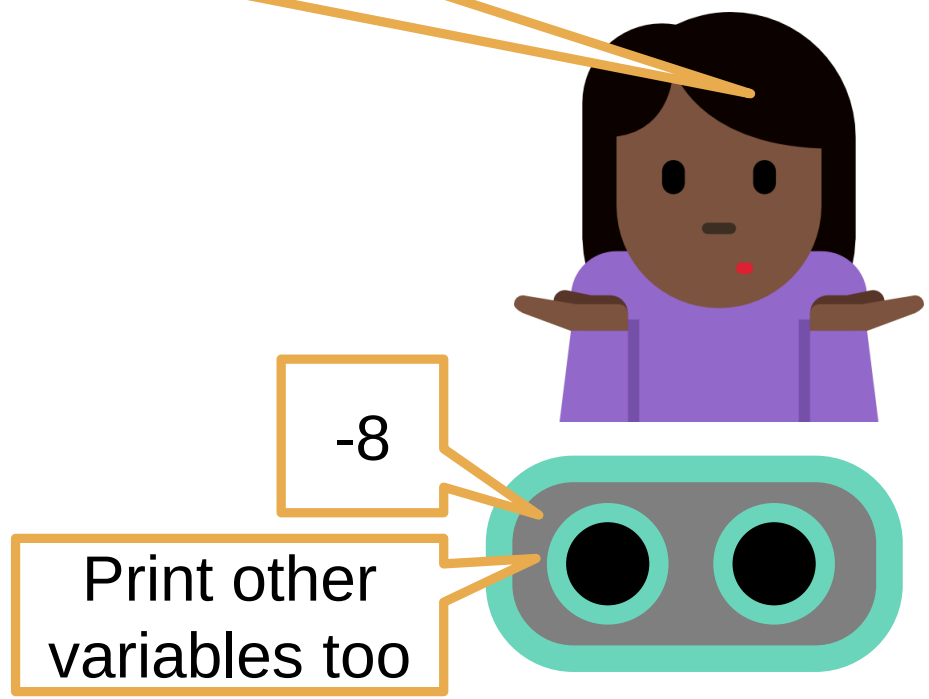
Print non-output variables as we ... useful

```
int sum = 0, i, num;
for(i = 1; i <= 5; i++){
    scanf("%d", &num);
    if (num > 0)
        continue;
    sum += num;
}
```

```
int sum = 0, i, num;
for(i = 1; i <= 5; i++){
    scanf("%d", &num);
    if (num > 0) continue;
    sum += num;
    printf("add %d\n", num);
}
```

Console | Activity Log | Input | Output

```
1 2 -3 -5 0
```



Some Tips on Correcting Errors

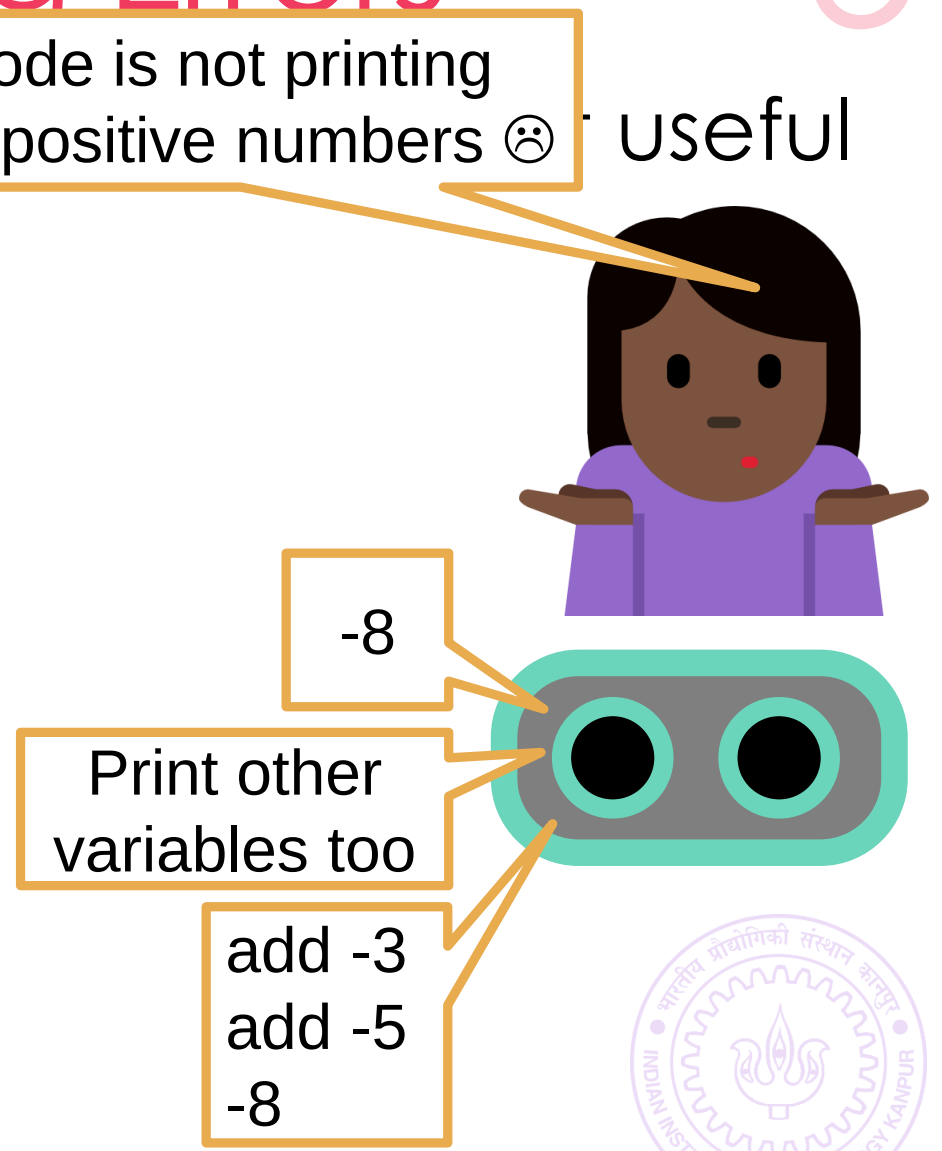
Print non-output variables as we[illegible] useful

```
int sum = 0, i, num;
for(i = 1; i <= 5; i++){
    scanf("%d", &num);
    if (num > 0)
        continue;
    sum += num;
}
```

```
int sum = 0, i, num;
for(i = 1; i <= 5; i++){
    scanf("%d", &num);
    if (num > 0) continue;
    sum += num;
    printf("add %d\n", num);
}
```

Console | Activity Log | Input | Output

```
1 2 -3 -5 0
```



Some Tips on Correcting Errors

Print non-output variables as we[illegible] useful

```
int sum = 0, i, num;
for(i = 1; i <= 5; i++){
    scanf("%d", &num);
    if (num > 0)
        continue;
    sum += num;
}
```

```
int sum = 0, i, num;
for(i = 1; i <= 5; i++){
    scanf("%d", &num);
    if (num > 0) continue;
    sum += num;
    printf("add %d\n", num);
}
```

Console | Activity Log | Input | Output

```
1 2 -3 -5 0
```

My code is not printing sum of positive numbers ☹

-8

Print other variables too

add -3
add -5
-8

Some Tips on Correcting Errors

Print non-output variables as we useful

```
int sum = 0, i, num;
for(i = 1; i <= 5; i++){
    scanf("%d", &num);
    if (num > 0)
        continue;
    sum += num;
}
```

```
int sum = 0, i, num;
for(i = 1; i <= 5; i++){
    scanf("%d", &num);
    if (num > 0) continue;
    sum += num;
    printf("add %d\n", num);
}
```

Console | Activity Log | Input | Output

```
1 2 -3 -5 0
```

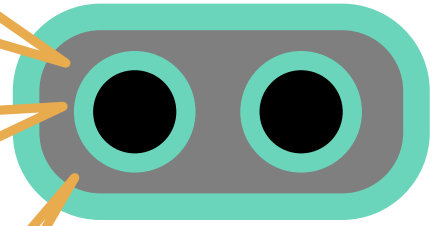
My code is not printing sum of positive numbers ☹

My relational expression in if statement is wrong!

-8

Print other variables too

```
add -3
add -5
-8
```



Some Tips on Correcting Errors

Print non-output variables as we useful

```
int sum = 0, i, num;
for(i = 1; i <= 5; i++){
    scanf("%d", &num);
    if (num > 0)
        continue;
    sum += num;
}
```

```
int sum = 0, i, num;
for(i = 1; i <= 5; i++){
    scanf("%d", &num);
    if (num > 0) continue;
    sum += num;
    printf("add %d\n", num);
}
```

Console | Activity Log | Input | Output

```
1 2 -3 -5 0
```

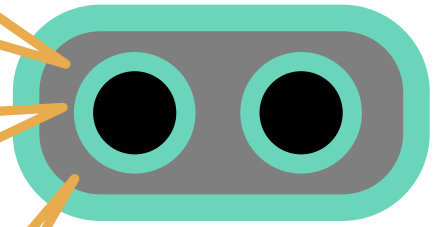
My code is not printing sum of positive numbers ☹

My relational expression in if statement is wrong!

-8

Print other variables too

add -3
add -5
-8



Some Tips on Correcting Errors

Some Tips on Correcting Errors

Execute your own code in your own mind to see why code is not working as expected

Some Tips on Correcting Errors

Execute your own code in your own mind to see why code is not working as expected

This step is known as *tracing the program*

Some Tips on Correcting Errors

Execute your own code in your own mind to see why code is not working as expected

This step is known as *tracing the program*

Just as you do in the "What will be the output" style questions in minor quiz.

Some Tips on Correcting Errors

Execute your own code in your own mind to see why code is not working as expected

This step is known as *tracing the program*

Just as you do in the "What will be the output" style questions in minor quiz.

Do this on simple inputs first otherwise tracing will become very confusing and difficult

Some Tips on Correcting Errors

Execute your own code in your own mind to see why code is not working as expected

This step is known as *tracing the program*

Just as you do in the "What will be the output" style questions in minor quiz.

Do this on simple inputs first otherwise tracing will become very confusing and difficult

Prutor offers a visualizer (Run > Visualize) – not a magical tool, just helps you trace your program more comfortably

Some Tips on Correcting Errors

Execute your own code in your own mind to see why code is not working as expected

This step is known as *tracing the program*

Just as you do in the "What will be the output" style questions in minor quiz.

