Mr C has Character

ESC101: Fundamentals of Computing

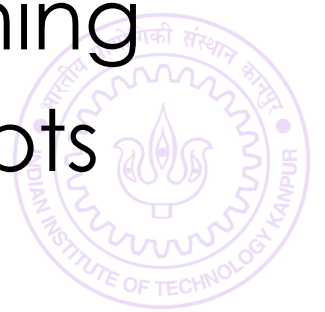
Purushottam Kar

Mid-sem Theory Exam

- September 22nd, 2018 (Saturday)
- Time: 1PM – 3PM
- Room: announced shortly
- Syllabus: till whatever is covered till Sep 14th tutorial
- No Make-up Exam – do not miss this exam
- Open handwritten notes – no printouts, mobiles, iPads.

Doubt-clearing Session

- Date: September 15th, 2018 (coming Saturday)
- Time: 5PM – 7PM
- Room: CC-02
- Students not comfortable with English are welcome
- Other students also welcome to clear doubts
- Please revise and have list of doubts before coming
- Will not cover lectures again in detail – only doubts



Advanced Track

- 29 students selected for advanced track
- If you have accepted advanced track, no need to come to lab today onwards
- Will receive mail this evening with mentor allocation
- Please contact mentors immediately and decide on project ideas
- Must have a preliminary project idea by end of this week (Friday evening September 14th, 2018)

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*

In C, an array is a sequence of variables
can have 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*