Do this on simple inputs first otherwise tracing will become very confusing and difficult

Prutor offers a visualizer (Run > Visualize) – not a magical tool, just helps you trace your program more comfortably

Other tools called *debuggers* also widely used

Some Tips on Correcting Errors

Some Tips on Correcting Errors

If there is some portion of code not getting executed, write a printf statement there printing “Hello” or “XYZ”

Some Tips on Correcting Errors

If there is some portion of code not getting executed, write a printf statement there printing “Hello” or “XYZ”

Will act as a flag telling you whether that piece of code is indeed getting executed or not

Some Tips on Correcting Errors

If there is some portion of code not getting executed, write a printf statement there printing “Hello” or “XYZ”

Will act as a flag telling you whether that piece of code is indeed getting executed or not

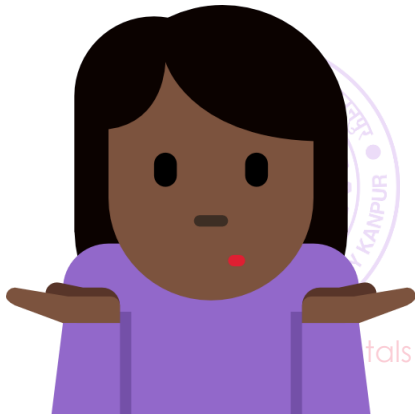
```
int sum = 0, i, j, num;
for(i = 1; i <= 5; i++){
    j = i % 2;
    if(j = 0)
        sum += num;
}
```

Some Tips on Correcting Errors

If there is some portion of code not getting executed, write a printf statement there printing “Hello” or “XYZ”

Will act as a flag telling you whether that piece of code is indeed getting executed or not

```
int sum = 0, i, j, num;
for(i = 1; i <= 5; i++){
    j = i % 2;
    if(j = 0)
        sum += num;
}
```



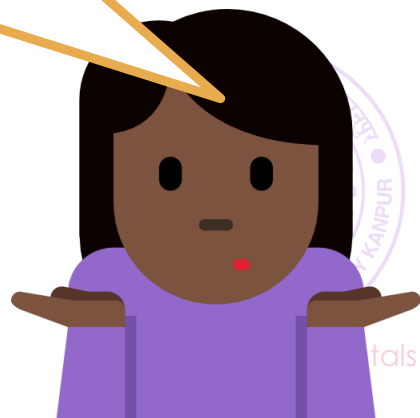
Some Tips on Correcting Errors

If there is some portion of code not getting executed, write a printf statement there printing “Hello” or “XYZ”

Will act as a flag telling you whether that piece of code is indeed getting executed or not

```
int sum = 0, i, j, num;
for(i = 1; i <= 5; i++){
    j = i % 2;
    if(j = 0)
        sum += num;
}
```

I feel that body of if statement is not getting executed at all!



Some Tips on Correcting Errors

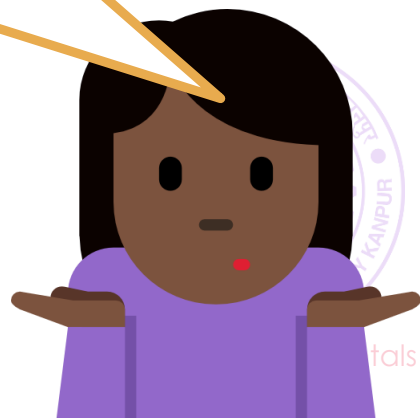
If there is some portion of code not getting executed, write a printf statement there printing “Hello” or “XYZ”

Will act as a flag telling you whether that piece of code is indeed getting executed or not

```
int sum = 0, i, j, num;
for(i = 1; i <= 5; i++){
    j = i % 2;
    if(j = 0)
        sum += num;
}
```

```
int sum = 0, i, j, num;
for(i = 1; i <= 5; i++){
    j = i % 2;
    if(j = 0){
        sum += num;
        printf(“Inside if\n”);
}}
```

I feel that body of if statement is not getting executed at all!



Some Tips on Correcting Errors

If there is some portion of code not getting executed, write a printf statement there printing “Hello” or “XYZ”

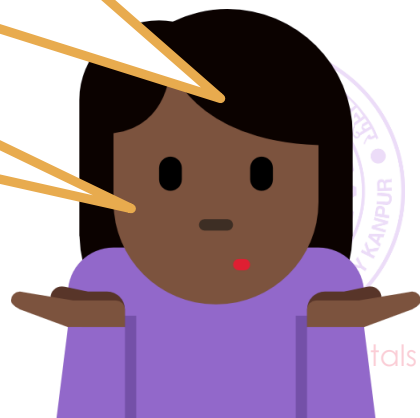
Will act as a flag telling you whether that piece of code is indeed getting executed or not

```
int sum = 0, i, j, num;
for(i = 1; i <= 5; i++){
    j = i % 2;
    if(j = 0)
        sum += num;
}
```

```
int sum = 0, i, j, num;
for(i = 1; i <= 5; i++){
    j = i % 2;
    if(j = 0){
        sum += num;
        printf(“Inside if\n”);
}}
```

I feel that body of if statement is not getting executed at all!

Hmm … “Inside if” never got printed …



Some Tips on Correcting Errors

If there is some portion of code not getting executed, write a printf statement there printing “Hello” or “XYZ”

Will act as a flag telling you whether that piece of code is indeed getting executed or not

```
int sum = 0, i, j, num;
for(i = 1; i <= 5; i++){
    j = i % 2;
    if(j = 0)
        sum += num;
}
```

```
int sum = 0, i, j, num;
for(i = 1; i <= 5; i++){
    j = i % 2;
    if(j = 0){
        sum += num;
        printf(“Inside if\n”);
}}
```

I feel that body of if statement is not getting executed at all!

Hmm … “Inside if” never got printed …

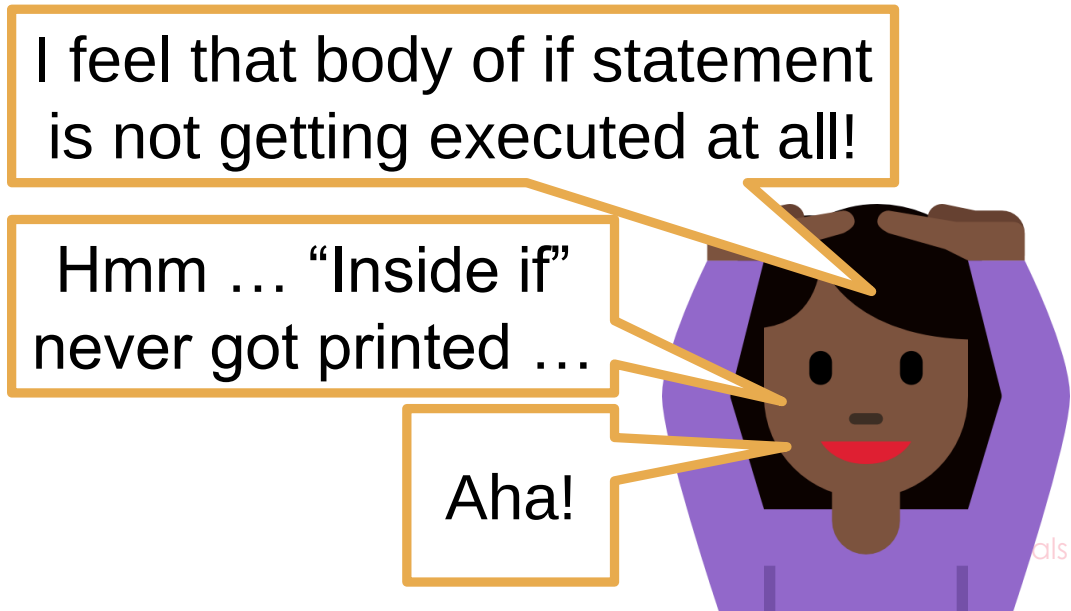
Some Tips on Correcting Errors

If there is some portion of code not getting executed, write a printf statement there printing “Hello” or “XYZ”

Will act as a flag telling you whether that piece of code is indeed getting executed or not

```
int sum = 0, i, j, num;
for(i = 1; i <= 5; i++){
    j = i % 2;
    if(j = 0)
        sum += num;
}
```

```
int sum = 0, i, j, num;
for(i = 1; i <= 5; i++){
    j = i % 2;
    if(j = 0){
        sum += num;
        printf(“Inside if\n”);
}}
```



Some Tips on Correcting Errors

If there is some portion of code not getting executed, write a printf statement there printing “Hello” or “XYZ”

Will act as a flag telling you whether that piece of code is indeed getting executed or not

```
int sum = 0, i, j, num;
for(i = 1; i <= 5; i++){
    j = i % 2;
    if(j = 0)
        sum += num;
}
```

```
int sum = 0, i, j, num;
for(i = 1; i <= 5; i++){
    j = i % 2;
    if(j = 0){
        sum += num;
        printf(“Inside if\n”);
}}
```

I feel that body of if statement is not getting executed at all!

Hmm … “Inside if” never got printed …

Aha!

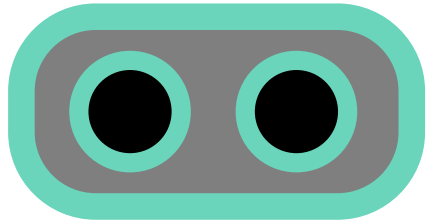
Some Tips on Correcting Errors

If there is some portion of code not getting executed, write a printf statement there printing “Hello” or “XYZ”

Will act as a flag telling you whether that piece of code is indeed getting executed or not

```
int sum = 0, i, j, num;
for(i = 1; i <= 5; i++){
    j = i % 2;
    if(j = 0)
        sum += num;
}
```

```
int sum = 0, i, j, num;
for(i = 1; i <= 5; i++){
    j = i % 2;
    if(j = 0){
        sum += num;
        printf(“Inside if\n”);
}}
```



I feel that body of if statement is not getting executed at all!

Hmm … “Inside if” never got printed …

Aha!

Some Tips on Correcting Errors

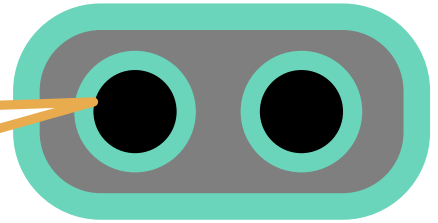
If there is some portion of code not getting executed, write a printf statement there printing “Hello” or “XYZ”

Will act as a flag telling you whether that piece of code is indeed getting executed or not

```
int sum = 0, i, j, num;
for(i = 1; i <= 5; i++){
    j = i % 2;
    if(j = 0)
        sum += num;
}
```

```
int sum = 0, i, j, num;
for(i = 1; i <= 5; i++){
    j = i % 2;
    if(j = 0){
        sum += num;
        printf(“Inside if\n”);
}}
```

Bad indentation ☹



I feel that body of if statement is not getting executed at all!

Hmm … “Inside if” never got printed …

Aha!

Some Tips on Correcting Errors

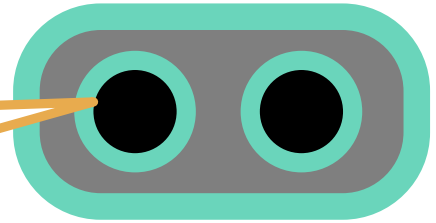
If there is some portion of code not getting executed, write a printf statement there printing “Hello” or “XYZ”

Will act as a flag telling you whether that piece of code is indeed getting executed or not

```
int sum = 0, i, j, num;
for(i = 1; i <= 5; i++){
    j = i % 2;
    if(j = 0)
        sum += num;
}
```

```
int sum = 0, i, j, num;
for(i = 1; i <= 5; i++){
    j = i % 2;
    if(j = 0){
        sum += num;
        printf(“Inside if\n”);
}}
```

Bad indentation ☹



I feel that body of if statement is not getting executed at all!