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

```
int a[6],b=5;
```

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*

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

```
int a[6],b=5;
```

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*

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

```
int a[6],b=5;
```

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*

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

```
int a[6],b=5;
```

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*

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

```
int a[6],b=5;
```

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*

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

```
int a[6],b=5;
```

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*

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

```
int a[6],b=5;
```

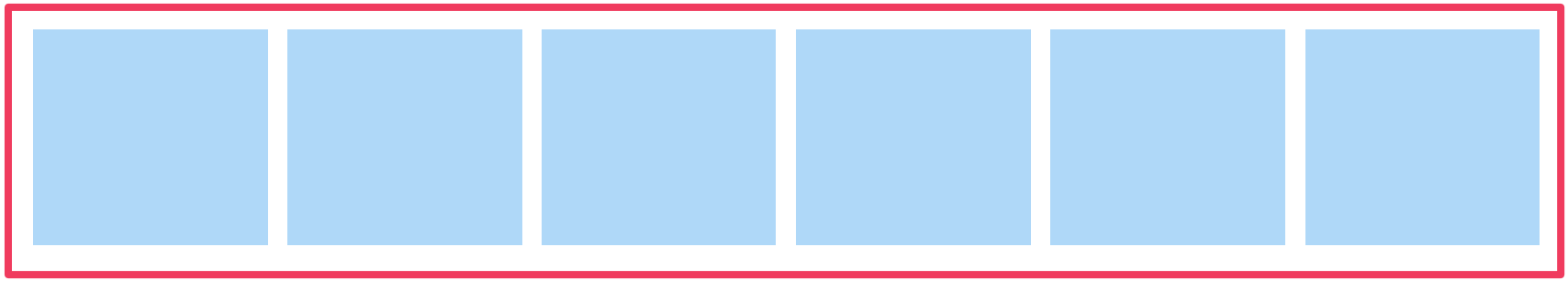
Arrays

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*

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

```
int a[6],b=5;
```

a

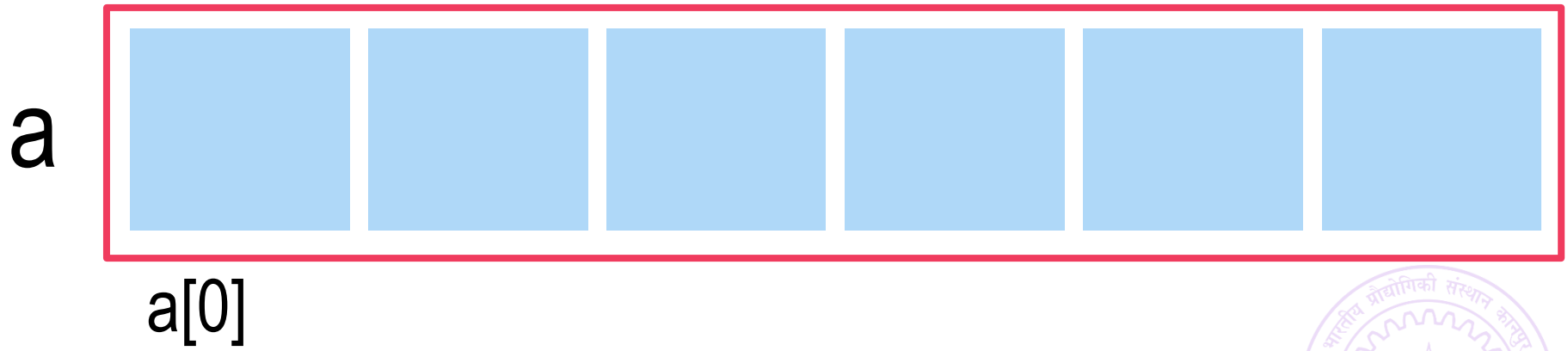


Arrays

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*

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

```
int a[6],b=5;
```

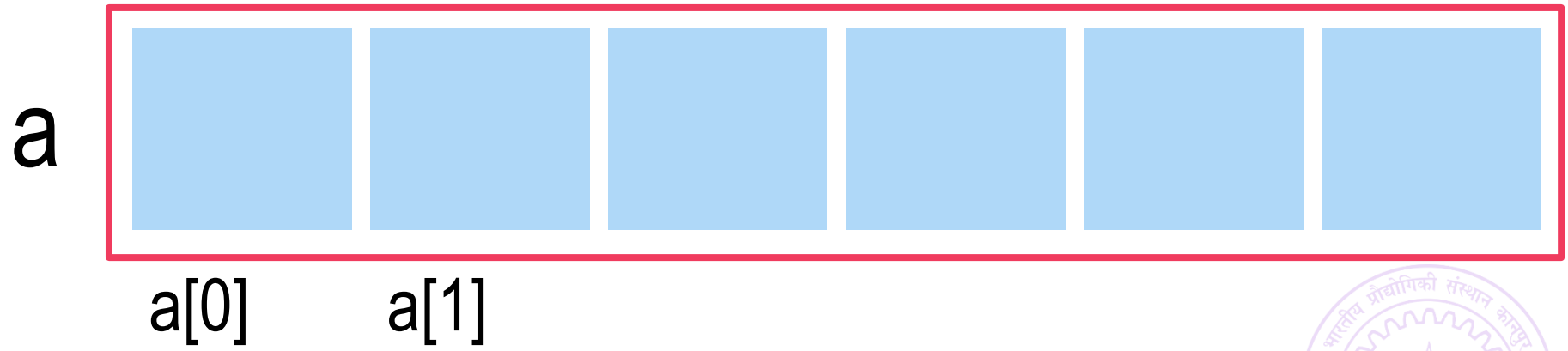


Arrays

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*

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

```
int a[6],b=5;
```

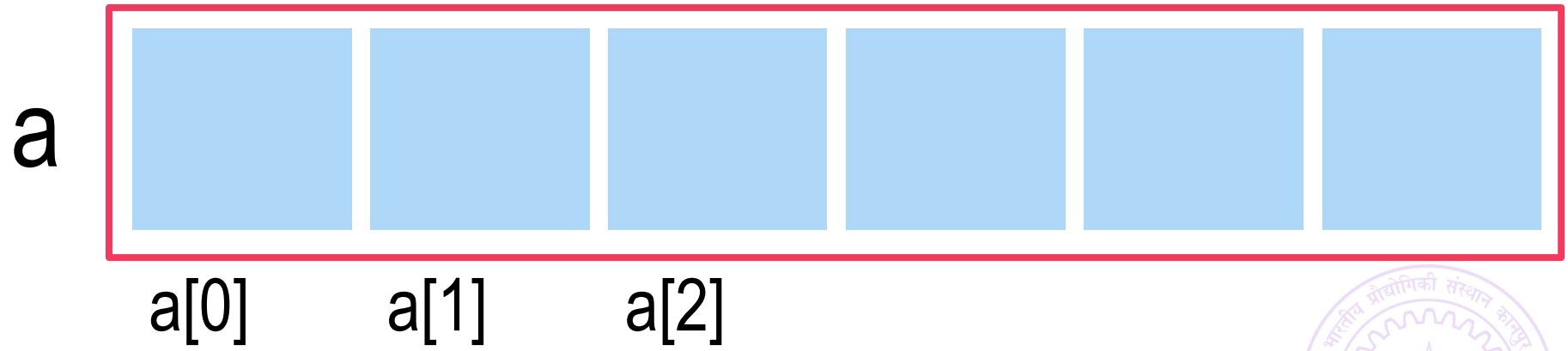


Arrays

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*

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

```
int a[6],b=5;
```

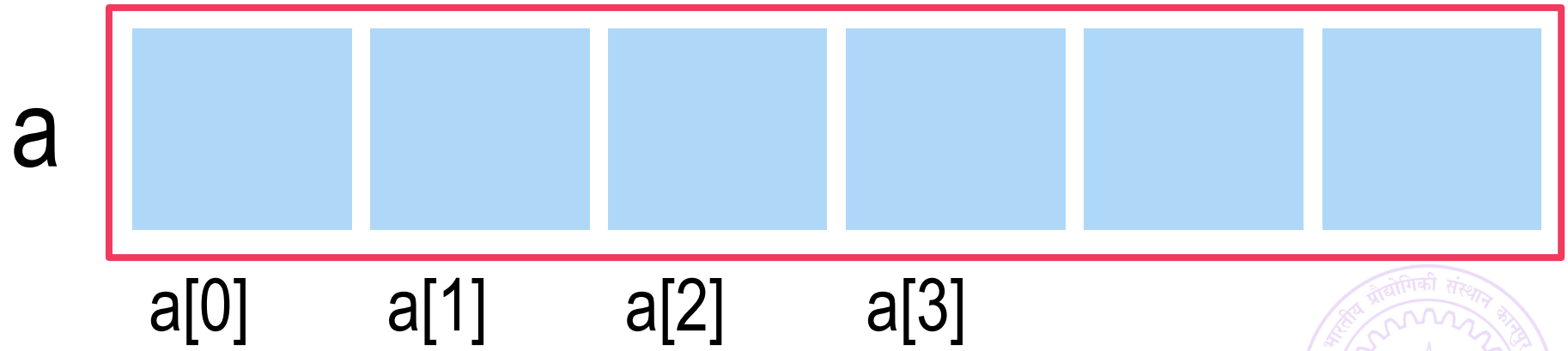


Arrays

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*

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

```
int a[6],b=5;
```

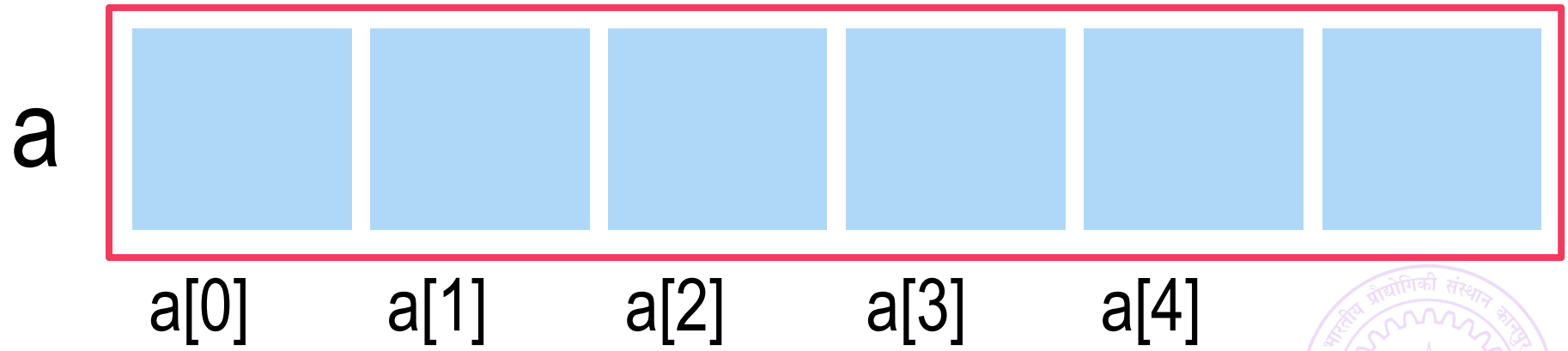


Arrays

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*

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

```
int a[6],b=5;
```

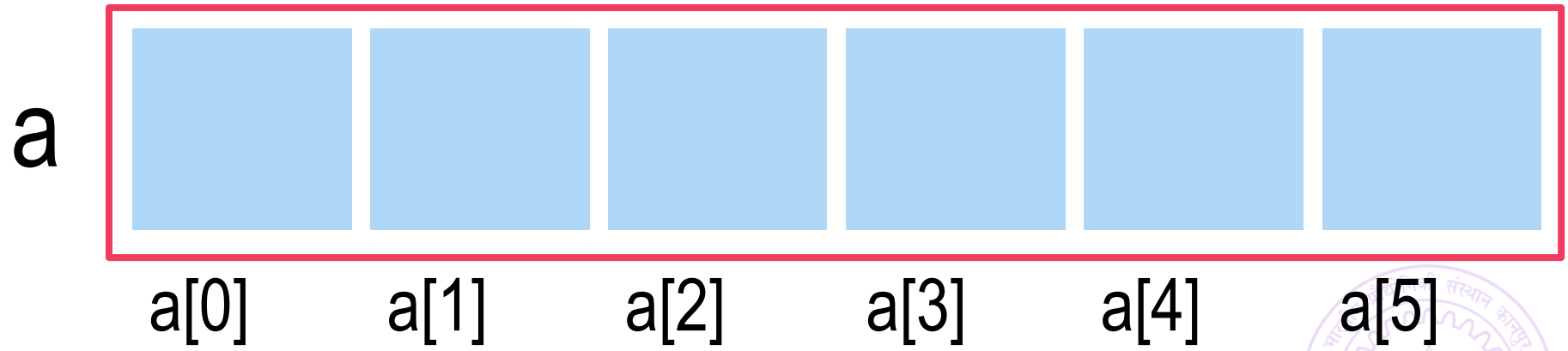


Arrays

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*

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

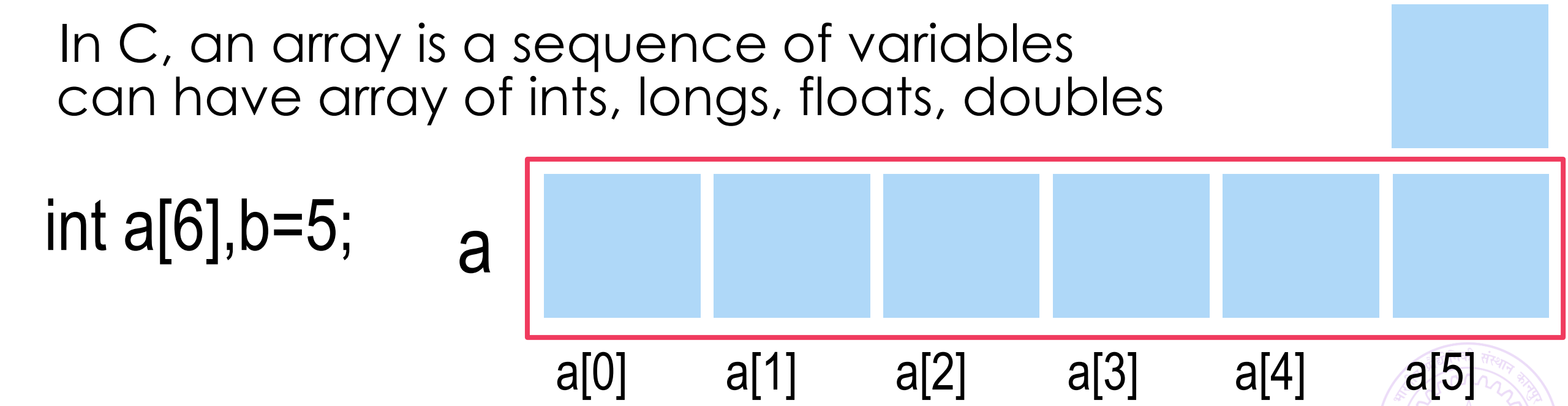
```
int a[6],b=5;
```



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*

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

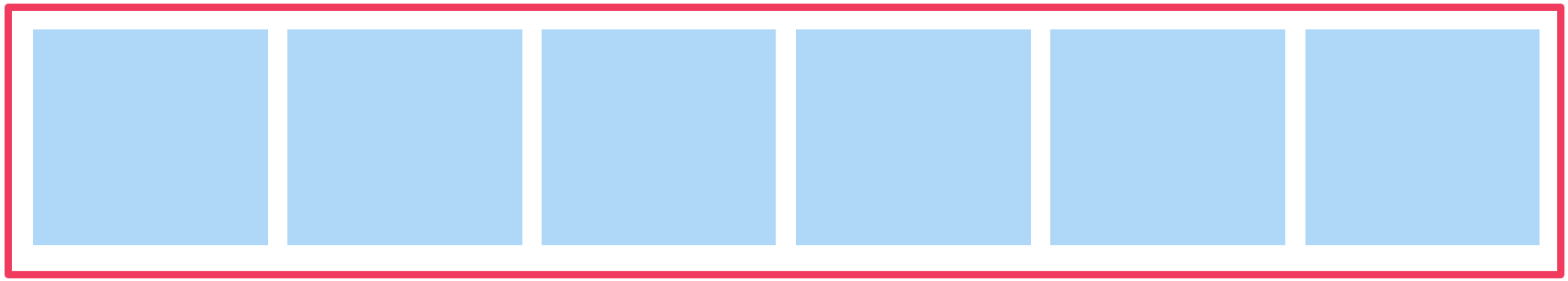


Arrays

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*

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

b



int a[6],b=5;

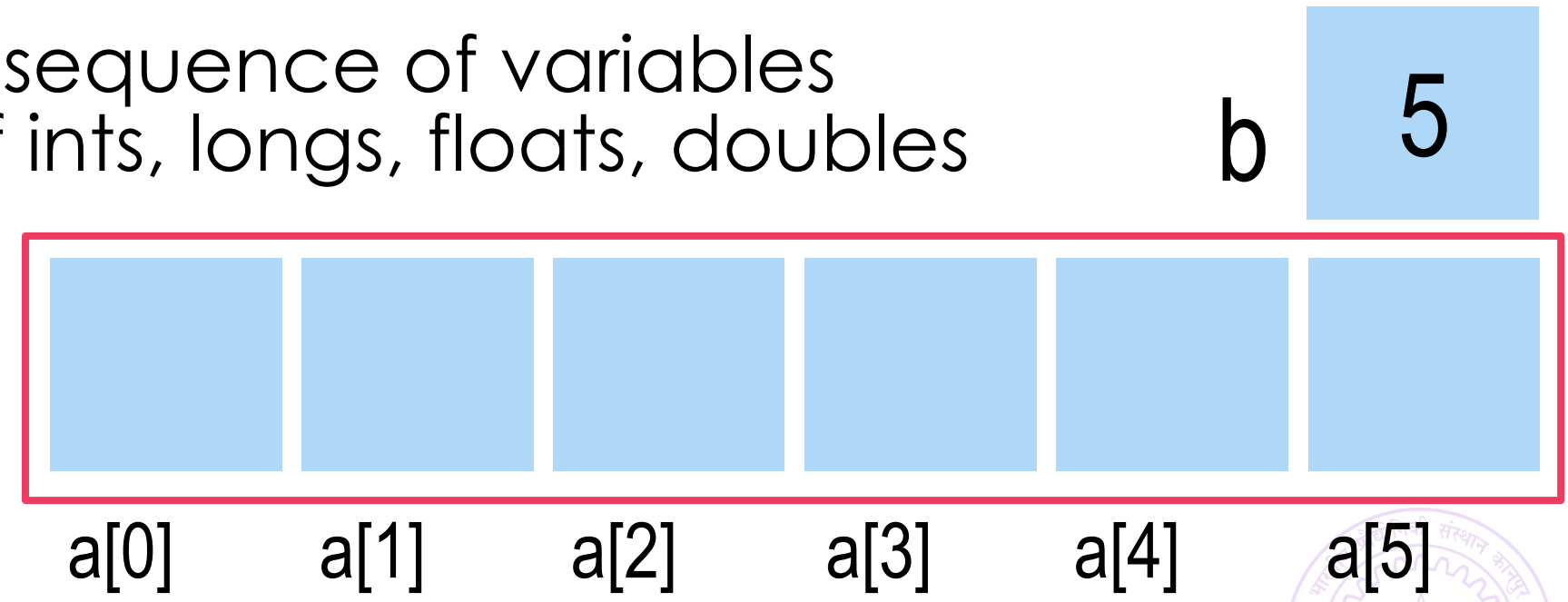
a

a[0] a[1] a[2] a[3] a[4] a[5]

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*

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles



Arrays

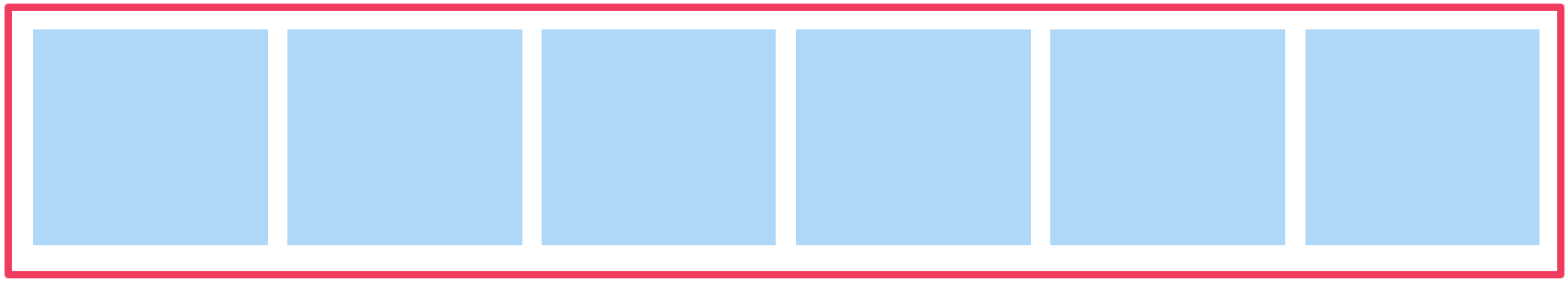
The English word array means "objects in a line" or "objects in an ordered series or arrangement" – *The soldiers stan… an array impressed the visiting head of the state on 26 Jan*

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

b 5

int a[6],b=5;

a



Arrays

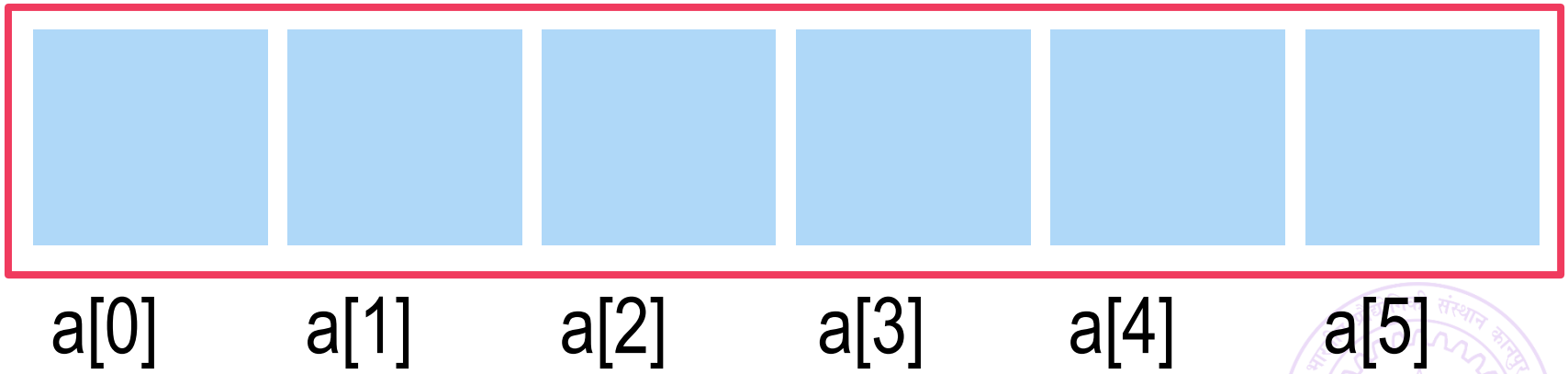
a is like a street on which here are several houses with addresses a[0], a[3] etc

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

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

b 5

int a[6],b=5;



Arrays

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

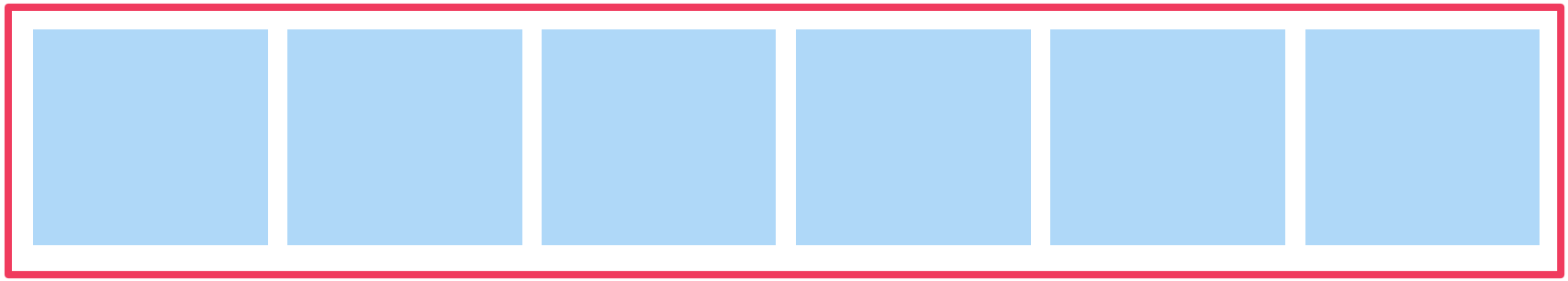
The English word array means "objects in a line" or "objects in an ordered series or arrangement" – *The soldiers stan… an array impressed the visiting head of the state on 26 Jan*

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

b 5

```
int a[6],b=5;
a[b-4] = 4;
```

a



Arrays

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

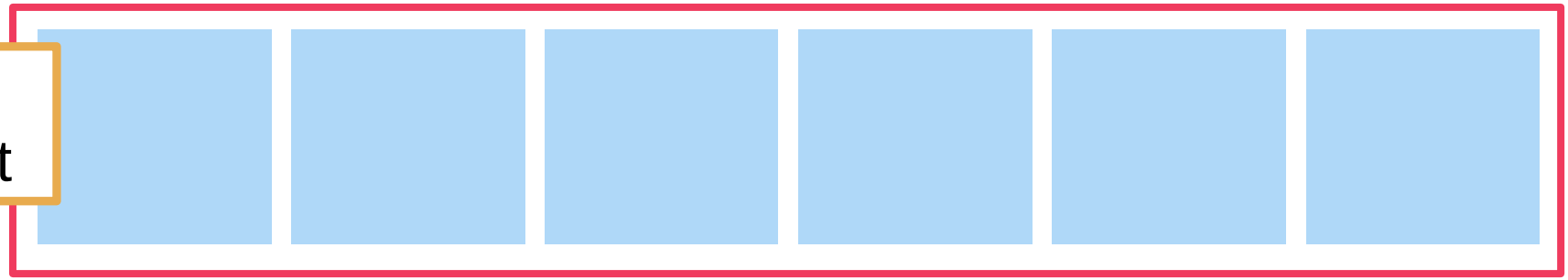
The English word array means "objects in a line" ... ordered series or arrangement" – *The soldiers stan... an array impressed the visiting head of the state on 26 Jan*

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

b 5

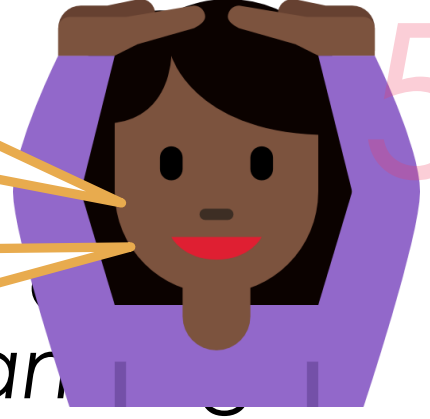
```
int a[6],b=5
a[b-4] = 4;
```

Array subscript



Arrays

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



Can use **integer** expressions as array subscripts **not float/double**

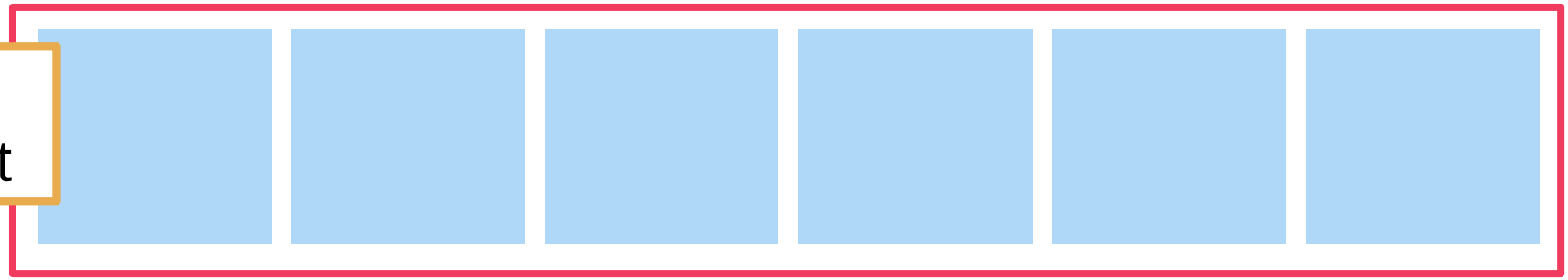
The English word array
ordered series or arra
an array impressed the visiting head of the state on 26 Jan

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

b 5

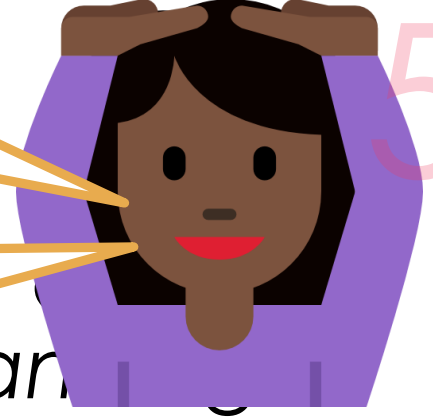
```
int a[6],b=5
a[b-4] = 4;
```

Array subscript



Arrays

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



Can use **integer** expressions as array subscripts **not float/double**

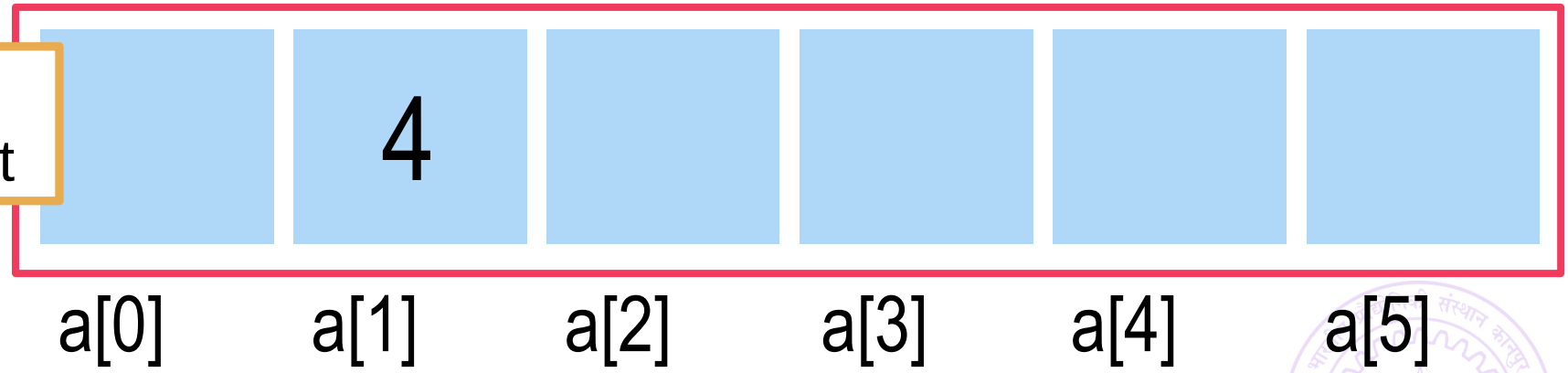
The English word array
ordered series or arra
an array impressed the visiting head of the state on 26 Jan

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

b 5

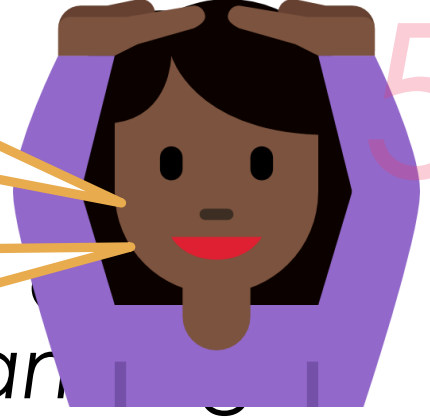
```
int a[6],b=5
a[b-4] = 4;
```

Array subscript



Arrays

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



Can use **integer** expressions as array subscripts **not float/double**

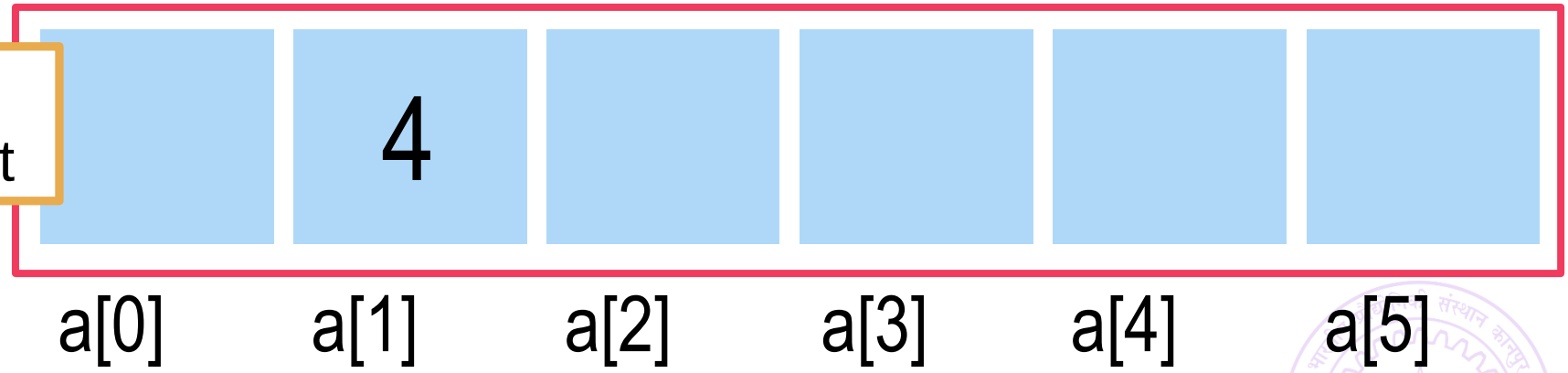
The English word array
ordered series or arra
an array impressed the visiting head of the state on 26 Jan

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

b 5

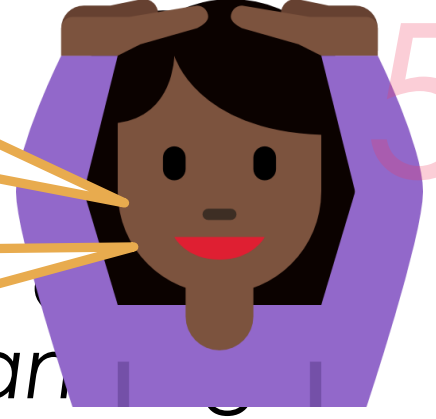
```
int a[6],b=5;
a[b-4] = 4;
a[4] = 3;
```

Array subscript



Arrays

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



Can use **integer** expressions as array subscripts **not float/double**

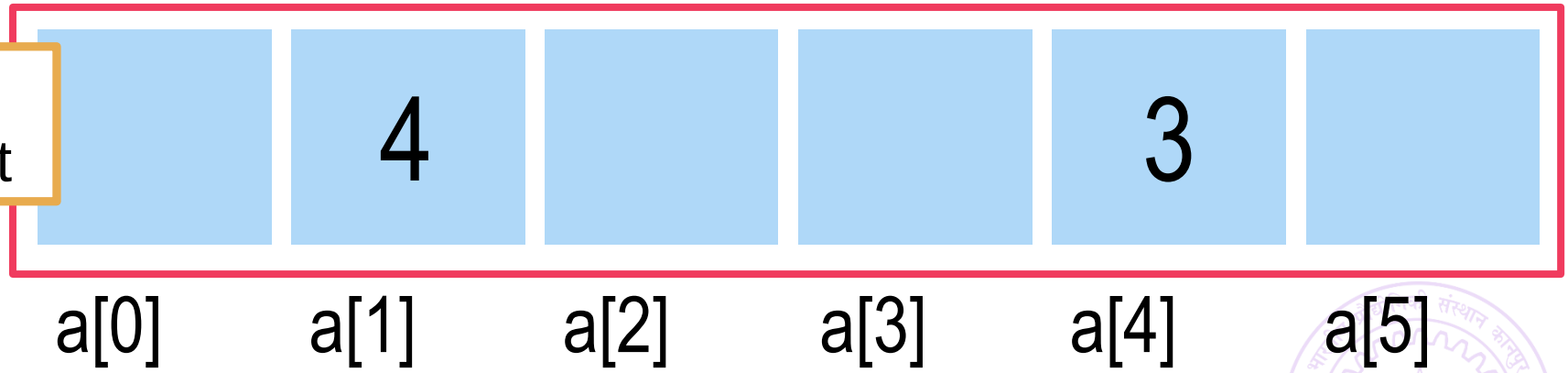
The English word array
ordered series or arra
an array impressed the visiting head of the state on 26 Jan

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

b 5

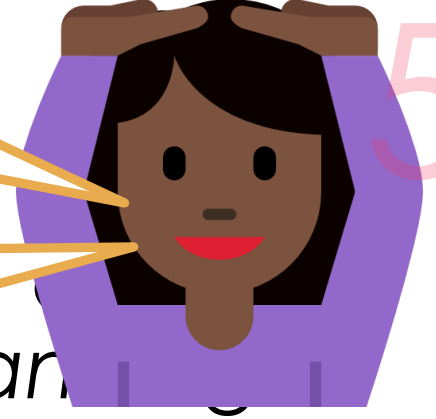
```
int a[6],b=5;
a[b-4] = 4;
a[4] = 3;
```

Array subscript



Arrays

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



Can use **integer** expressions as array subscripts **not float/double**

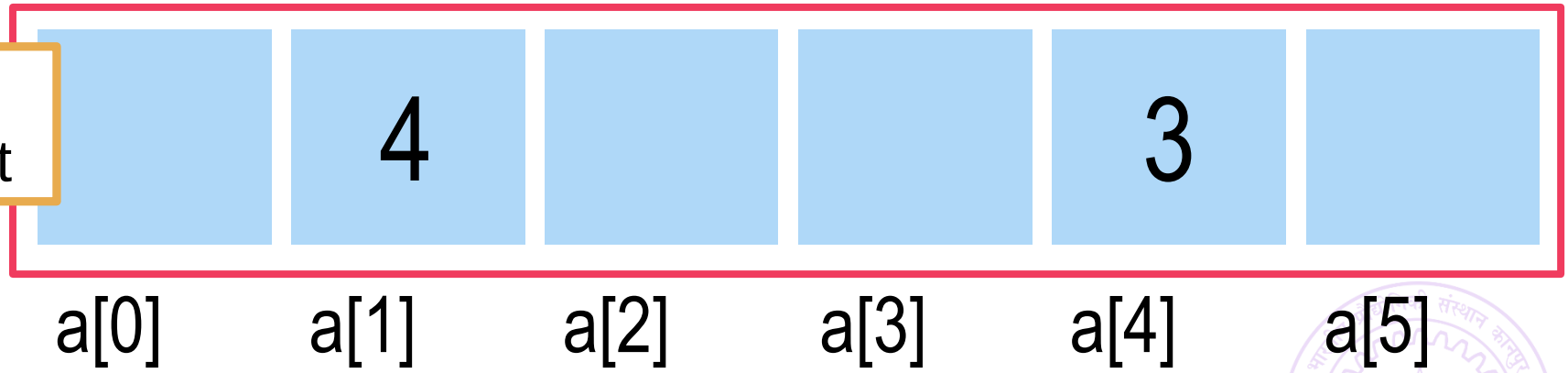
The English word array
ordered series or arra
an array impressed the visiting head of the state on 26 Jan

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

b 5

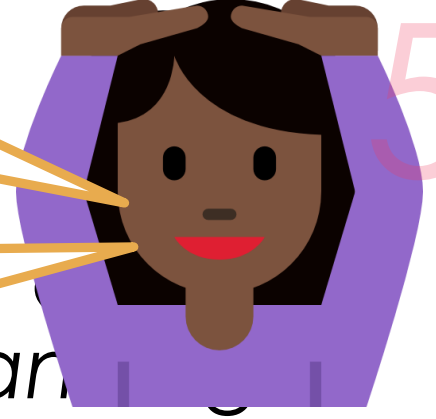
```
int a[6],b=5;
a[b-4] = 4;
a[4] = 3;
printf("%d", a[a[1]]);
```

Array subscript



Arrays

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



Can use **integer** expressions as array subscripts **not float/double**

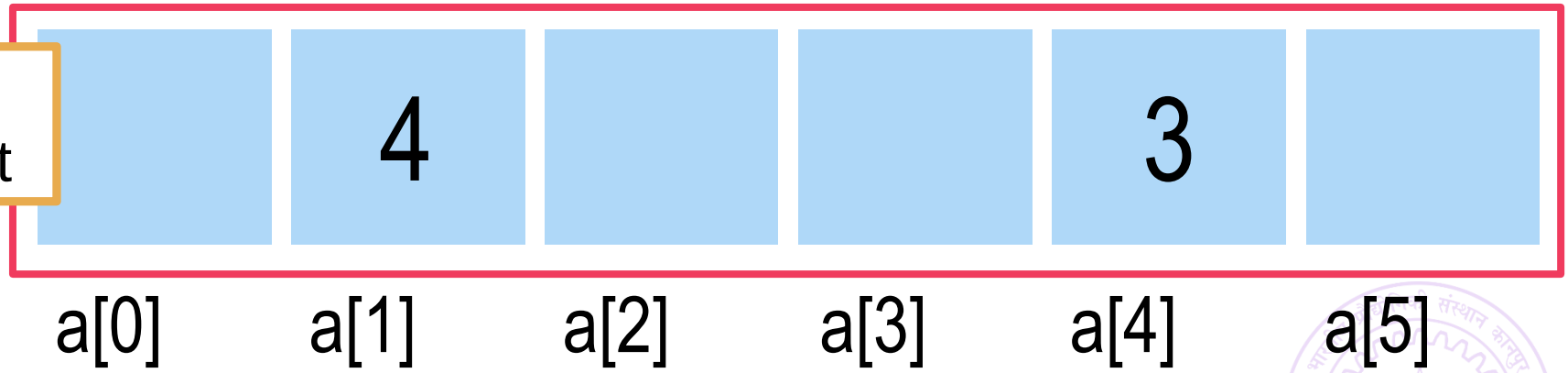
The English word array
ordered series or arra
an array impressed the visiting head of the state on 26 Jan