Hmm … “Inside if” never got printed …

Aha!

Some Tips on Correcting Errors

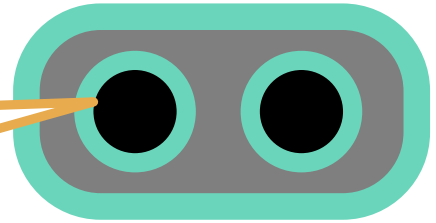
If there is some portion of code not getting executed, write a printf statement there printing “Hello” or “XYZ”

Will act as a flag telling you whether that piece of code is indeed getting executed or not

```
int sum = 0, i, j, num;
for(i = 1; i <= 5; i++){
    j = i % 2;
    if(j = 0)
        sum += num;
}
```

```
int sum = 0, i, j, num;
for(i = 1; i <= 5; i++){
    j = i % 2;
    if(j = 0){
        sum += num;
        printf(“Inside if\n”);
}}
```

Bad indentation ☹



I feel that body of if statement is not getting executed at all!

Hmm … “Inside if” never got printed …

Sorry!

Aha!

How to debug infinite loops

How to debug infinite loops

printf trick won't work if infinite loop – nothing gets printed at all!

How to debug infinite loops

printf trick won't work if infinite loop – nothing gets printed at all!

Avoid adventures like infinite loops if possible ☺

How to debug infinite loops

printf trick won't work if infinite loop – nothing gets printed at all!

Avoid adventures like infinite loops if possible ☺

```
int num = 10, i;
for(i = 1; ;){
    if(!(i = num))
        break;
    else i++;
}
```

How to debug infinite loops

printf trick won't work if infinite loop – nothing gets printed at all!

Avoid adventures like infinite loops if possible ☺

Temporarily restrict loop to terminate after, say 1000 iterations

```
int num = 10, i;
for(i = 1; ;){
    if(!(i = num))
        break;
    else i++;
}
```

How to debug infinite loops

printf trick won't work if infinite loop – nothing gets printed at all!

Avoid adventures like infinite loops if possible ☺

Temporarily restrict loop to terminate after, say 1000 iterations

Add printf statements to find out what is going on!

```
int num = 10, i;
for(i = 1; ;){
    if(!(i = num))
        break;
    else i++;
}
```

How to debug infinite loops

printf trick won't work if infinite loop – nothing gets printed at all!

Avoid adventures like infinite loops if possible ☺

Temporarily restrict loop to terminate after, say 1000 iterations

Add printf statements to find out what is going on!

```
int num = 10, i;
for(i = 1; ;){
    if(!(i = num))
        break;
    else i++;
}
```

```
int num = 10, i, j = 0;
for(i = 1; j < 1000;){
    j++;
    if(!(i = num))
        break;
    else{
        i++;
        printf("At %d\n",i);
    }
}
```

How to debug infinite loops

printf trick won't work if infinite loop – nothing gets printed at all!

Avoid adventures like infinite loops if possible

Temporarily restrict loop to terminate after, say 1000 iterations

Add printf statements to find out what is going on!

```
int num = 10, i;
for(i = 1; ;){
m))
}
```

Safest to increment this dummy counter variable j as the first statement in body

```
int num = 10, i, j = 0;
for(i = 1; j < 1000;){
    j++;
    if(!(i = num))
        break;
    else{
        i++;
        printf("At %d\n",i);
    }
}
```

How to debug infinite loops

printf trick won't work if infinite loop – nothing gets printed at all!

Avoid adventures like infinite loops if possible

Temporarily restrict loop to terminate after, say 1000 iterations

Add printf statements to find out what is going on!

```
int num = 10, i;
for(i = 1; :){
```

```
int num = 10, i, j = 0;
for(i = 1; j < 1000;){
    j++;
    if(!(i = num))
        break;
    else{
        j++;
        printf("At %d\n",i);
    }
}
```

Safest to increment this dummy counter variable j as the first statement in body

If you put j++ here, it might get skipped due to a wrong continue statement

How to debug infinite loops

printf trick won't work if infinite loop – nothing gets printed at all!

Avoid adventures like infinite loops if possible

Temporarily restrict loop to terminate after, say 1000 iterations

Add printf statements to find out what is going on!

Once error found, remove restrictions

```
int num = 10, i;
for(i = 1; ;){
```

```
int num = 10, i, j = 0;
for(i = 1; j < 1000;){
    j++;
    if(!(i = num))
        break;
    else{
        j++;
        printf("At %d\n",i);
    }
}
```

Safest to increment this dummy counter variable j as the first statement in body

If you put j++ here, it might get skipped due to a wrong continue statement

How to debug infinite loops

printf trick won't work if infinite loop – nothing gets printed at all!

Avoid adventures like infinite loops if possible ☺

Temporarily restrict loop to terminate after, say 1000 iterations

Add printf statements to find out what is going on!

Once error found, remove restrictions

```
int num = 10, i;
for(i = 1; ;){
    if(!(i = num))
        break;
    else i++;
}
```

```
int num = 10, i;
for(i = 1; ;){

    if(!(i == num))
        break;
    else{
        i++;

    }
}
```

Reverse the Stream

Reverse the Stream

Read 100 numbers from the input and print them back in reverse order

Reverse the Stream

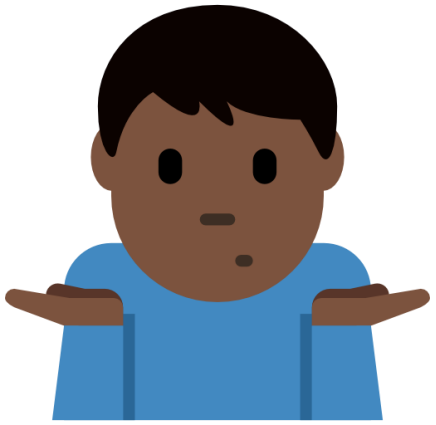
Read 100 numbers from the input and print them back in reverse order

```
int a1, a2, a3, .... ,a100;
scanf("%d", &a1);
scanf("%d", &a2);
...
scanf("%d", &a100);


printf("%d", a100);
printf("%d", a99);
...
printf("%d", a1);
```

Reverse the Stream

Read 100 numbers from the input and print them back in reverse order

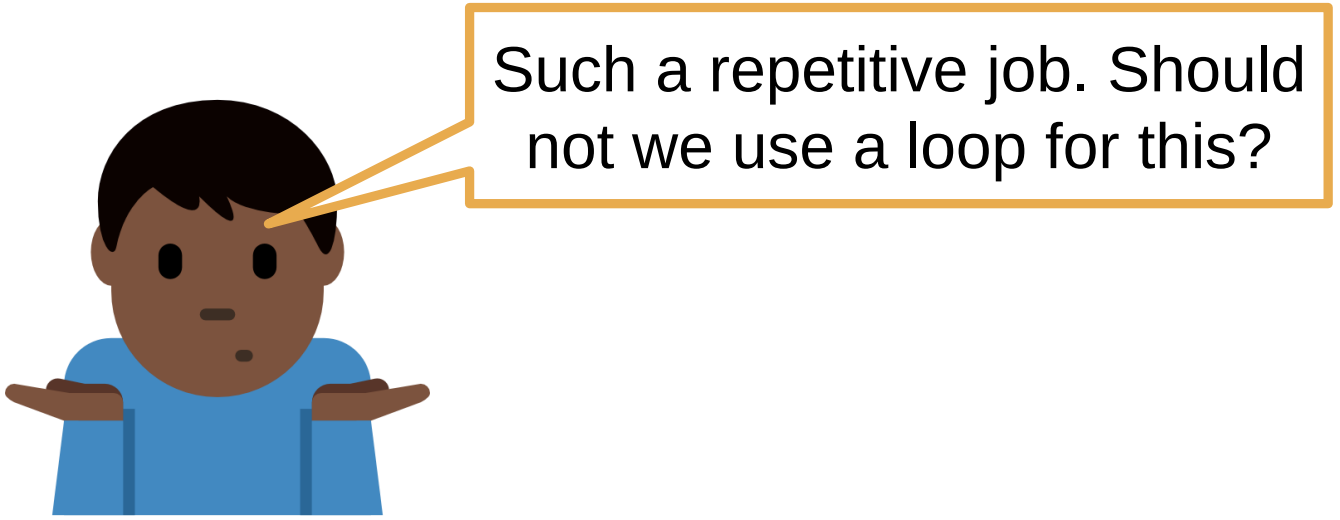
```
int a1, a2, a3, .... ,a100;
scanf("%d", &a1);
scanf("%d", &a2);
...
scanf("%d", &a100);


printf("%d", a100);
printf("%d", a99);
...
printf("%d", a1);
```



Reverse the Stream

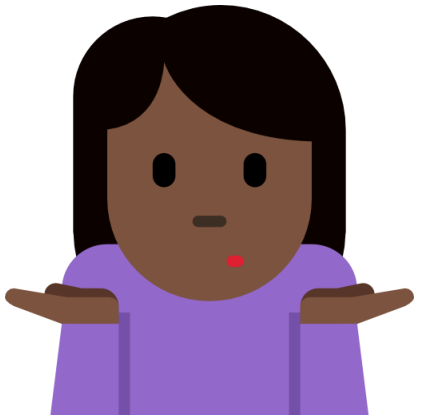
Read 100 numbers from the input and print them back in reverse order

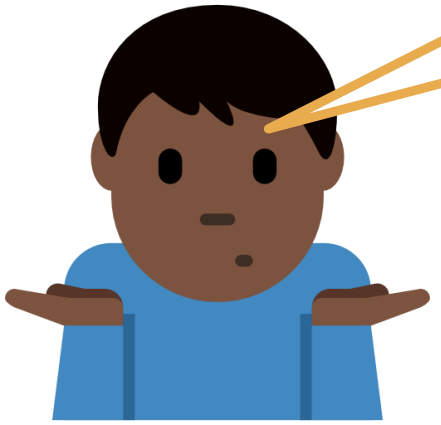
```
int a1, a2, a3, .... ,a100;
scanf("%d", &a1);
scanf("%d", &a2);
...
scanf("%d", &a100);


printf("%d", a100);
printf("%d", a99);
...
printf("%d", a1);
```



Reverse the Stream

Read 100 numbers from the input and print them back in reverse order



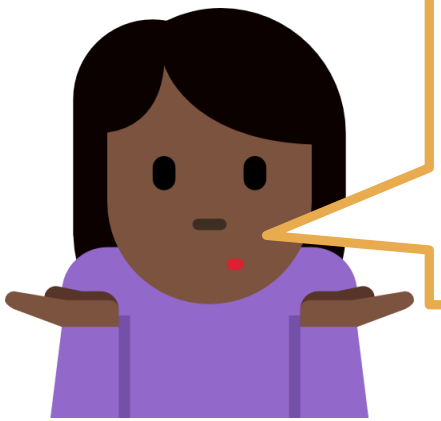


Such a repetitive job. Should not we use a loop for this?