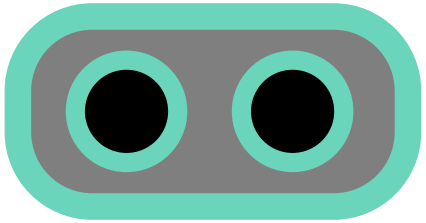
In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

b 5

```
int a[6],b=5
a[b-4] = 4;
a[4] = 3;
printf("%d", a[a[1]]);
```

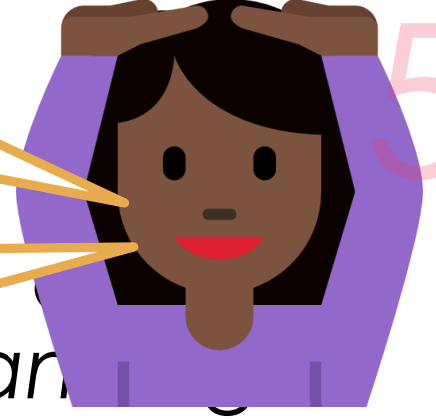
Array subscript





Arrays

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



Can use **integer** expressions as array subscripts **not float/double**

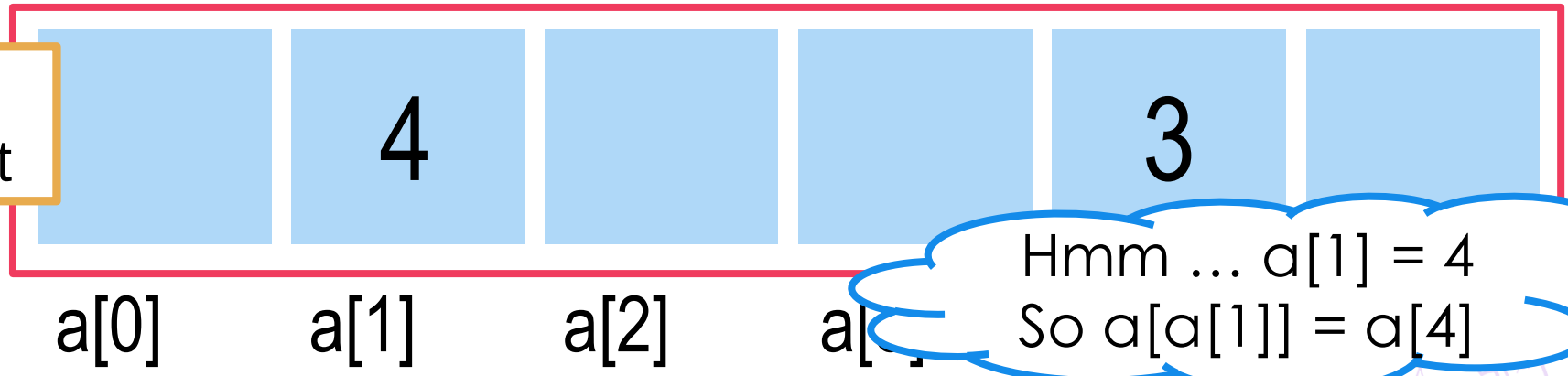
The English word array
ordered series or arra
an array impressed the visiting head of the state on 26 Jan

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

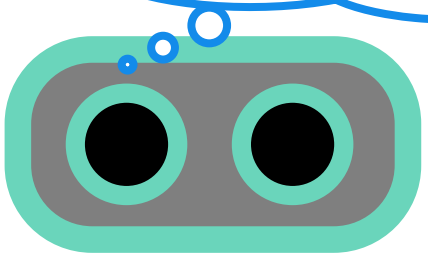
b 5

```
int a[6],b=5
a[b-4] = 4;
a[4] = 3;
printf("%d", a[a[1]]);
```

Array subscript

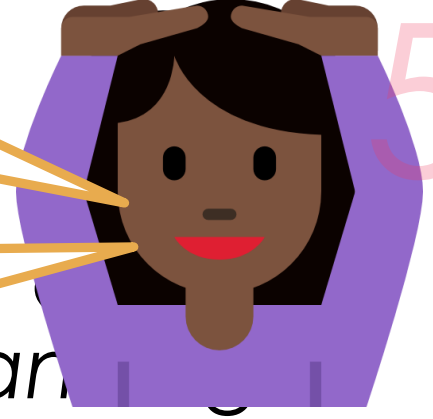


Hmm … a[1] = 4
So a[a[1]] = a[4]



Arrays

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



Can use **integer** expressions as array subscripts **not float/double**

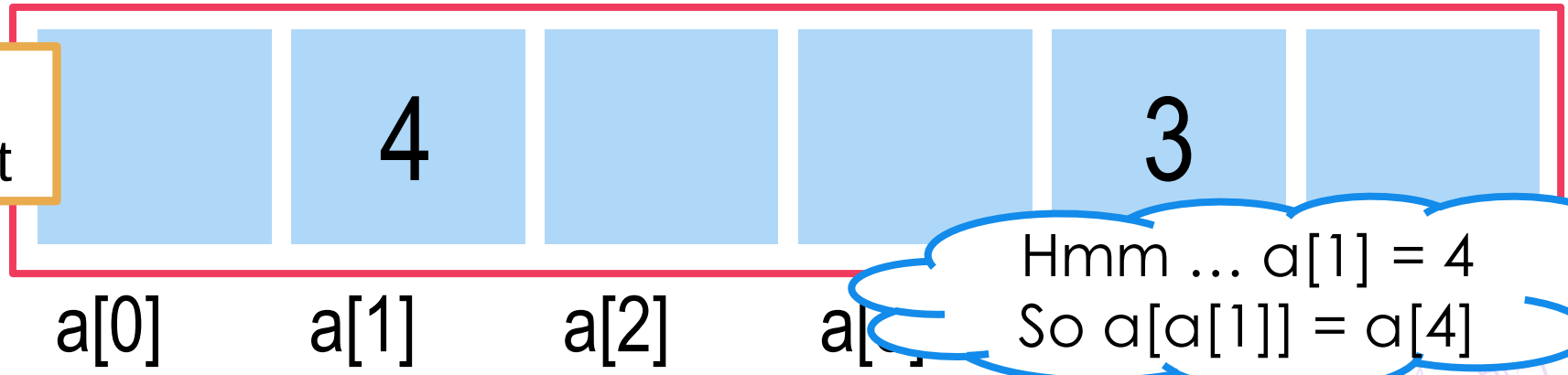
The English word array
ordered series or arra
an array impressed the visiting head of the state on 26 Jan

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

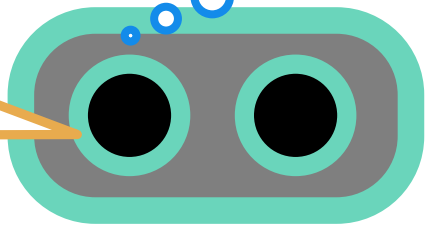
b 5

```
int a[6],b=5
a[b-4] = 4;
a[4] = 3;
printf("%d", a[a[1]]);
```

Array subscript



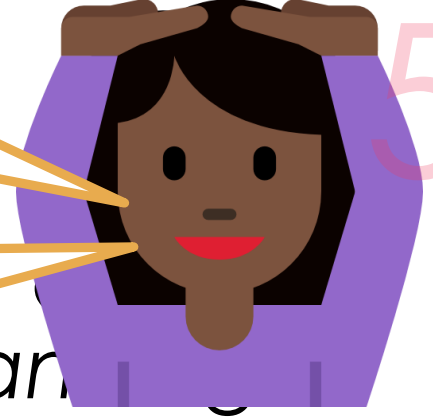
Hmm … a[1] = 4
So a[a[1]] = a[4]



3

Arrays

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



Can use **integer** expressions as array subscripts **not float/double**

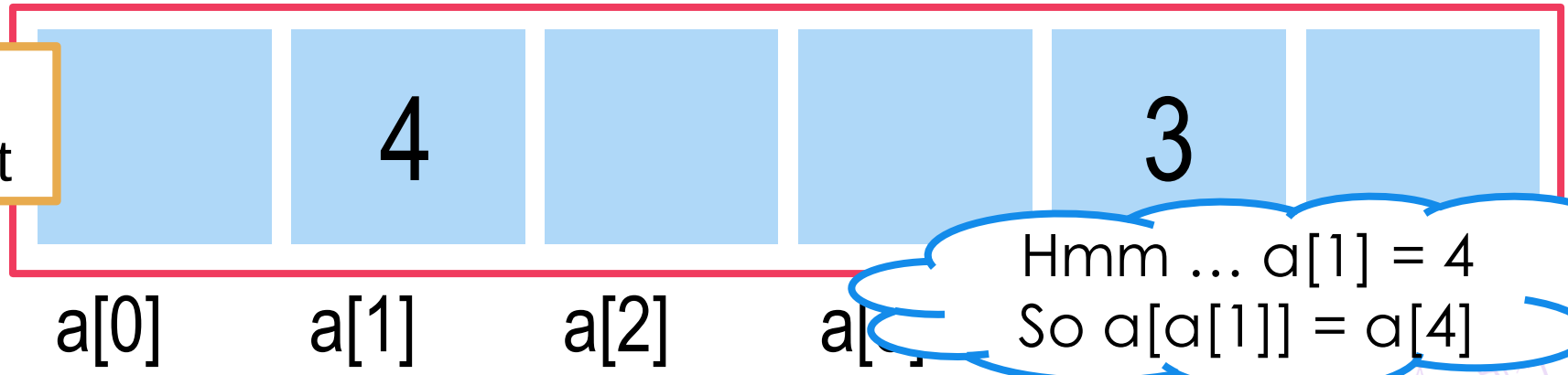
The English word array
ordered series or arra
an array impressed the visiting head of the state on 26 Jan

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

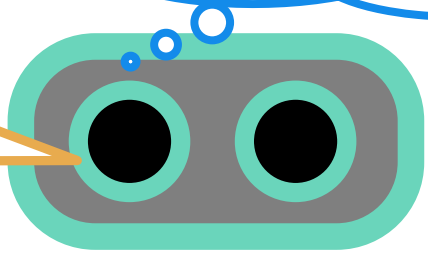
b 5

```
int a[6],b=5
a[b-4] = 4;
a[4] = 3;  a[-1] = 6;
printf("%d", a[a[1]]);
```

Array subscript



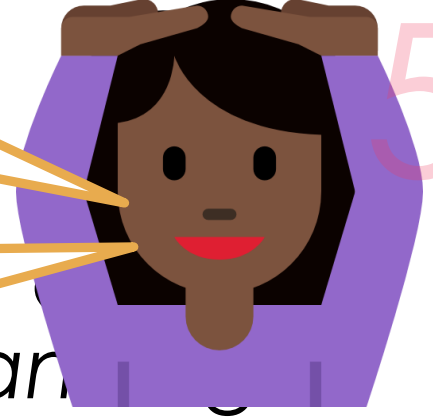
Hmm … a[1] = 4
So a[a[1]] = a[4]



3

Arrays

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



Can use **integer** expressions as array subscripts **not float/double**

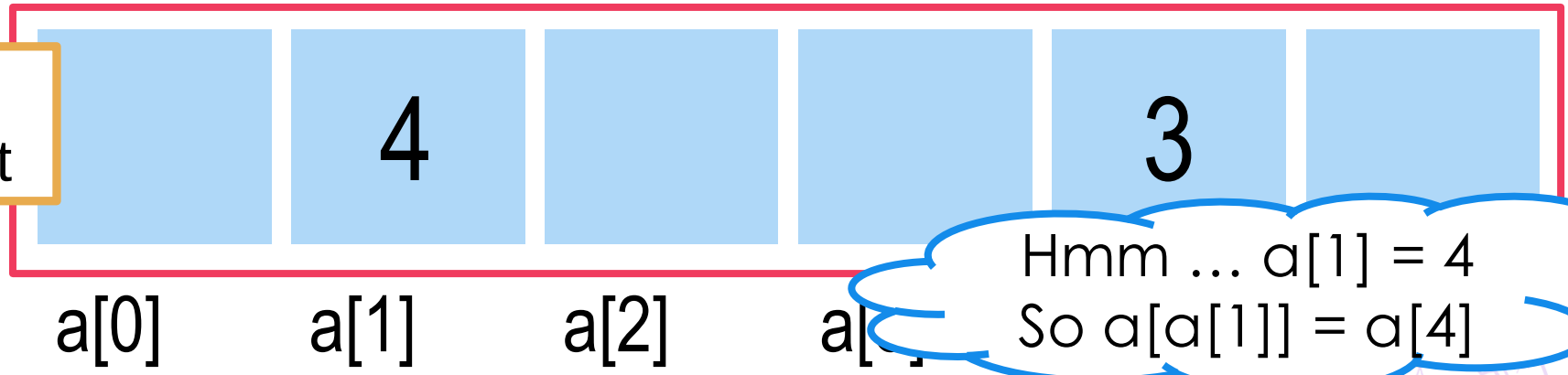
The English word array
ordered series or arra
an array impressed the visiting head of the state on 26 Jan

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

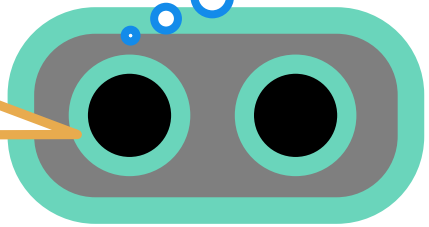
b 5

```
int a[6],b=5
a[b-4] = 4;
a[4] = 3;  a[-1] = 6;
printf("%d", a[a[1]]);  a[6] = 4;
```

Array subscript



Hmm … a[1] = 4
So a[a[1]] = a[4]



3

Arrays

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

Can use **integer** expressions as array subscripts **not float/double**

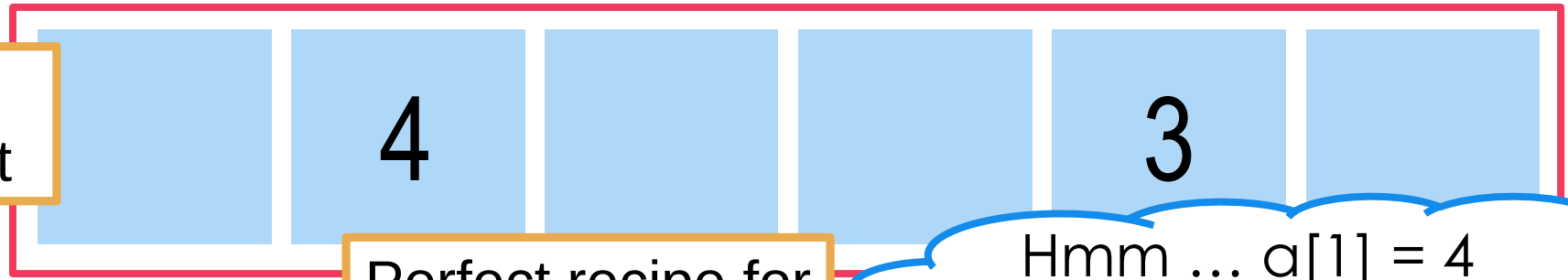
The English word array ... ordered series or arran... *an array impressed the visiting head of the state on 26 Jan*