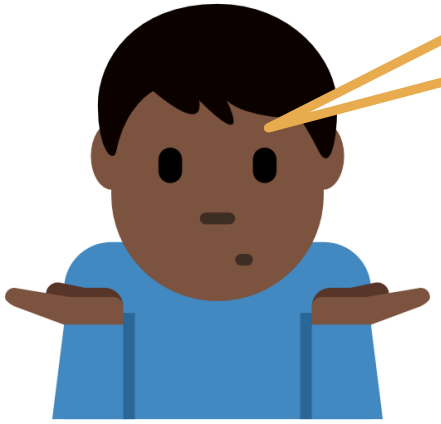
```
int a1, a2, a3, .... ,a100;
scanf("%d", &a1);
scanf("%d", &a2);
...
scanf("%d", &a100);


printf("%d", a100);
printf("%d", a99);
...
printf("%d", a1);
```

Reverse the Stream

Read 100 numbers from the input and print them back in reverse order



Yes but if I write
for(i = 1; i <=100; i++) scanf("%d",&a);
for(i = 100; i >=1; i--) printf("%d",a);
Only the last number gets printed 100 times

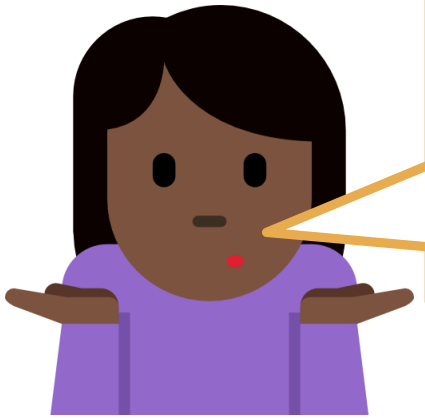


Such a repetitive job. Should not we use a loop for this?

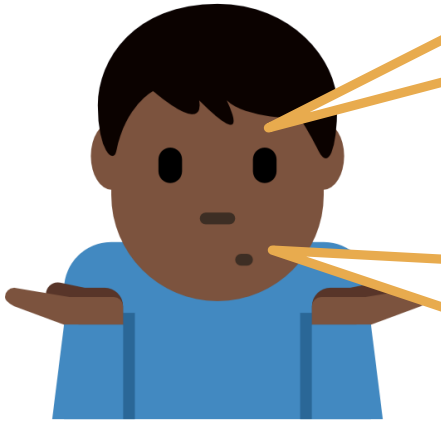
```
int a1, a2, a3, .... ,a100;
scanf("%d", &a1);
scanf("%d", &a2);
...
scanf("%d", &a100);


printf("%d", a100);
printf("%d", a99);
...
printf("%d", a1);
```

Reverse the Stream

Read 100 numbers from the input and print them back in reverse order



Yes but if I write
for(i = 1; i <=100; i++) scanf("%d",&a);
for(i = 100; i >=1; i--) printf("%d",a);
Only the last number gets printed 100 times



Such a repetitive job. Should not we use a loop for this?

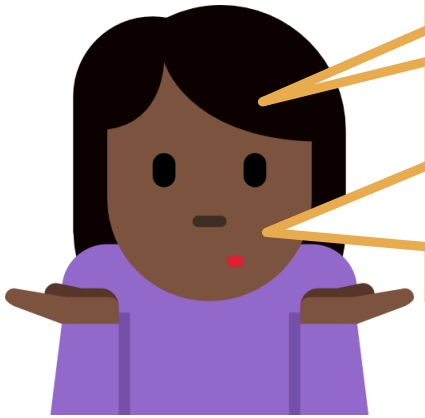
You are right … I cannot write printf("%d",ai); inside the loop since there is no variable with the name ai.

```
int a1, a2, a3, .... ,a100;
scanf("%d", &a1);
scanf("%d", &a2);
...
scanf("%d", &a100);


printf("%d", a100);
printf("%d", a99);
...
printf("%d", a1);
```

Reverse the Stream

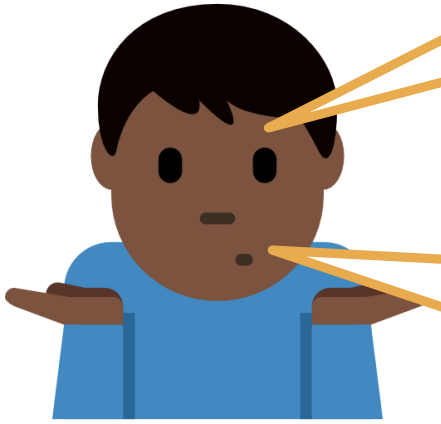
Read 100 ... nd print them back in reverse o...

Wouldn't it be nice to have a way of naming 100 variables so that something like printf("%d",ai); makes sense as i goes from 1 to 100!



Yes but if I write
for(i = 1; i <=100; i++) scanf("%d",&a);
for(i = 100; i >=1; i--) printf("%d",a);
Only the last number gets printed 100 times

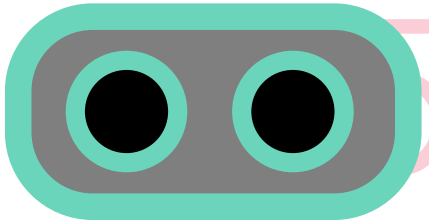
Such a repetitive job. Should not we use a loop for this?



You are right … I cannot write printf("%d",ai); inside the loop since there is no variable with the name ai.

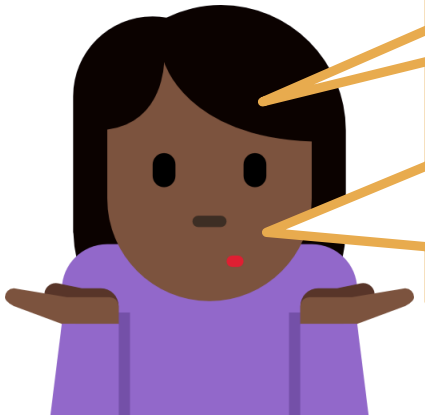
```
int a1, a2, a3, .... ,a100;
scanf("%d", &a1);
scanf("%d", &a2);
...
scanf("%d", &a100);


printf("%d", a100);
printf("%d", a99);
...
printf("%d", a1);
```

Reverse the Stream

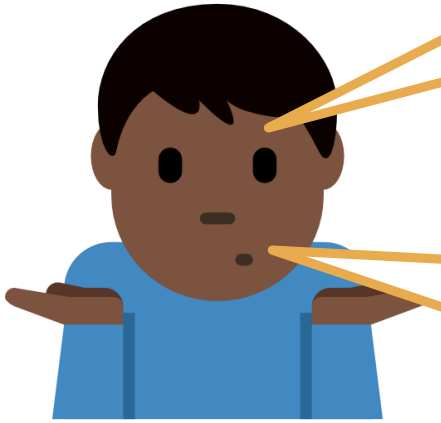


Read 100 ... nd print them back in reverse o...

Wouldn't it be nice to have a way of naming 100 variables so that something like printf("%d",ai); makes sense as i goes from 1 to 100!



Yes but if I write
for(i = 1; i <=100; i++) scanf("%d",&a);
for(i = 100; i >=1; i--) printf("%d",a);
Only the last number gets printed 100 times



Such a repetitive job. Should not we use a loop for this?

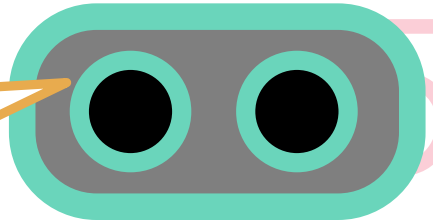
You are right … I cannot write printf("%d",ai); inside the loop since there is no variable with the name ai.

```
int a1, a2, a3, .... ,a100;
scanf("%d", &a1);
scanf("%d", &a2);
...
scanf("%d", &a100);


printf("%d", a100);
printf("%d", a99);
...
printf("%d", a1);
```

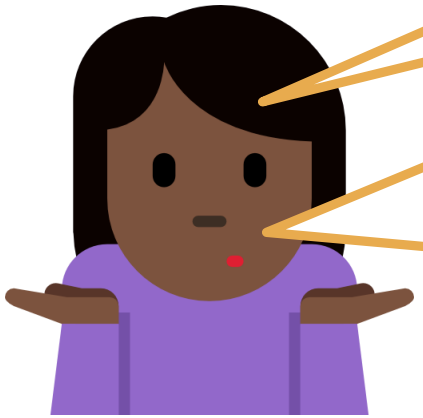
Reverse the Stream

Read 100 ... nd print them back in reverse o...

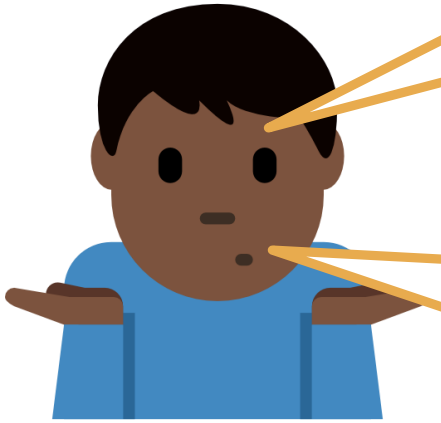
I have just the thing for you



Wouldn't it be nice to have a way of naming 100 variables so that something like printf("%d",ai); makes sense as i goes from 1 to 100!



Yes but if I write
for(i = 1; i <=100; i++) scanf("%d",&a);
for(i = 100; i >=1; i--) printf("%d",a);
Only the last number gets printed 100 times



Such a repetitive job. Should not we use a loop for this?