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

b 5

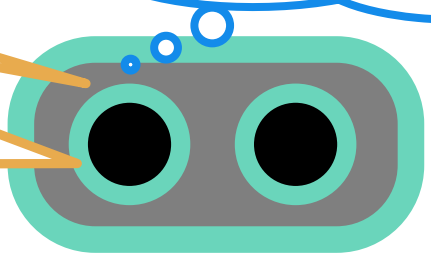
```
int a[6],b=5
a[b-4] = 4;
a[4] = 3;  a[-1] = 6;
printf("%d", a[a[1]]);  a[6] = 4;
```

Array subscript



Perfect recipe for program crashes

Hmm … a[1] = 4
So a[a[1]] = a[4]



3

Arrays

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

Can use **integer** expressions as array subscripts **not float/double**

The English word array
ordered series or arran
an array impressed the visiting head of the state on 26 Jan

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

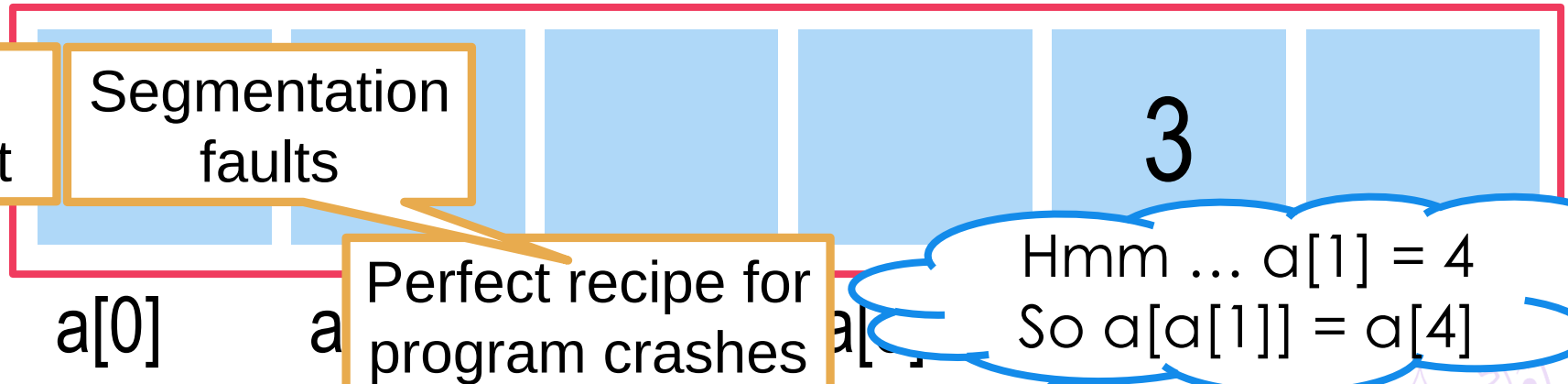
b 5

```
int a[6],b=5
a[b-4] = 4;
a[4] = 3;  a[-1] = 6;
printf("%d", a[a[1]]);  a[6] = 4;
```

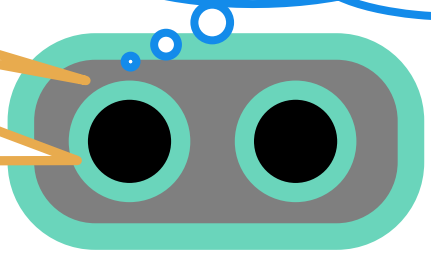
Array subscript

Segmentation faults

Perfect recipe for program crashes



Hmm … a[1] = 4
So a[a[1]] = a[4]



3

Arrays

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

Can use **integer** expressions as array subscripts **not float/double**

The English word array
ordered series or arran
an array impressed the visiting head of the state on 26 Jan

In C, an array is a sequence of variables
can have array of ints, longs, floats, doubles

b 5

```
int a[6],b=5
a[b-4] = 4;
a[4] = 3;  a[-1] = 6;
printf("%d", a[a[1]]);  a[6] = 4;
```

Array subscript

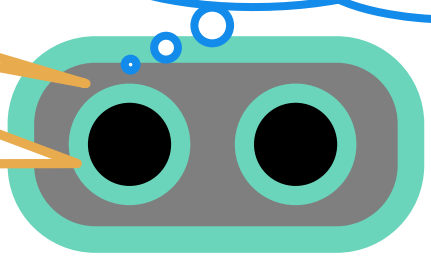
Segmentation faults

I may not even give a warning when you compile

Perfect recipe for program crashes

a[0]

Hmm … a[1] = 4
So a[a[1]] = a[4]



3

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]**

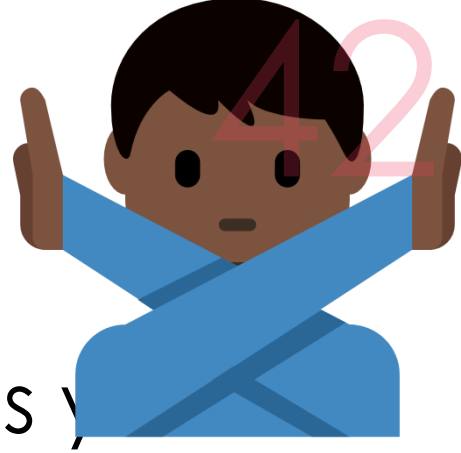
Arrays – take care of syntax


```
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

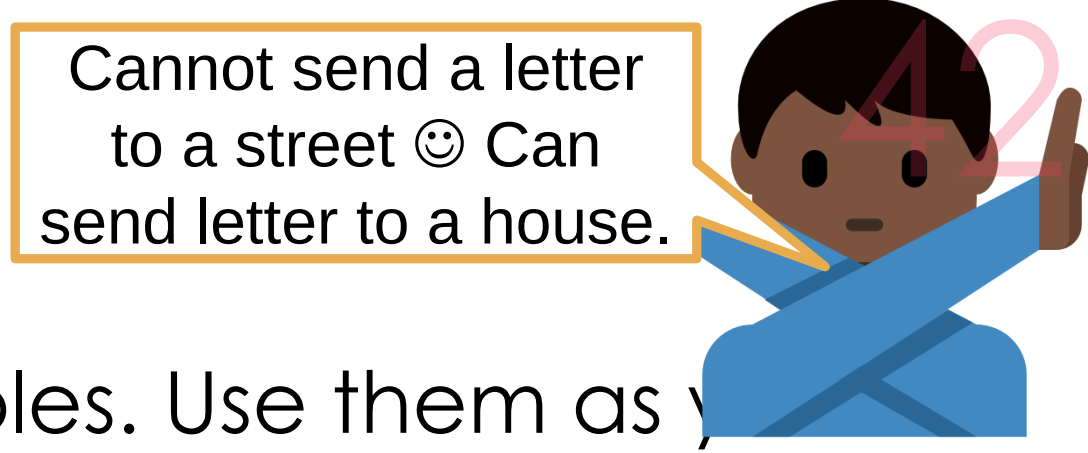
Arrays – take care of


```
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

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

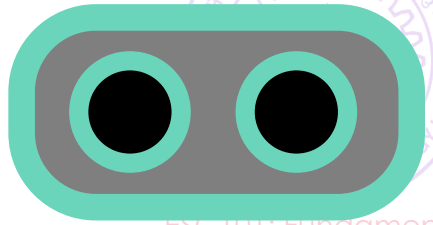
If you want to give values to whole array

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;
```



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

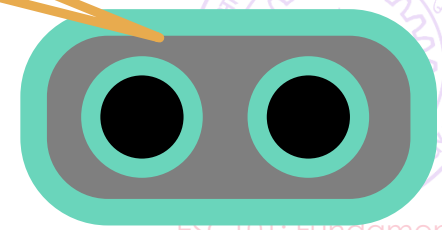
Can do it at the time of declaring the array itself

```
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

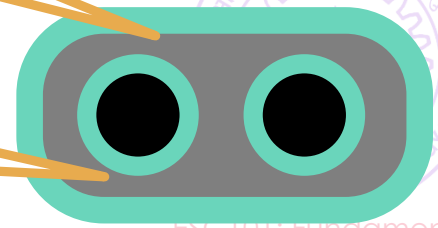
```
int a[6] = {3,7,6,2,1,0};
```

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

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
```

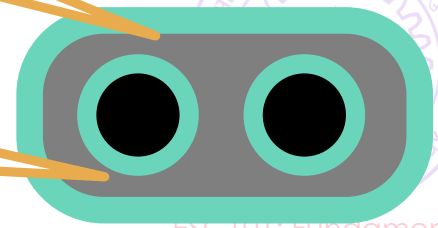
a[(int)1.0] is okay too since it has been typecast to int

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

Can do it later as well

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

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



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

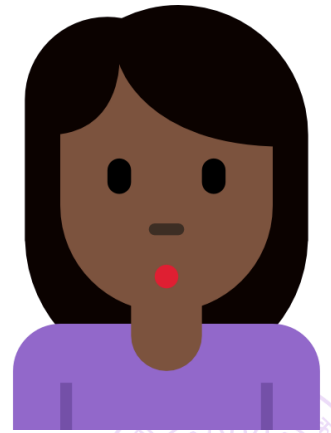
If you want to give values to whole array

Can do it at the time of declaring the array itself

```
int a[6] = {3,7
```

a[(int)1.0] is okay too since it has been typecast to int

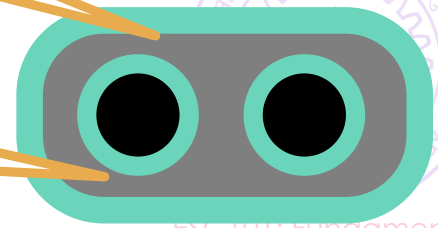
a[1.0] is illegal. Array subscript must be integer



Can do it later as well

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

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



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 giv

Wow … so a[int_expr] is a perfect way to refer to elements of array a whenever int_expr is an expression that takes int values.

Can do it at the time

```
int a[6] = {3,7
```

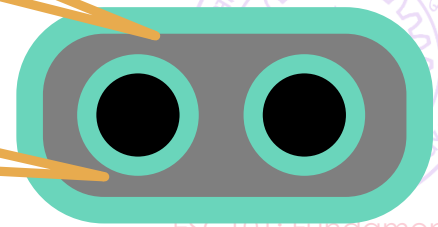
a[(int)1.0] is okay too since it has been typecast to int

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

Can do it later as well

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

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



A Few Sample Problems

A Few Sample Problems

Given a 10 dimensional array in input, find the sum of first, third, fifth … i.e. odd location elements of the array and sum of even location elements in the array

A Few Sample Problems

Given a 10 dimensional array in input, find the sum of first, third, fifth … i.e. odd location elements of the array and sum of even location elements in the array

Given two lists of integers on two separate lines, find the smallest integer in the first list that is not present in the second list.

A Few Sample Problems

Given a 10 dimensional array in input, find the sum of first, third, fifth … i.e. odd location elements of the array and sum of even location elements in the array

Given two lists of integers on two separate lines, find the smallest integer in the first list that is not present in the second list.

In second problem, have to revisit list elements – go back and forth – arrays allow conveniently storing the full list

A Few Sample Problems

Given a 10 dimensional array in input, find the sum of first, third, fifth … i.e. odd location elements of the array and sum of even location elements in the array

Given two lists of integers on two separate lines, find the smallest integer in the first list that is not present in the second list.

In second problem, have to revisit list elements – go back and forth – arrays allow conveniently storing the full list

In first problem, need not store the entire list first

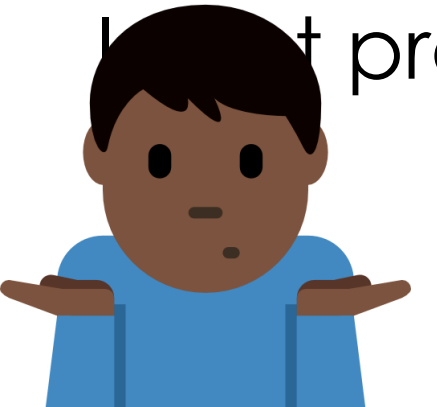
A Few Sample Problems

Given a 10 dimensional array in input, find the sum of first, third, fifth … i.e. odd location elements of the array and sum of even location elements in the array

Given two lists of integers on two separate lines, find the smallest integer in the first list that is not present in the second list.

In second problem, have to revisit list elements – go back and forth – arrays allow conveniently storing the full list

In first problem, need not store the entire list first



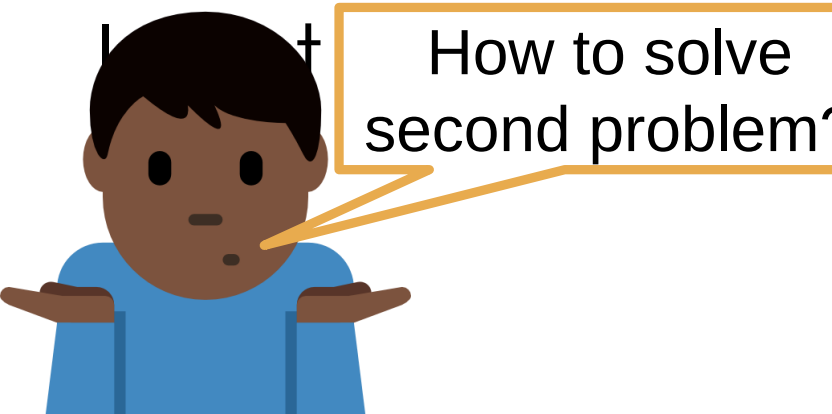
A Few Sample Problems

Given a 10 dimensional array in input, find the sum of first, third, fifth … i.e. odd location elements of the array and sum of even location elements in the array

Given two lists of integers on two separate lines, find the smallest integer in the first list that is not present in the second list.

In second problem, have to revisit list elements – go back and forth – arrays allow conveniently storing the full list

I t ed not store the entire list first



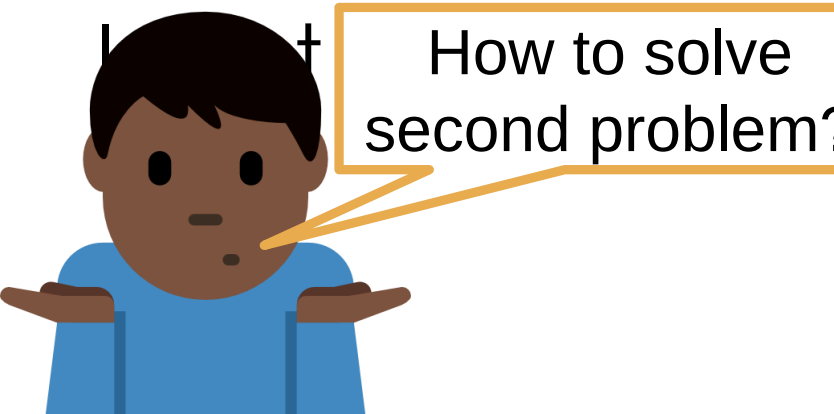
A Few Sample Problems

Given a 10 dimensional array in input, find the sum of first, third, fifth … i.e. odd location elements of the array and sum of even location elements in the array

Given two lists of integers on two separate lines, find the smallest integer in the first list that is not present in the second list.

In second problem, have to revisit list elements – go back and forth – arrays allow conveniently storing the full list

I[illegible]t[illegible]ed not store the entire list first



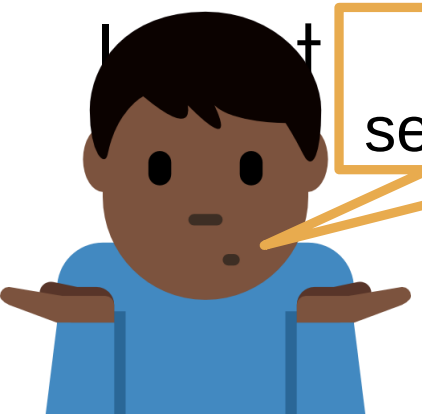
A Few Sample Problems

Given a 10 dimensional array in input, find the sum of first, third, fifth … i.e. odd location elements of the array and sum of even location elements in the array

Given two lists of integers on two separate lines, find the smallest integer in the first list that is not present in the second list.

In second problem, ho back and forth – arrays allow st

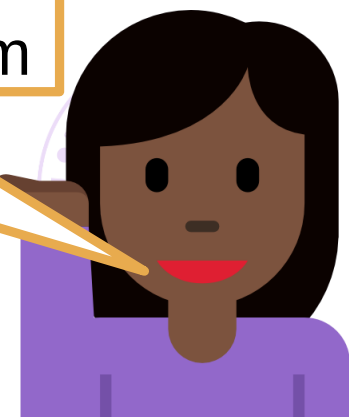
ed r



Break it up into smaller sub-problems

Sub-prob 1: Find all integers in first array that are not present in second array

Sub-prob 2: Give the above numbers, find the smallest using a running-minimum



A Few Sample Problems

Given a 10 dimensional array in input, find the sum of first, third, fifth … i.e. odd location elements of the array and sum of even location elements in the array

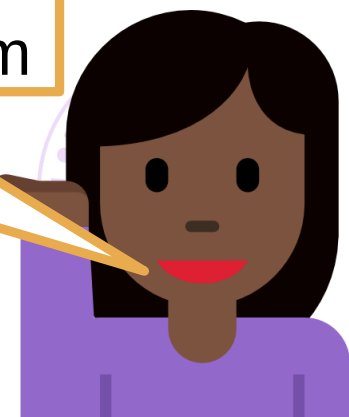
Given two lists of integers on two separate lines, find the smallest integer in the first list that is not present in the second list.

In second problem, ho back and forth – arrays allow st ed r

Break it up into smaller sub-problems

Sub-prob 1: Find all integers in first array that are not present in second array

Sub-prob 2: Give the above numbers, find the smallest using a running-minimum

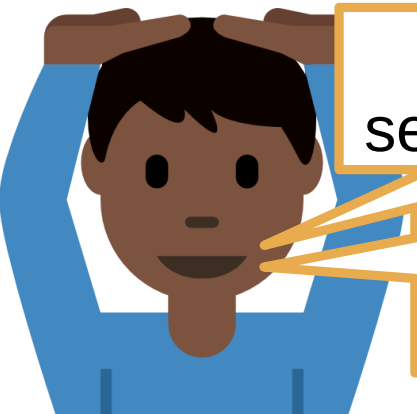


A Few Sample Problems

Given a 10 dimensional array in input, find the sum of first, third, fifth … i.e. odd location elements of the array and sum of even location elements in the array

Given two lists of integers on two separate lines, find the smallest integer in the first list that is not present in the second list.

In second problem, ho back and forth – arrays allov st ed r



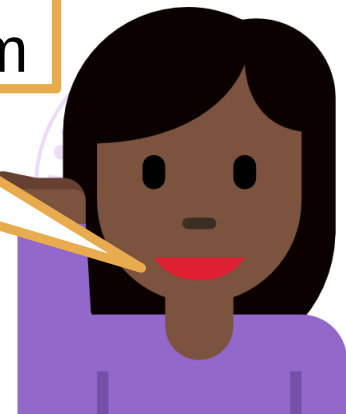
How to solve second problem?

I already know how to solve both sub-problems!

Break it up into smaller sub-problems

Sub-prob 1: Find all integers in first array that are not present in second array

Sub-prob 2: Give the above numbers, find the smallest using a running-minimum

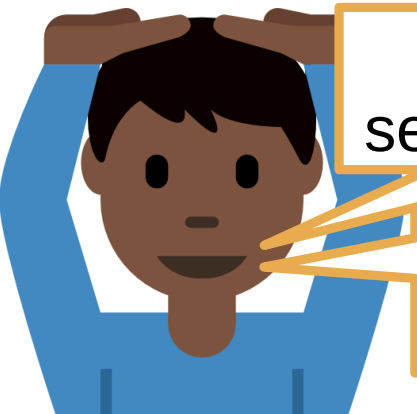


A Few Sample Problems

Given a 10 dimensional array in input, find the sum of first, third, fifth … i.e. odd location elements of the array and sum of even location elements in the array

Given two lists of integers on two separate lines, find the smallest integer in the first list that is not present in the second list.

In second problem, ho back and forth – arrays allow st ed r



How to solve second problem?

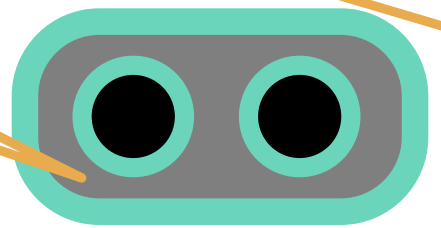
I already know how to solve both sub-problems!

Break it up into smaller sub-problems

Sub-prob 1: Find all integers in first array that are not present in second array

Sub-prob 2: Give the above numbers, find the smallest using a running-minimum

Then let's code!



Streaming and Online Problems

Streaming and Online Problems

Many problems involving a list of integers can be solved without storing the entire list in an array first

Streaming and Online Problems

Many problems involving a list of integers can be solved without storing the entire list in an array first

- Find sum of all odd location integers in a list

Streaming and Online Problems

Many problems involving a list of integers can be solved without storing the entire list in an array first

- Find sum of all odd location integers in a list
- Find sum of all odd integers in a list

Streaming and Online Problems

Many problems involving a list of integers can be solved without storing the entire list in an array first

- Find sum of all odd location integers in a list
- Find sum of all odd integers in a list
- Find longest sequence of consecutive ones in a list
- etc, etc …

Streaming and Online Problems

Many problems involving a list of integers can be solved without storing the entire list in an array first

- Find sum of all odd location integers in a list
- Find sum of all odd integers in a list
- Find longest sequence of consecutive ones in a list
- etc, etc …

Such problems often called online or streaming problems

Streaming and Online Problems

Many problems involving a list of integers can be solved without storing the entire list in an array first

- Find sum of all odd location integers in a list
- Find sum of all odd integers in a list
- Find longest sequence of consecutive ones in a list
- etc, etc …

Such problems often called online or streaming problems

Streaming and online algorithms are critical in machine learning, big data processing

Streaming and Online Problems

Many problems involving a list of integers can be solved without storing the entire list in an array first

- Find sum of all odd location integers in a list
- Find sum of all odd integers in a list
- Find longest sequence of consecutive ones in a list
- etc, etc …

Such problems often called online or streaming problems

Streaming and online algorithms are critical in machine learning, big data processing

- Companies like Google cannot afford to store all data

Streaming and Online Problems

Many problems involving a list of integers can be solved without storing the entire list in an array first

- Find sum of all odd location integers in a list
- Find sum of all odd integers in a list
- Find longest sequence of consecutive ones in a list
- etc, etc …

Such problems often called online or streaming problems

Streaming and online algorithms are critical in machine learning, big data processing

- Companies like Google cannot afford to store all data
- Get data, process it, and throw it away

Streaming and Online Problems

Many problems involving a list of integers can be solved without storing the entire list in an array first