You are right … I cannot write printf("%d",ai); inside the loop since there is no variable with the name ai.

```
int a1, a2, a3, .... ,a100;
scanf("%d", &a1);
scanf("%d", &a2);
...
scanf("%d", &a100);


printf("%d", a100);
printf("%d", a99);
...
printf("%d", a1);
```

Arrays

भारतीय प्रौद्योगिकी संस्थान कानपुर
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

Arrays

The English word array means "objects in a line" or "an ordered series or arrangement" – *The soldiers standing in an array impressed the visiting head of the state on 26 Jan*

Arrays

The English word array means "objects in a line" or "an ordered series or arrangement" – *The soldiers standing in an array impressed the visiting head of the state on 26 Jan*

For Mr C, an array is a sequence of variables with very convenient names - can have an array of ints, longs, floats, doubles

Arrays

The English word array means "objects in a line" or "an ordered series or arrangement" – *The soldiers standing in an array impressed the visiting head of the state on 26 Jan*

For Mr C, an array is a sequence of variables with very convenient names - can have an array of ints, longs, floats, doubles

```
int a[6];
```

Arrays

The English word array means "objects in a line" or "an ordered series or arrangement" – *The soldiers standing in an array impressed the visiting head of the state on 26 Jan*

For Mr C, an array is a sequence of variables with very convenient names - can have an array of ints, longs, floats, doubles

```
int a[6];
```

Arrays

The English word array means "objects in a line" or "an ordered series or arrangement" – *The soldiers standing in an array impressed the visiting head of the state on 26 Jan*

For Mr C, an array is a sequence of variables with very convenient names - can have an array of ints, longs, floats, doubles

```
int a[6];
```

Arrays

The English word array means "objects in a line" or "an ordered series or arrangement" – *The soldiers standing in an array impressed the visiting head of the state on 26 Jan*

For Mr C, an array is a sequence of variables with very convenient names - can have an array of ints, longs, floats, doubles

```
int a[6];
```

Arrays

The English word array means "objects in a line" or "an ordered series or arrangement" – *The soldiers standing in an array impressed the visiting head of the state on 26 Jan*

For Mr C, an array is a sequence of variables with very convenient names - can have an array of ints, longs, floats, doubles

```
int a[6];
```

Arrays

The English word array means "objects in a line" or "an ordered series or arrangement" – *The soldiers standing in an array impressed the visiting head of the state on 26 Jan*

For Mr C, an array is a sequence of variables with very convenient names - can have an array of ints, longs, floats, doubles

```
int a[6];
```

Arrays

The English word array means "objects in a line" or "an ordered series or arrangement" – *The soldiers standing in an array impressed the visiting head of the state on 26 Jan*

For Mr C, an array is a sequence of variables with very convenient names - can have an array of ints, longs, floats, doubles

```
int a[6];
```

Arrays

The English word array means "objects in a line" or "an ordered series or arrangement" – *The soldiers standing in an array impressed the visiting head of the state on 26 Jan*

For Mr C, an array is a sequence of variables with very convenient names - can have an array of ints, longs, floats, doubles

```
int a[6];
```

a

Arrays

The English word array means "objects in a line" or "an ordered series or arrangement" – *The soldiers standing in an array impressed the visiting head of the state on 26 Jan*

For Mr C, an array is a sequence of variables with very convenient names - can have an array of ints, longs, floats, doubles

```
int a[6];
```

a

a[0]

Arrays

The English word array means "objects in a line" or "an ordered series or arrangement" – *The soldiers standing in an array impressed the visiting head of the state on 26 Jan*

For Mr C, an array is a sequence of variables with very convenient names - can have an array of ints, longs, floats, doubles

```
int a[6];
```

a

Arrays

The English word array means "objects in a line" or "an ordered series or arrangement" – *The soldiers standing in an array impressed the visiting head of the state on 26 Jan*

For Mr C, an array is a sequence of variables with very convenient names - can have an array of ints, longs, floats, doubles

```
int a[6];
```

a

Arrays

The English word array means "objects in a line" or "an ordered series or arrangement" – *The soldiers standing in an array impressed the visiting head of the state on 26 Jan*

For Mr C, an array is a sequence of variables with very convenient names - can have an array of ints, longs, floats, doubles

```
int a[6];
```

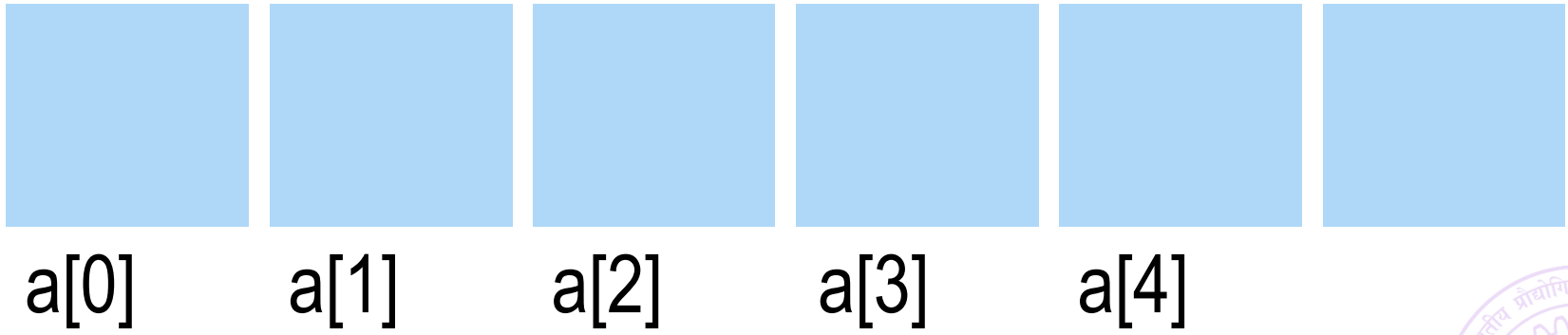
a

Arrays

The English word array means "objects in a line" or "an ordered series or arrangement" – *The soldiers standing in an array impressed the visiting head of the state on 26 Jan*

For Mr C, an array is a sequence of variables with very convenient names - can have an array of ints, longs, floats, doubles

int a[6];

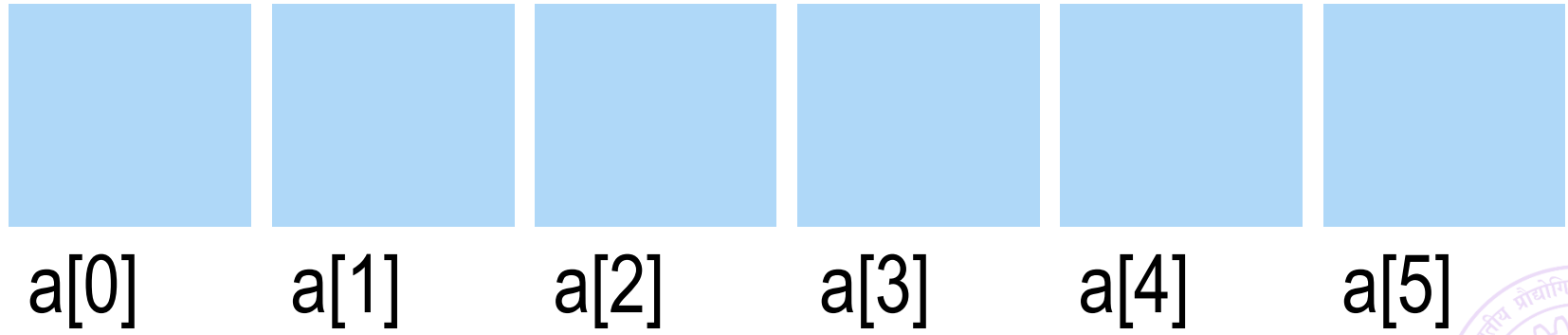


Arrays

The English word array means "objects in a line" or "an ordered series or arrangement" – *The soldiers standing in an array impressed the visiting head of the state on 26 Jan*

For Mr C, an array is a sequence of variables with very convenient names - can have an array of ints, longs, floats, doubles

int a[6];



Arrays

The English word array means "objects in a line" or "ordered series or arrangement" – *The soldiers stan… an array impressed the visiting head of the state on 26 Jan*

For Mr C, an array is a sequence of variables with very convenient names - can have an array of ints, longs, floats, doubles

```
int a[6];
```

a

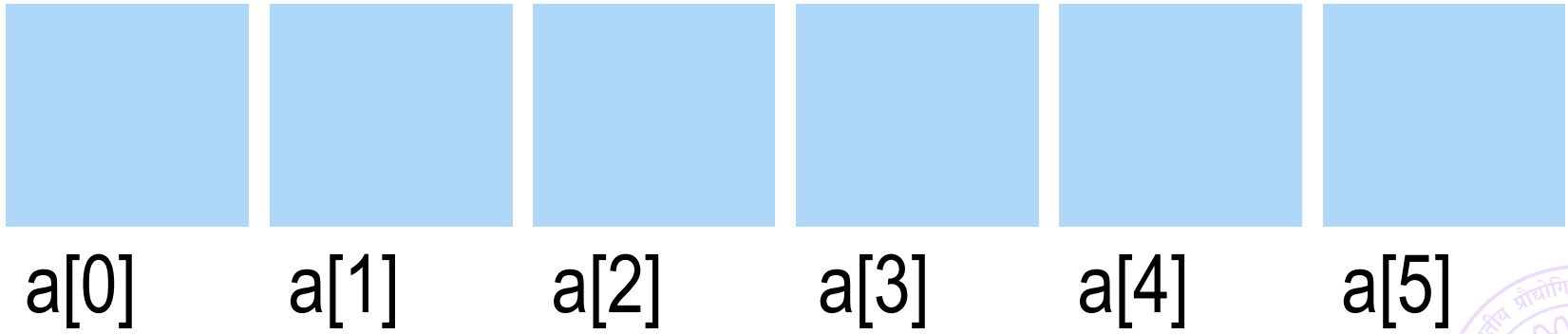
Arrays

a is like a street on which here are several houses with addresses a[0], a[3] etc

The English word array means "objects in a line" or "ordered series or arrangement" – *The soldiers stan… an array impressed the visiting head of the state on 26 Jan*

For Mr C, an array is a sequence of variables with very convenient names - can have an array of ints, longs, floats, doubles

int a[6];



Arrays

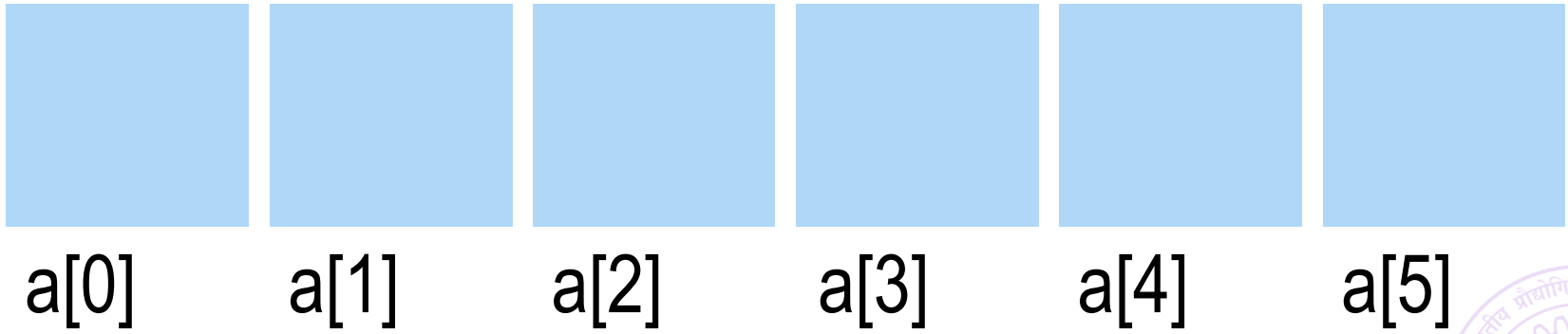
a is like a street on which here are several houses with addresses a[0], a[3] etc