- Find sum of all odd location integers in a list
- Find sum of all odd integers in a list
- Find longest sequence of consecutive ones in a list
- etc, etc …

Such problems often called online or streaming problems

Streaming and online algorithms are critical in machine learning, big data processing

- Companies like Google cannot afford to store all data
- Get data, process it, and throw it away
- Entire areas devoted to this: data streaming, online machine learning

More on initializing arrays

More on initializing arrays

Can be initialized at time of declaration itself

More on initializing arrays

Can be initialized at time of declaration itself

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

More on initializing arrays

Can be initialized at time of declaration itself

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

More on initializing arrays

Can be initialized at time of declaration itself

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

More on initializing arrays

Can be initialized at time of declaration itself

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

More on initializing arrays

Can be initialized at time of declaration itself

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

More on initializing arrays

Can be initialized at time of declaration itself

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

More on initializing arrays

Can be initialized at time of declaration itself

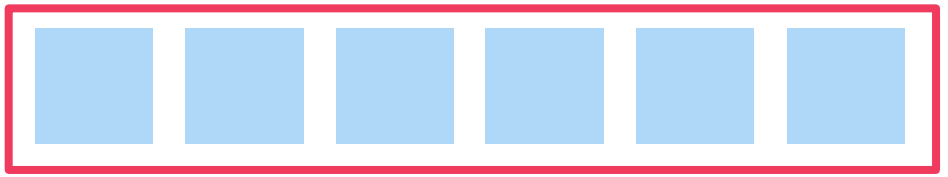
```
int a[6] = {3,7,6,2,1,0};
```

More on initializing arrays

Can be initialized at time of declaration itself

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

a

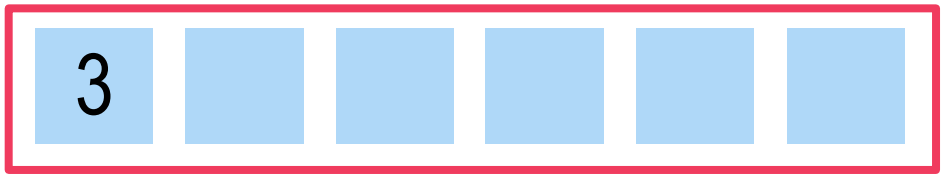


More on initializing arrays

Can be initialized at time of declaration itself

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

a

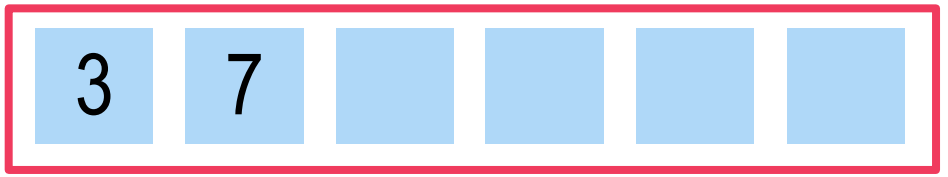


More on initializing arrays

Can be initialized at time of declaration itself

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

a

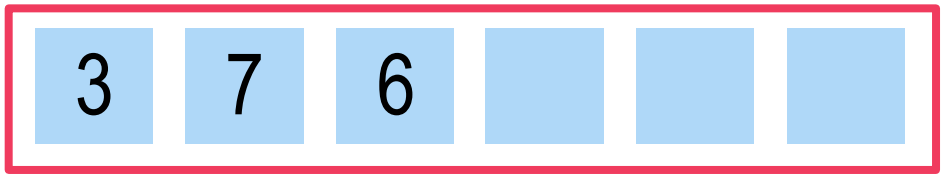


More on initializing arrays

Can be initialized at time of declaration itself

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

a

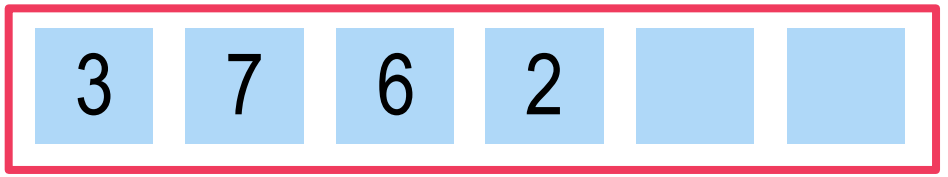


More on initializing arrays

Can be initialized at time of declaration itself

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

a

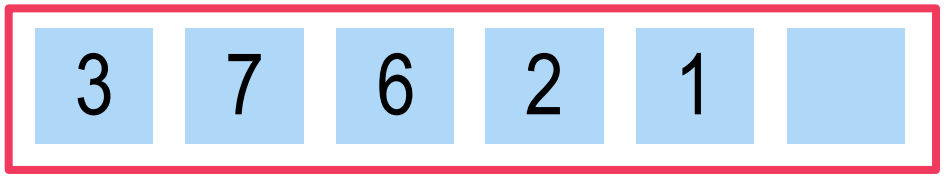


More on initializing arrays

Can be initialized at time of declaration itself

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

a

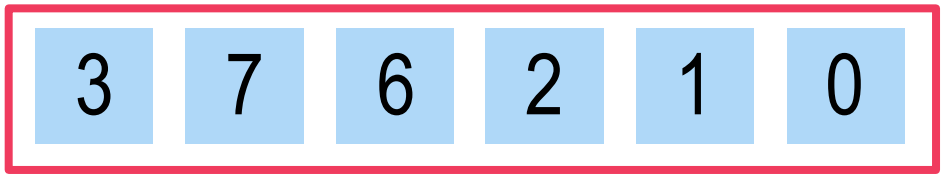


More on initializing arrays

Can be initialized at time of declaration itself

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

a

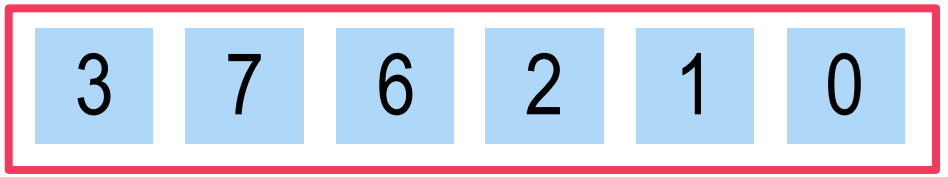


More on initializing arrays

Can be initialized at time of declaration itself

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

a



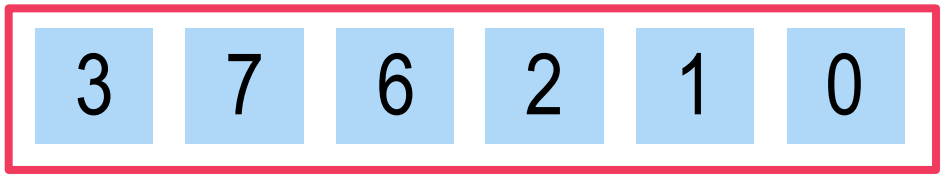
Can be partly initialized as well

More on initializing arrays

Can be initialized at time of declaration itself

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

a



Can be partly initialized as well

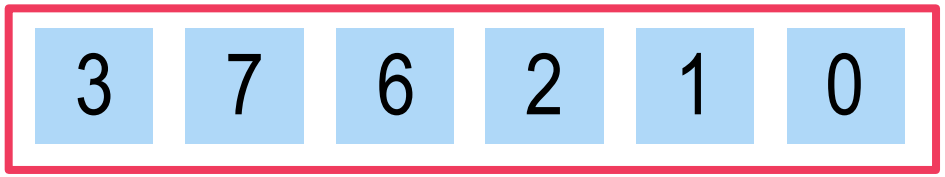
```
int a[6] = {3,7,6};
```

More on initializing arrays

Can be initialized at time of declaration itself

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

a



Can be partly initialized as well

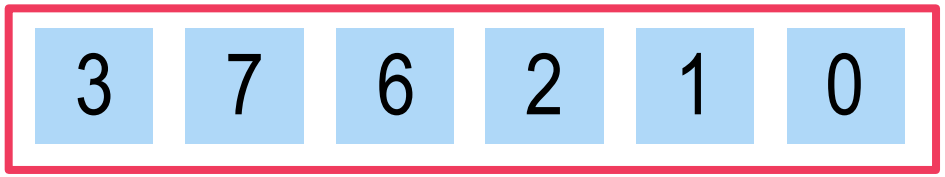
```
int a[6] = {3,7,6};
```

More on initializing arrays

Can be initialized at time of declaration itself

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

a



Can be partly initialized as well

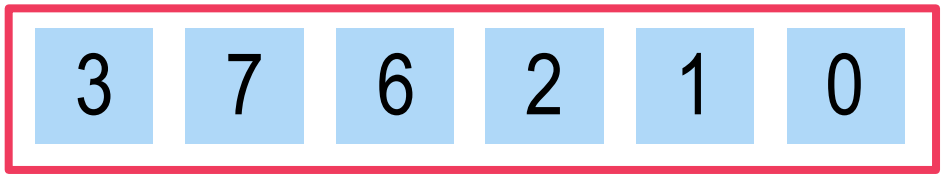
```
int a[6] = {3,7,6};
```

More on initializing arrays

Can be initialized at time of declaration itself

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

a



Can be partly initialized as well

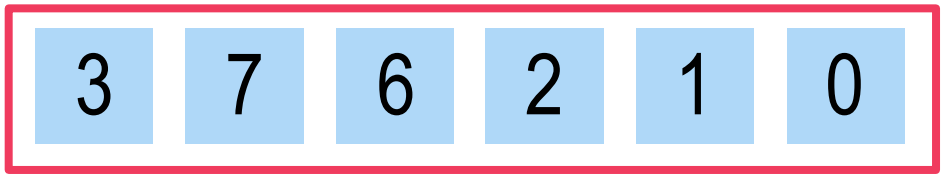
```
int a[6] = {3,7,6};
```

More on initializing arrays

Can be initialized at time of declaration itself

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

a



Can be partly initialized as well

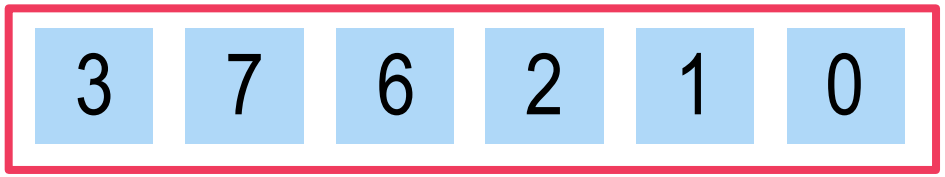
```
int a[6] = {3,7,6};
```

More on initializing arrays

Can be initialized at time of declaration itself

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

a



Can be partly initialized as well

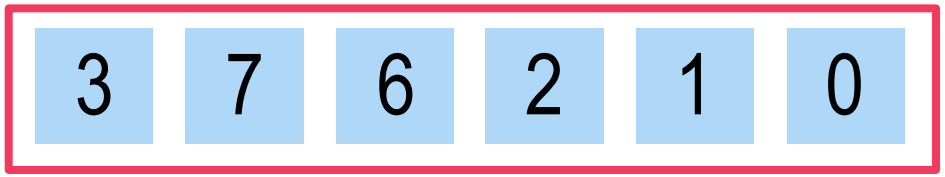
```
int a[6] = {3,7,6};
```

More on initializing arrays

Can be initialized at time of declaration itself

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

a



Can be partly initialized as well

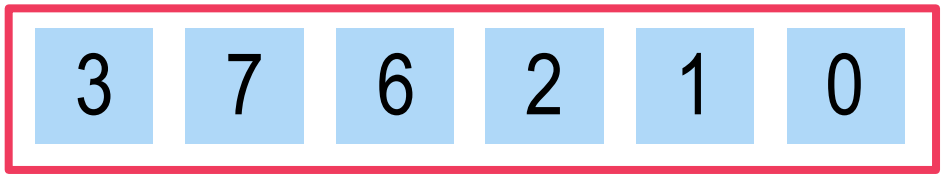
```
int a[6] = {3,7,6};
```

More on initializing arrays

Can be initialized at time of declaration itself

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

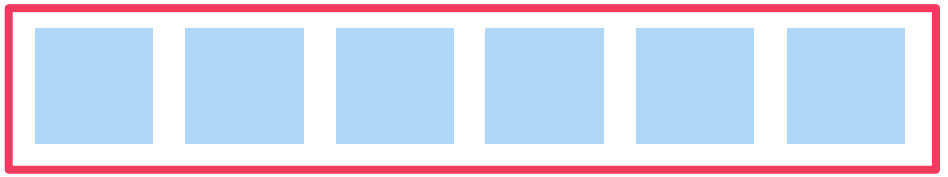
a



Can be partly initialized as well

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

a

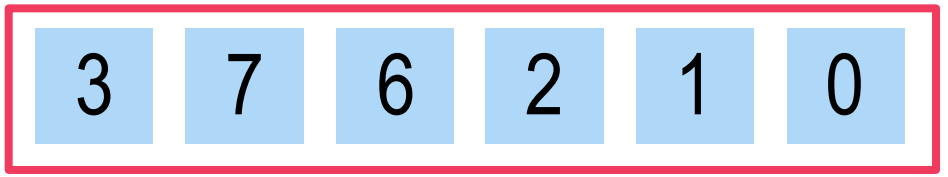


More on initializing arrays

Can be initialized at time of declaration itself

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

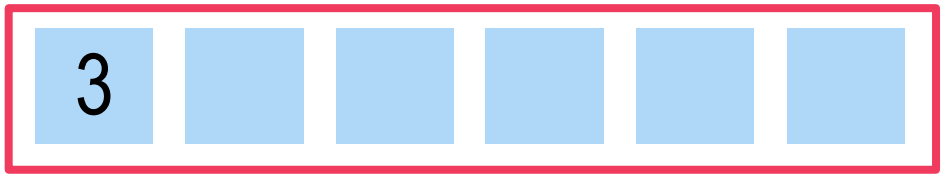
a



Can be partly initialized as well

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

a

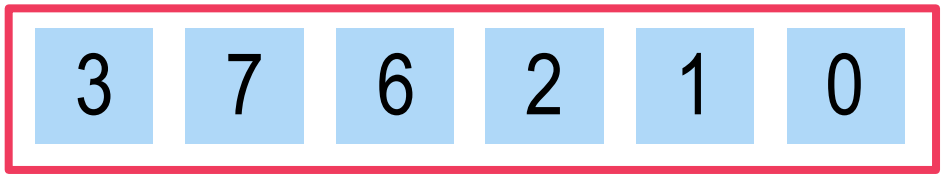


More on initializing arrays

Can be initialized at time of declaration itself

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

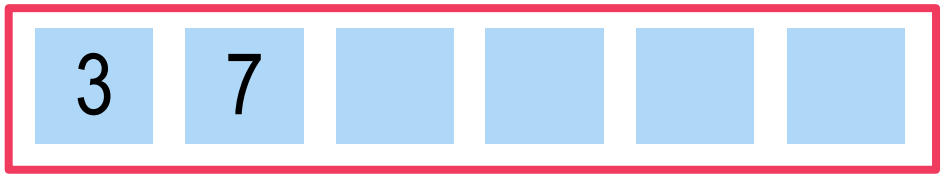
a



Can be partly initialized as well

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

a

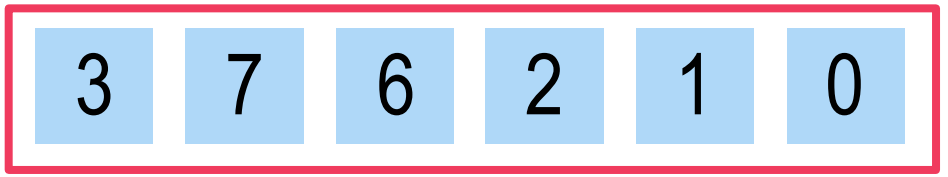


More on initializing arrays

Can be initialized at time of declaration itself

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

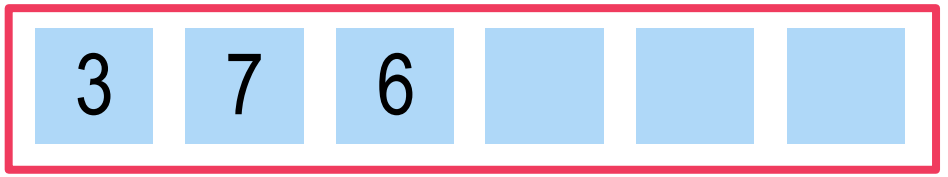
a



Can be partly initialized as well

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

a

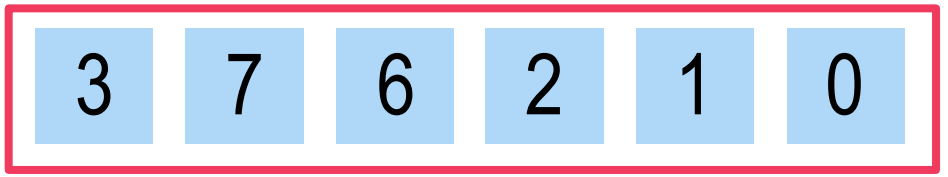


More on initializing arrays

Can be initialized at time of declaration itself

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

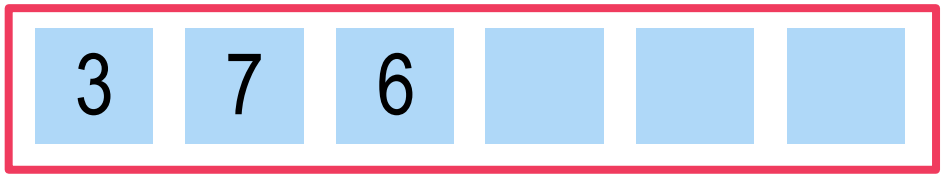
a



Can be partly initialized as well

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

a



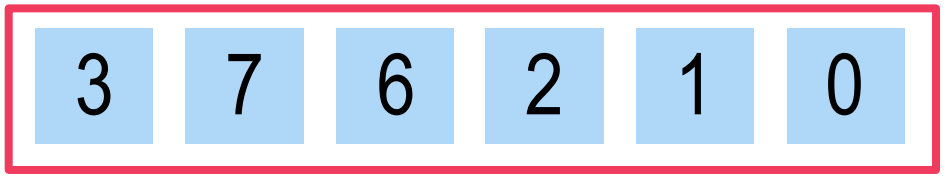
However, after declaration done, have to be initialized one by one!

More on initializing arrays

Can be initialized at time of declaration itself

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

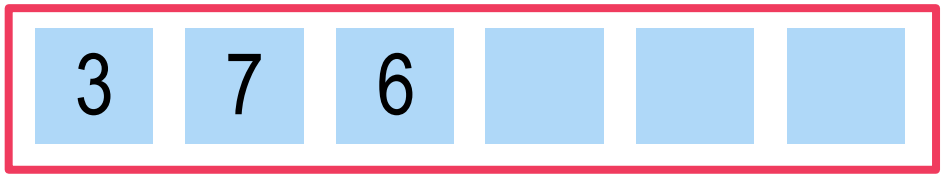
a



Can be partly initialized as well

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

a



However, after declaration done, have to be initialized one by one!

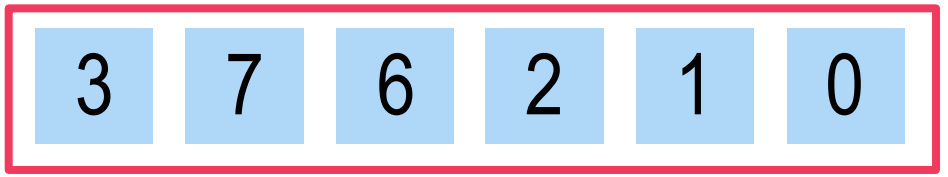
```
int a[6];
a = {3,7,6,2,1,0};
```

More on initializing arrays

Can be initialized at time of declaration itself

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

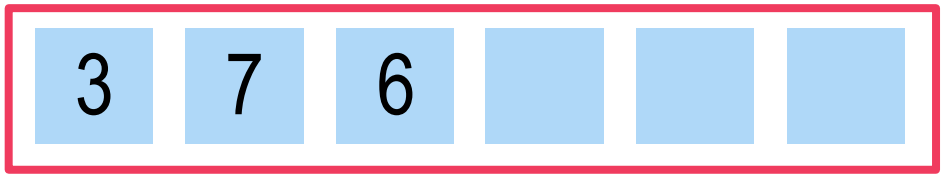
a



Can be partly initialized as well

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

a



However, after declaration done, have to be initialized one by one!

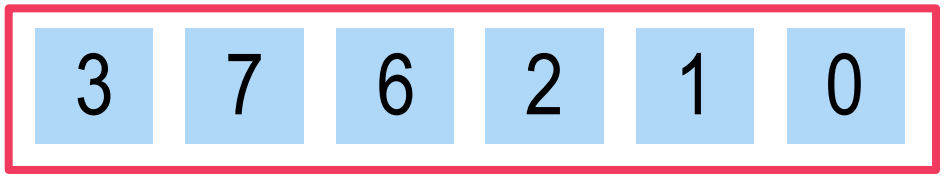
```
int a[6];
a = {3,7,6,2,1,0};
```

More on initializing arrays

Can be initialized at time of declaration itself

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

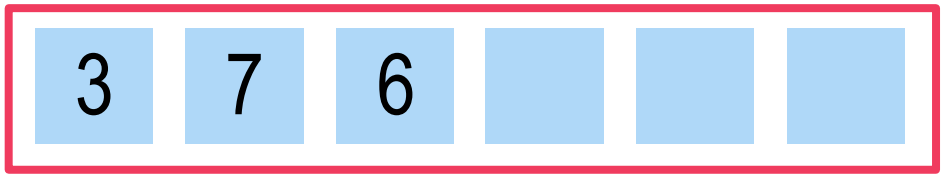
a



Can be partly initialized as well

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

a



However, after declaration done, have to be initialized one by one!

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

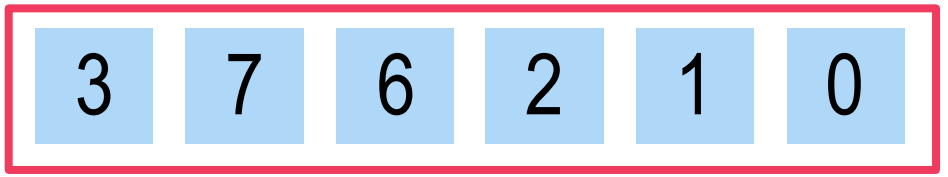
```
int a[6];
a[2] = 6;
```

More on initializing arrays

Can be initialized at time of declaration itself

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

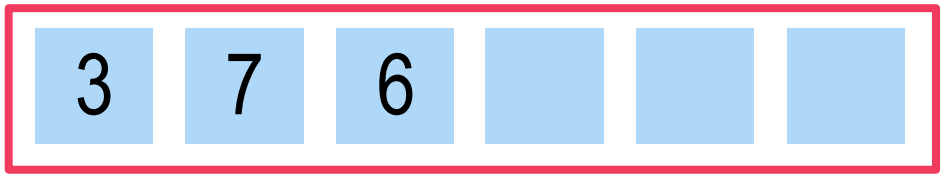
a



Can be partly initialized as well

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

a



However, after declaration done, have to be initialized one by one!

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

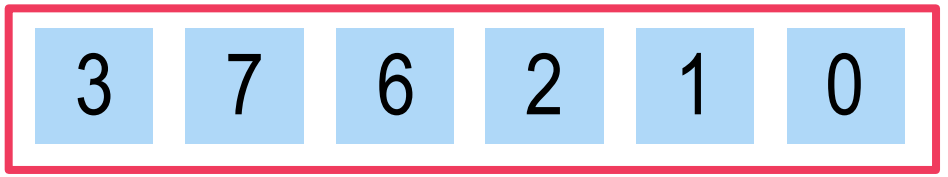
```
int a[6];
a[2] = 6;
```

More on initializing arrays

Can be initialized at time of declaration itself

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

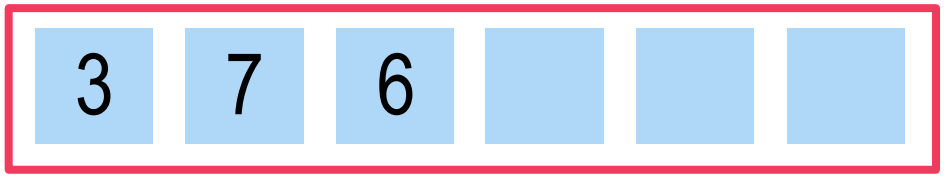
a



Can be partly initialized as well

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

a

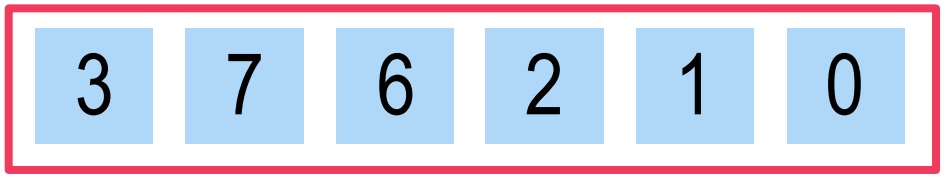


More on initializing arrays

Can be initialized at time of declaration itself

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

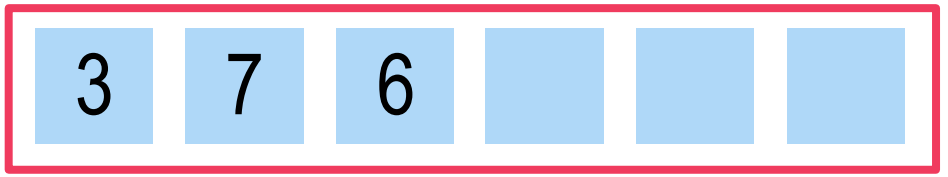
a



Can be partly initialized as well

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

a



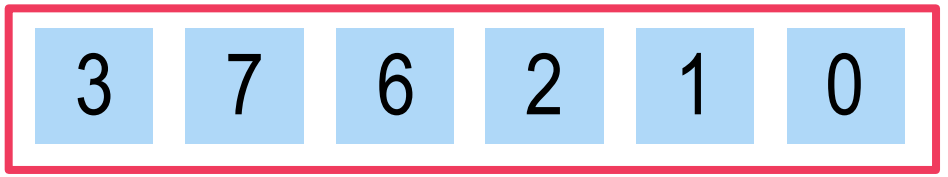
Over initialization may crash

More on initializing arrays

Can be initialized at time of declaration itself

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

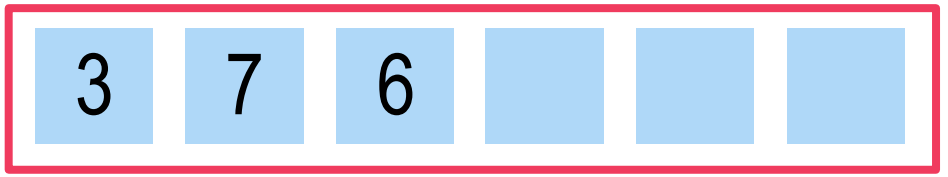
a



Can be partly initialized as well

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

a



Over initialization may crash