The English word array means "objects in a line" or "objects in an ordered series or arrangement" – *The soldiers stan[illegible] an array impressed the visiting head of the state on 26 Jan*

For Mr C, an array is a sequence of variables with very convenient names - can have an array of ints, longs, floats, doubles

```
int a[6];
a[2] = 7;
```

a



Arrays

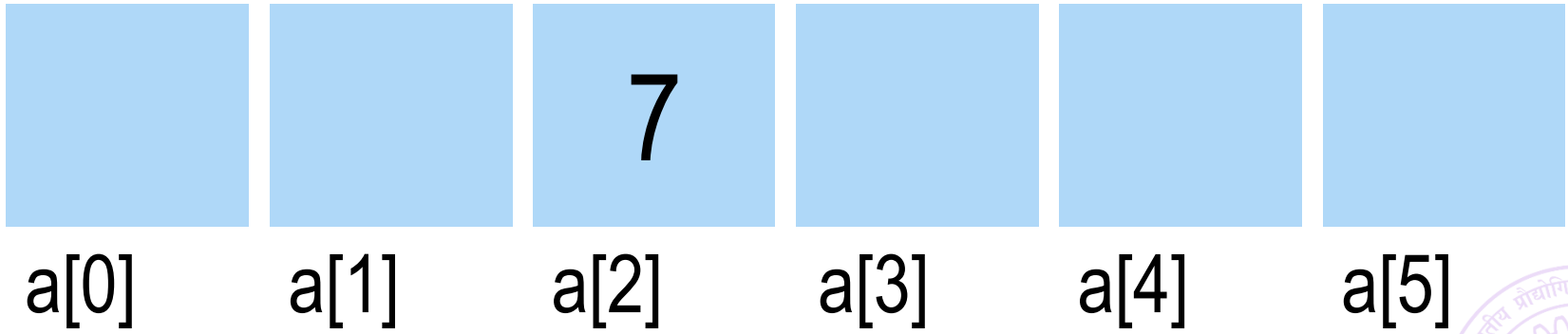
a is like a street on which here are several houses with addresses a[0], a[3] etc

The English word array means "objects in a line" or "ordered series or arrangement" – *The soldiers stan… an array impressed the visiting head of the state on 26 Jan*

For Mr C, an array is a sequence of variables with very convenient names - can have an array of ints, longs, floats, doubles

```
int a[6];
a[2] = 7;
```

a



Arrays

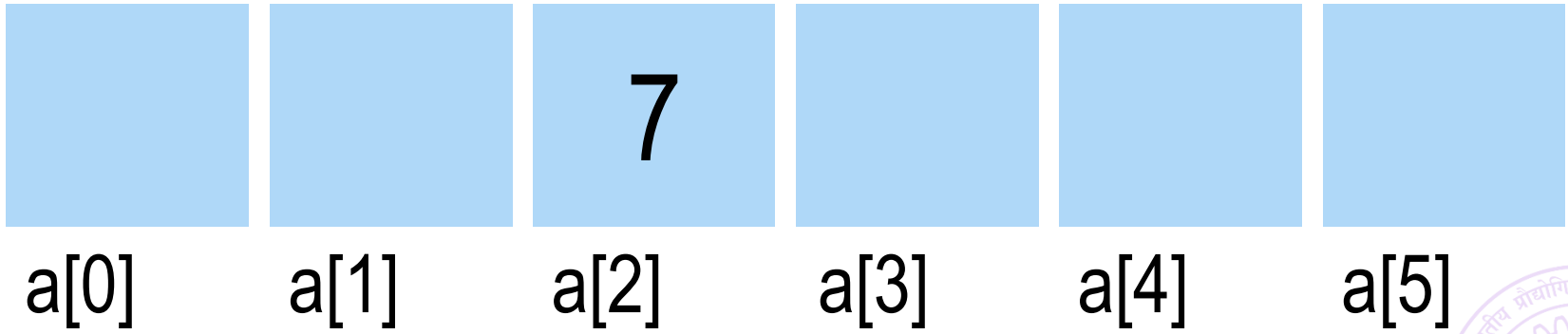
a is like a street on which here are several houses with addresses a[0], a[3] etc

The English word array means "objects in a line" or "ordered series or arrangement" – *The soldiers stan… an array impressed the visiting head of the state on 26 Jan*

For Mr C, an array is a sequence of variables with very convenient names - can have an array of ints, longs, floats, doubles

```
int a[6];
a[2] = 7;
a[4] = 3;
```

a



Arrays

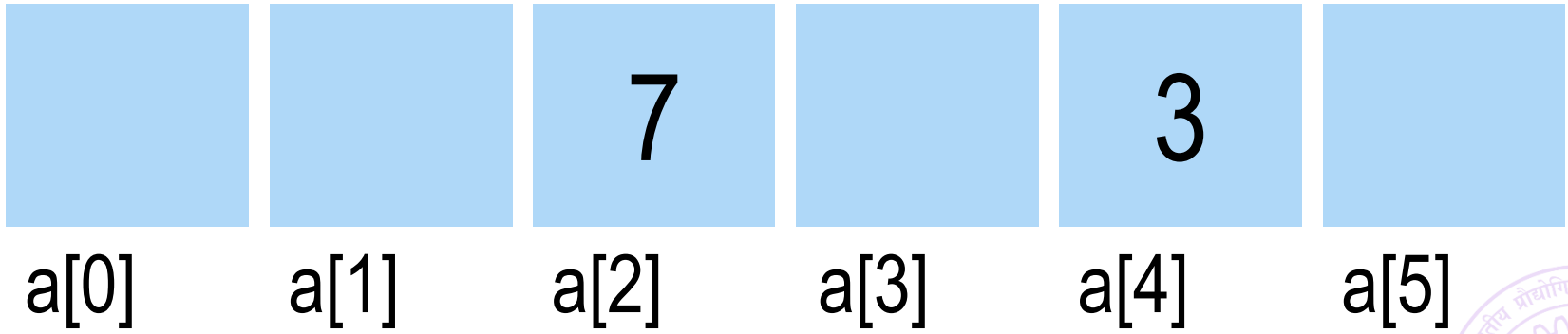
a is like a street on which here are several houses with addresses a[0], a[3] etc

The English word array means "objects in a line" or "ordered series or arrangement" – *The soldiers stan… an array impressed the visiting head of the state on 26 Jan*

For Mr C, an array is a sequence of variables with very convenient names - can have an array of ints, longs, floats, doubles

```
int a[6];
a[2] = 7;
a[4] = 3;
```

a



Arrays

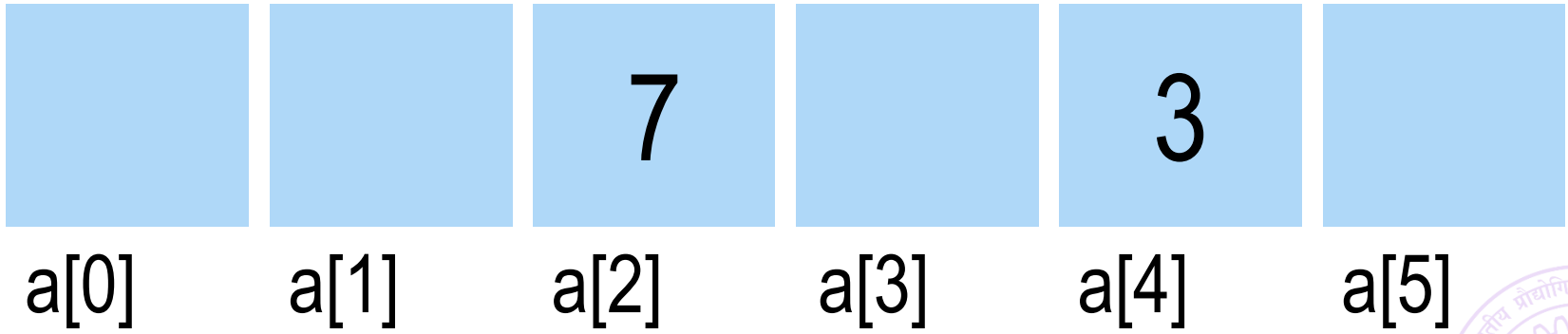
a is like a street on which here are several houses with addresses a[0], a[3] etc

The English word array means "objects in a line" or "ordered series or arrangement" – *The soldiers stand in an array impressed the visiting head of the state on 26 Jan*

For Mr C, an array is a sequence of variables with very convenient names - can have an array of ints, longs, floats, doubles

```
int a[6];
a[2] = 7;
a[4] = 3;
printf("%d", a[2]);
```

a



Arrays

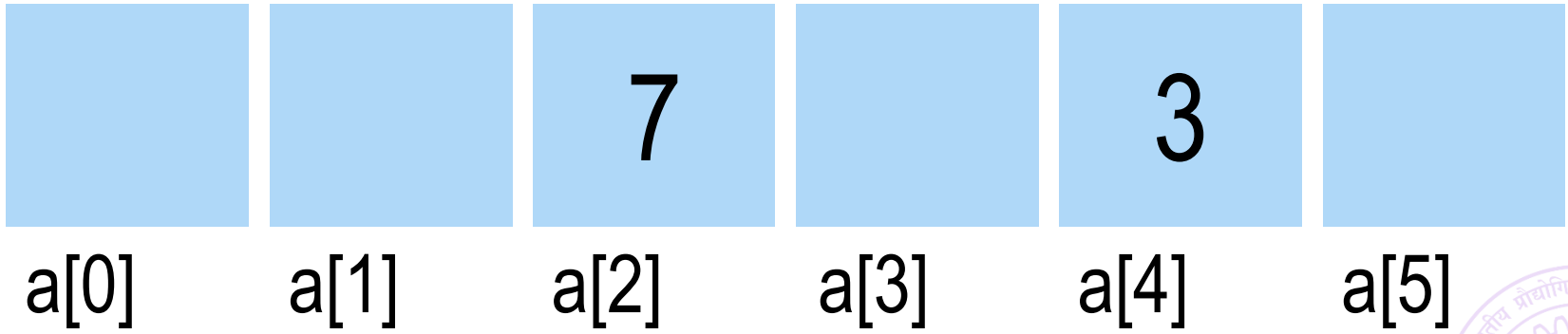
a is like a street on which here are several houses with addresses a[0], a[3] etc

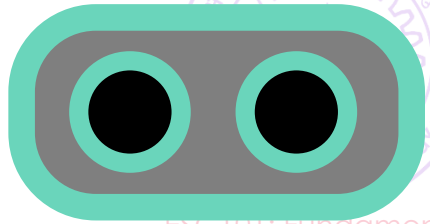
The English word array means "objects in a line" or "ordered series or arrangement" – *The soldiers stan… an array impressed the visiting head of the state on 26 Jan*

For Mr C, an array is a sequence of variables with very convenient names - can have an array of ints, longs, floats, doubles

```
int a[6];
a[2] = 7;
a[4] = 3;
printf("%d", a[2]);
```

a





Arrays

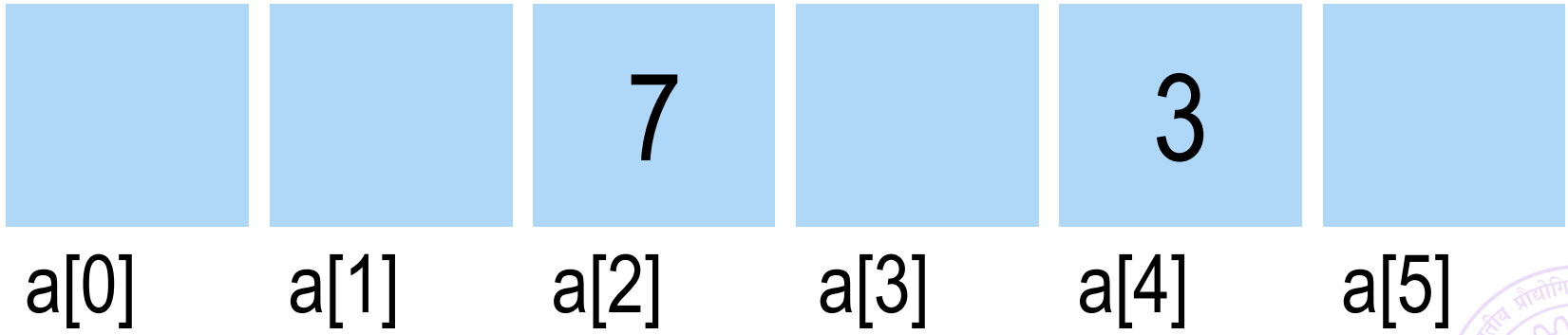
a is like a street on which here are several houses with addresses a[0], a[3] etc

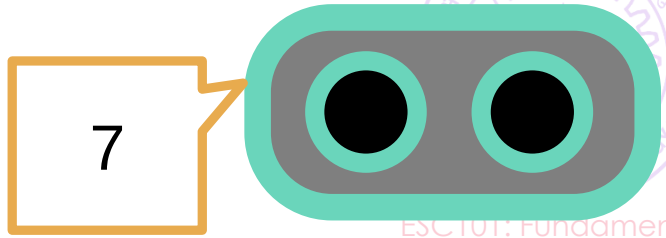
The English word array means "objects in a line" or "ordered series or arrangement" – *The soldiers stan… an array impressed the visiting head of the state on 26 Jan*

For Mr C, an array is a sequence of variables with very convenient names - can have an array of ints, longs, floats, doubles

```
int a[6];
a[2] = 7;
a[4] = 3;
printf("%d", a[2]);
```

a





Arrays

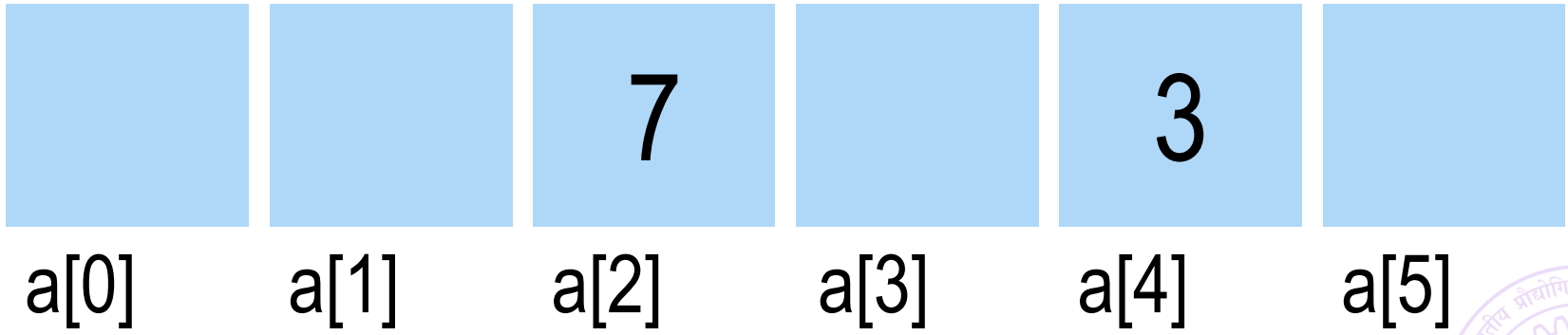
a is like a street on which here are several houses with addresses a[0], a[3] etc

The English word array means "objects in a line" or "ordered series or arrangement" – *The soldiers stan... an array impressed the visiting head of the state on 26 Jan*

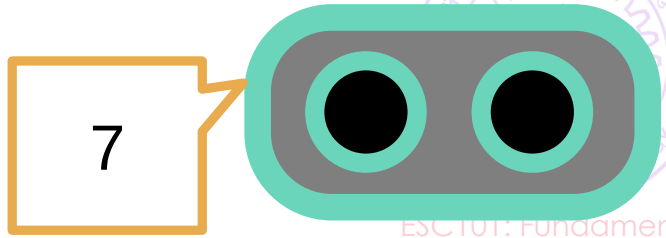
For Mr C, an array is a sequence of variables with very convenient names - can have an array of ints, longs, floats, doubles

```
int a[6];
a[2] = 7;
a[4] = 3;
printf("%d", a[2]);
```

a



a[-1] = 6;



Arrays

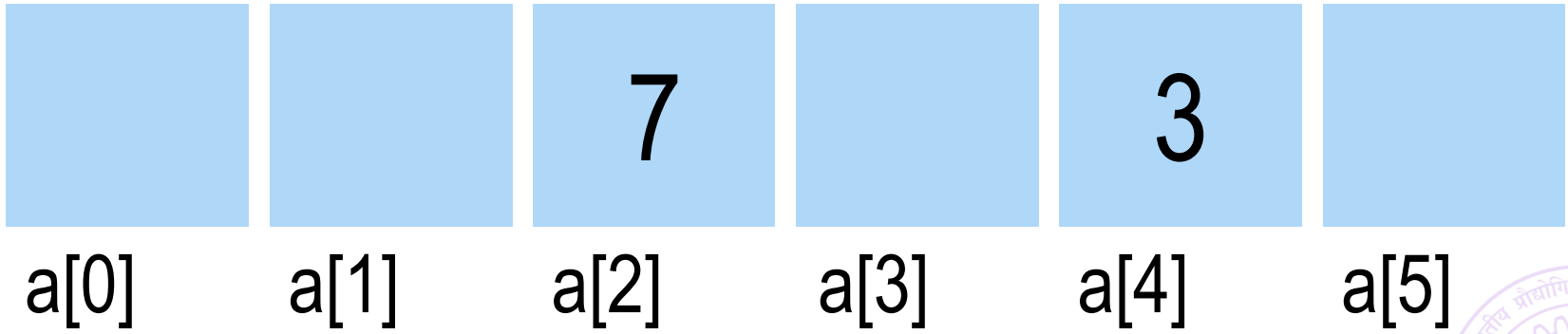
a is like a street on which here are several houses with addresses a[0], a[3] etc

The English word array means "objects in a line" or "ordered series or arrangement" – *The soldiers stan... an array impressed the visiting head of the state on 26 Jan*

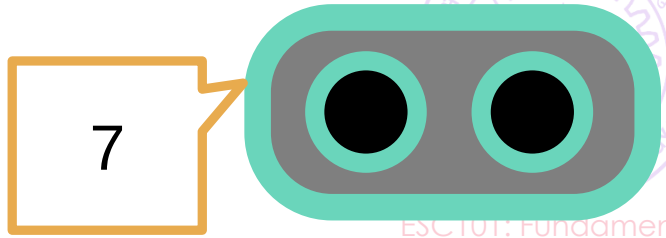
For Mr C, an array is a sequence of variables with very convenient names - can have an array of ints, longs, floats, doubles

```
int a[6];
a[2] = 7;
a[4] = 3;
printf("%d", a[2]);
```

a



a[-1] = 6; a[6] = 4;



Arrays

a is like a street on which here are several houses with addresses a[0], a[3] etc

The English word array means "objects in a line" or "ordered series or arrangement" – *The soldiers stan… an array impressed the visiting head of the state on 26 Jan*

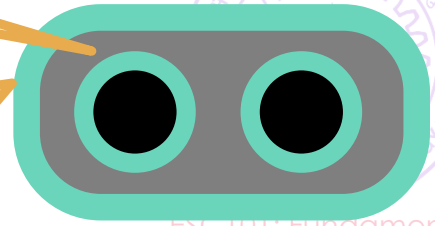
For Mr C, an array is a sequence of variables with very convenient names - can have an array of ints, longs, floats, doubles

```
int a[6];
a[2] = 7;
a[4] = 3;
printf("%d", a[2]);
```

a

a[-1] = 6; a[6] = 4;

Perfect recipe for program crashes



7

Arrays

a is like a street on which here are several houses with addresses a[0], a[3] etc

The English word array means "objects in a line" or "ordered series or arrangement" – *The soldiers stan… an array impressed the visiting head of the state on 26 Jan*

For Mr C, an array is a sequence of variables with very convenient names - can have an array of ints, longs, floats, doubles

```
int a[6];
a[2] = 7;
a[4] = 3;
printf("%d", a[2]);
```

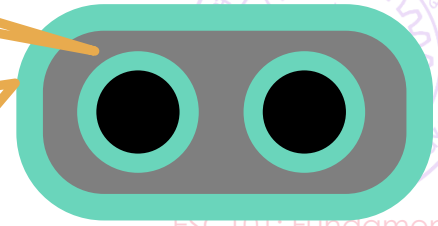
a

Segmentation faults

Perfect recipe for program crashes

```
a[-1] = 6;
a[6] = 4;
```

7



Arrays

a is like a street on which here are several houses with addresses a[0], a[3] etc

The English word array means "objects in a line" or "ordered series or arrangement" – *The soldiers stan... an array impressed the visiting head of the state on 26 Jan*

For Mr C, an array is a sequence of variables with very convenient names - can have an array of ints, longs, floats, doubles

```
int a[6];
a[2] = 7;
a[4] = 3;
printf("%d", a[2]);
```

a

Segmentation faults

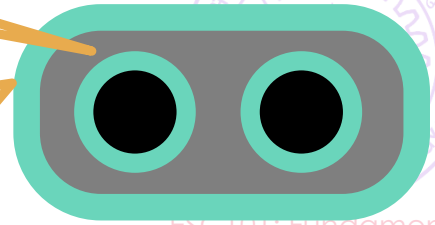
I may not even give a warning when you compile

Perfect recipe for program crashes

a[-1] = 6;

a[6] = 4;

7



Arrays – take care of syntax

Arrays – take care of syntax


```
int a[6];
```

Arrays – take care of syntax


```
int a[6];
```

a[0] to a[5] are just integer variables. Use them as you did any other integer variable – **first variable is a[0] not a[1]**

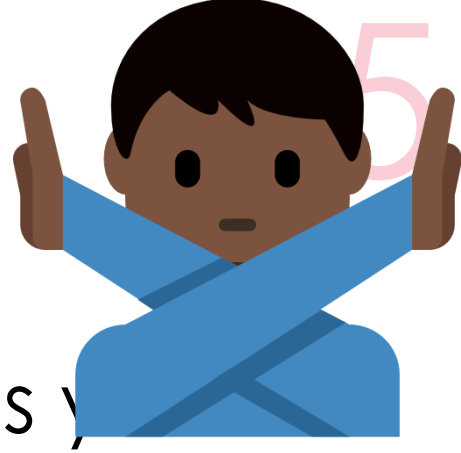
Arrays – take care of syntax


```
int a[6];
```

a[0] to a[5] are just integer variables. Use them as you did any other integer variable – **first variable is a[0] not a[1]**

a = 564; does not make sense – a isn't a single int variable

Arrays – take care of syntax




```
int a[6];
```

a[0] to a[5] are just integer variables. Use them as y
any other integer variable – **first variable is a[0] not a[1]**

a = 564; does not make sense – a isn't a single int variable

Arrays – take care of


```
int a[6];
```

a[0] to a[5] are just integer variables. Use them as y
any other integer variable – **first variable is a[0] not a[1]**

a = 564; does not make sense – a isn't a single int variable

Arrays – take care of


```
int a[6];
```

a[0] to a[5] are just integer variables. Use them as y
any other integer variable – **first variable is a[0] not a[1]**

a = 564; does not make sense – a isn't a single int variable

If you want to give values to whole array

Arrays – take care of

int a[6];

a[0] to a[5] are just integer variables. Use them as y
any other integer variable – **first variable is a[0] not a[1]**

a = 564; does not make sense – a isn't a single int variable

If you want to give values to whole array

Can do it at the time of declaring the array itself

Arrays – take care of


```
int a[6];
```

a[0] to a[5] are just integer variables. Use them as y
any other integer variable – **first variable is a[0] not a[1]**

a = 564; does not make sense – a isn't a single int variable

If you want to give values to whole array

Can do it at the time of declaring the array itself

```
int a[6] = {3,7,6,2,1,0};
```

Arrays – take care of


```
int a[6];
```

a[0] to a[5] are just integer variables. Use them as y
any other integer variable – **first variable is a[0] not a[1]**

a = 564; does not make sense – a isn't a single int variable

If you want to give values to whole array

Can do it at the time of declaring the array itself

```
int a[6] = {3,7,6,2,1,0};
```

Can do it later as well

Arrays – take care of


```
int a[6];
```

a[0] to a[5] are just integer variables. Use them as y
any other integer variable – **first variable is a[0] not a[1]**

a = 564; does not make sense – a isn't a single int variable

If you want to give values to whole array

Can do it at the time of declaring the array itself

```
int a[6] = {3,7,6,2,1,0};
```

Can do it later as well

```
int a[6];
for(i=0;i<6;i++) a[i] = 10;
```

Arrays – take care of


```
int a[6];
```

a[0] to a[5] are just integer variables. Use them as y
any other integer variable – **first variable is a[0] not a[1]**

a = 564; does not make sense – a isn't a single int variable

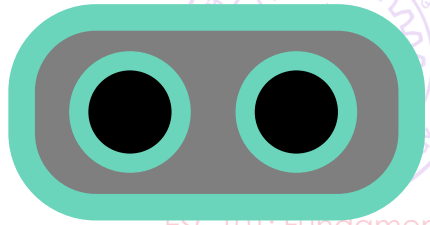
If you want to give values to whole array

Can do it at the time of declaring the array itself

```
int a[6] = {3,7,6,2,1,0};
```

Can do it later as well

```
int a[6];
for(i=0;i<6;i++) a[i] = 10;
```



Arrays – take care of

Cannot send a letter to a street ☺ Can send letter to a house.

```
int a[6];
```

a[0] to a[5] are just integer variables. Use them as y
any other integer variable – **first variable is a[0] not a[1]**

a = 564; does not make sense – a isn't a single int variable

If you want to give values to whole array

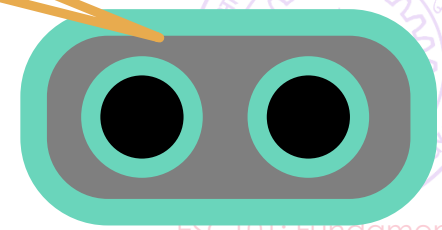
Can do it at the time of declaring the array itself

```
int a[6] = {3,7,6,2,1,0};
```

Can do it later as well

```
int a[6];
for(i=0;i<6;i++) a[i] = 10;
```

a[1.0] is illegal. Array subscript must be integer



Arrays – take care of

Cannot send a letter to a street ☺ Can send letter to a house.


```
int a[6];
```

a[0] to a[5] are just integer variables. Use them as y
any other integer variable – **first variable is a[0] not a[1]**

a = 564; does not make sense – a isn't a single int variable

If you want to give values to whole array

Can do it at the time of declaring the array itself

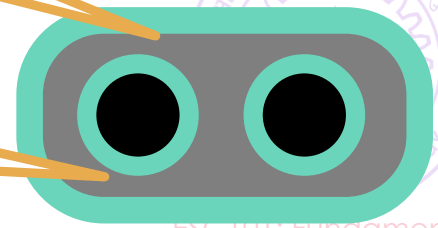
```
int a[6] = {3,7,6,2,1,0};
```

Can do it later as well

```
int a[6];
for(i=0;i<6;i++) a[i] = 10;
```

a[1.0] is illegal. Array subscript must be integer

However, a[2*i+1] where i is integer perfectly fine



Arrays – take care of


```
int a[6];
```

a[0] to a[5] are just integer variables. Use them as y
any other integer variable – **first variable is a[0] not a[1]**

a = 564; does not make sense – a isn't a single int variable

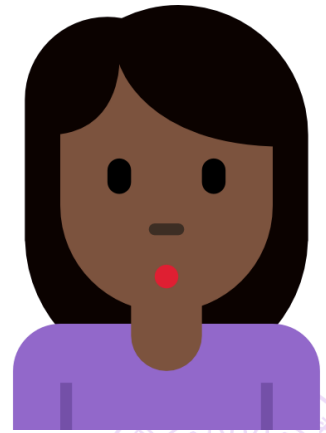
If you want to give values to whole array

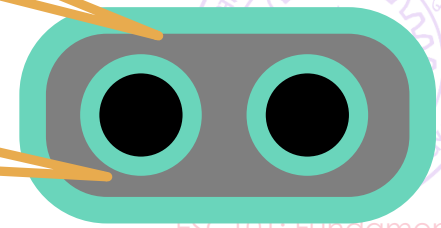
Can do it at the time of declaring the array itself

```
int a[6] = {3,7,6,2,1,0};
```

Can do it later as well

```
int a[6];
for(i=0;i<6;i++) a[i] = 10;
```





Arrays – take care of

Cannot send a letter to a street ☺ Can send letter to a house.

```
int a[6];
```

a[0] to a[5] are just integer variables. Use them as y
any other integer variable – **first variable is a[0] not a[1]**

a = 564; does not make sense – a isn't a single int variable

If you want to give valu

Can do it at the time of decl

```
int a[6] = {3,7,6,2,1,0};
```

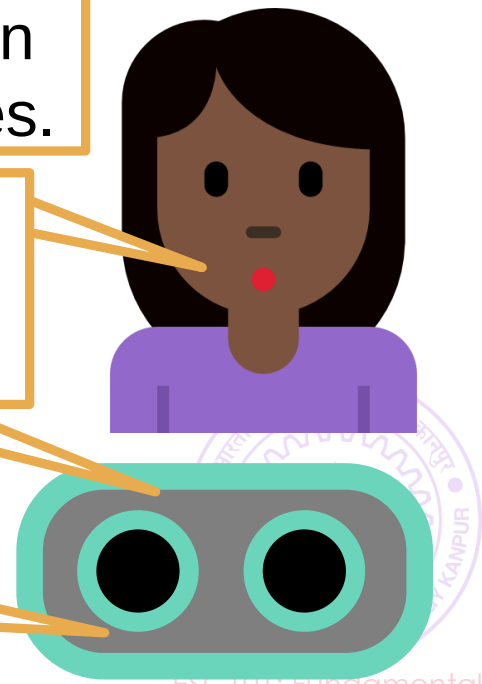
Can do it later as well

```
int a[6];
for(i=0;i<6;i++) a[i] = 10;
```

a[int_expr] is a perfect way to refer to elements of array a if int_expr is an expression that takes integer values.

a[1.0] is illegal. Array subscript must be integer

However, a[2*i+1] where i is integer perfectly fine



BODMAS table has more members

Operator Name	Symbol/Sign	Associativity
Brackets (**array subscript**), Post increment/decrement	(), **[]** ++, --	Left
Unary negation, Pre increment/decrement, NOT	-, ++, --, !	Right
Multiplication/division/ remainder	*, /, %	Left
Addition/subtraction	+, -	Left
Relational	<, <=, >, >=	Left
Relational	==, !=	Left
AND	&&	Left
OR	||	Left
Ternary Conditional	? :	Right
Assignment, Compound assignment	=, +=, -=, *=, /=, %=	Right

Operator Name	Symbol/Sign	Associativity
Brackets (**array subscript**), Post increment/decrement	(), **[]** ++, --	Left
Unary negation, Pre increment/decrement, NOT	-, ++, --, !	Right
Multiplication/division/ remainder	*, /, %	Left
Addition/subtraction	+, -	Left
Relational	<, <=, >, >=	Left
Relational	==, !=	Left
AND	&&	Left
OR	||	Left
Ternary Conditional	? :	Right
Assignment, Compound assignment	=, +=, -=, *=, /=, %=	Right

HIGH PRECEDENCE

LOW PRECEDENCE

Reading Array elements

Reading Array elements

Two ways

Reading Array elements

Two ways

Read into an integer and then transfer into array

Reading Array elements

Two ways

Read into an integer and then transfer into array

```
int a[6], temp;
scanf("%d", &temp);
a[2] = temp;
```

Reading Array elements

Two ways

Read into an integer and then transfer into array

```
int a[6], temp;
scanf("%d", &temp);
a[2] = temp;
```

Directly read into the array element

Reading Array elements

Two ways

Read into an integer and then transfer into array

```
int a[6], temp;
scanf("%d", &temp);
a[2] = temp;
```

Directly read into the array element

```
int a[6];
scanf("%d", &a[2]);
```

Reading Array elements

Two ways

Read into an integer and then transfer into array

```
int a[6], temp;
scanf("%d", &temp);
a[2] = temp;
```

Directly read into the array element

```
int a[6];
scanf("%d", &a[2]);
```

Precedence rules force &a[2] to be interpreted by Mr C as &(a[2]) and not as (&a)[2]