```
int a[6] = {1,2,3,4,5,6,7,8,9};
```

More on initializing arrays

Can be initialized at time of declaration itself

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

a

3	7	6	2	1	0

Can be partly initialized as well

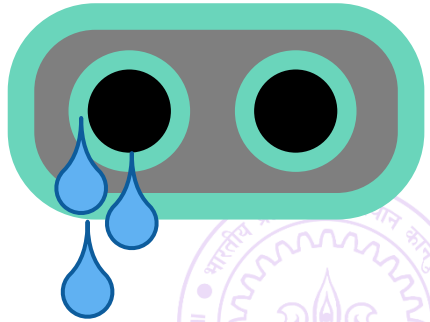
```
int a[6] = {3,7,6};
```

a

3	7	6			

Over initialization may crash

```
int a[6] = {1,2,3,4,5,6,7,8,9};
```



More on initializing arrays

Can be initialized at time of declaration itself

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

a

3	7	6	2	1	0

Can be partly initialized as well

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

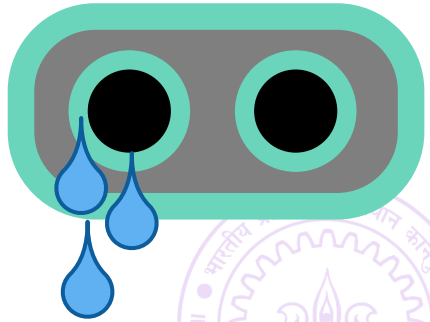
a

3	7	6			

Over initialization may crash

```
int a[6] = {1,2,3,4,5,6,7,8,9};
```

Better way to initialize is the following



More on initializing arrays

Can be initialized at time of declaration itself

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

a

3	7	6	2	1	0

Can be partly initialized as well

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

a

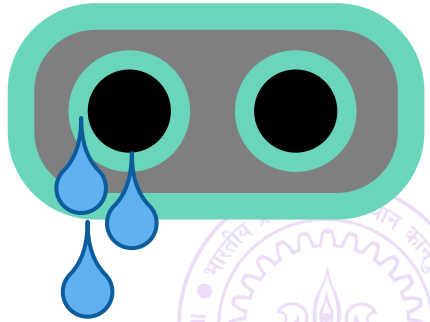
3	7	6			

Over initialization may crash

```
int a[6] = {1,2,3,4,5,6,7,8,9};
```

Better way to initialize is the following

```
int a[] = {1,2,3,4,5,6,7,8,9};
```



More on initializing arrays

Can be initialized at time of declaration itself

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

a

3	7	6	2	1	0

Can be partly initialized as well

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

a

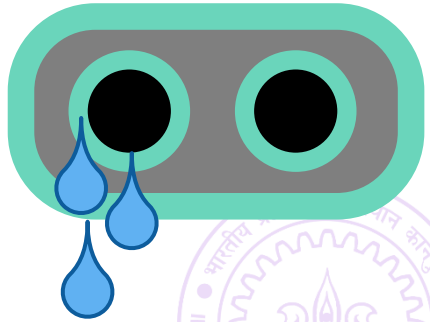
3	7	6			

Over initialization may crash

```
int a[6] = {1,2,3,4,5,6,7,8,9};
```

Better way to initialize is the following

```
int a[] = {1,2,3,4,5,6,7,8,9};
```



More on initializing arrays

Can be initialized at time of declaration itself

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

a

3	7	6	2	1	0

Can be partly initialized as well

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

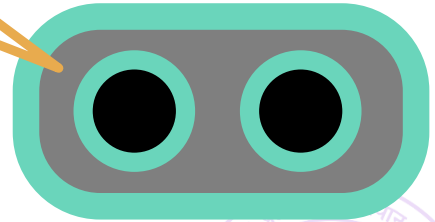
a

3	7	6			

Over initialization may crash

```
int a[6] = {1,2,3,4,5,6,7,8,9};
```

I will figure out how much space needed



Better way to initialize is the following

```
int a[] = {1,2,3,4,5,6,7,8,9};
```

More on initializing arrays

Can be initialized at time of declaration itself

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

a

3	7	6	2	1	0

Can be partly initialized as well

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

a

3	7	6			

Over initialization may crash

```
int a[6] = {1,2,3,4,5,6,7,8,9};
```

Better way to initialize is the following

```
int a[] = {1,2,3,4,5,6,7,8,9};
```

Warning: uninitialized arrays contain garbage, not zeros

More on initializing arrays

Can be initialized at time of declaration itself

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

a

3	7	6	2	1	0

Can be partly initialized as well

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

a

3	7	6			

Over initialization may crash

```
int a[6] = {1,2,3,4,5,6,7,8,9};
```

Better way to initialize is the following

```
int a[] = {1,2,3,4,5,6,7,8,9};
```

Warning: uninitialized arrays contain garbage, not zeros

Highly compiler dependent feature

More about arrays

More about arrays

Arrays can not be copied like normal variables

More about arrays

Arrays can not be copied like normal variables

```
int a[3] = {1,2,3}, b[3];
b = a;
```

More about arrays

Arrays can not be copied like normal variables

```
int a[3] = {1,2,3}, b[3];
b = a;
```

More about arrays

Arrays can not be copied like normal variables

```
int a[3] = {1,2,3}, b[3];
b = a;
```

Will result in an error

More about arrays

Arrays can not be copied like normal variables

```
int a[3] = {1,2,3}, b[3];
b = a;
```

Will result in an error

Arrays can also not be checked for equality directly

More about arrays

Arrays can not be copied like normal variables

```
int a[3] = {1,2,3}, b[3];
b = a;
```

Will result in an error

Arrays can also not be checked for equality directly

```
int a[3] = {1,2,3}, b[3] = {4,5,6};
if(b == a) printf("Equal");
```

More about arrays

Arrays can not be copied like normal variables

```
int a[3] = {1,2,3}, b[3];
b = a;
```

Will result in an error

Arrays can also not be checked for equality directly

```
int a[3] = {1,2,3}, b[3] = {4,5,6};
if(b == a) printf("Equal");
```

More about arrays

Arrays can not be copied like normal variables

```
int a[3] = {1,2,3}, b[3];
b = a;
```

Will result in an error

Arrays can also not be checked for equality directly

```
int a[3] = {1,2,3}, b[3] = {4,5,6};
if(b == a) printf("Equal");
```

Will not result in error but b == a will always be false

More about arrays

Arrays can not be copied like normal variables

```
int a[3] = {1,2,3}, b[3];
b = a;
```

Will result in an error

Arrays can also not be checked for equality directly

```
int a[3] = {1,2,3}, b[3] = {4,5,6};
if(b == a) printf("Equal");
```

Will not result in error but b == a will always be false

Reason will be clear in a couple of weeks

More about arrays

Arrays can not be copied like normal variables

```
int a[3] = {1,2,3}, b[3];
b = a;
```

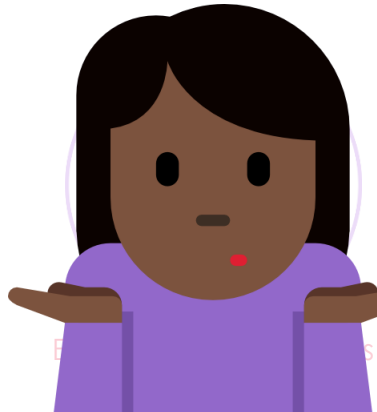
Will result in an error

Arrays can also not be checked for equality directly

```
int a[3] = {1,2,3}, b[3] = {4,5,6};
if(b == a) printf("Equal");
```

Will not result in error but b == a will always be false

Reason will be clear in a couple of weeks



More about arrays

Arrays can not be copied like normal variables

```
int a[3] = {1,2,3}, b[3];
b = a;
```

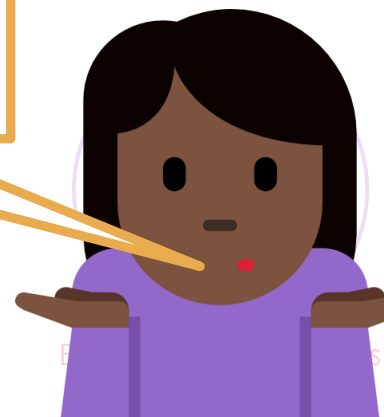
Will result in an error

Arrays can also not be checked for equality directly

```
int a[3] = {1,2,3}, b[3] = {4,5,6};
if(b == a) printf("Equal");
```

Will not result in error but b == a w

Reason will be clear in a couple of weeks



More about arrays

Arrays can not be copied like normal variables

```
int a[3] = {1,2,3}, b[3];
b = a;
```

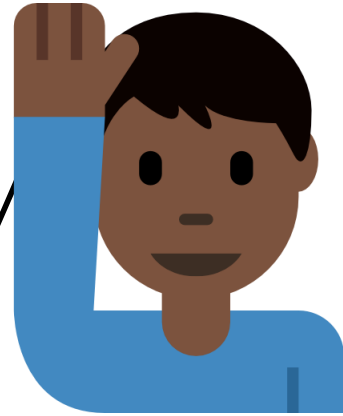
Will result in an error

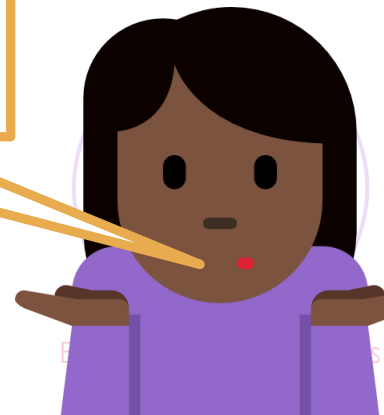
Arrays can also not be checked for equality directly

```
int a[3] = {1,2,3}, b[3] = {4,5,6};
if(b == a) printf("Equal");
```

Will not result in error but b == a w

Reason will be clear in a couple of weeks





More about arrays

Arrays can not be copied like normal variables

```
int a[3] = {1,2,3}, b[3];
b = a;
```

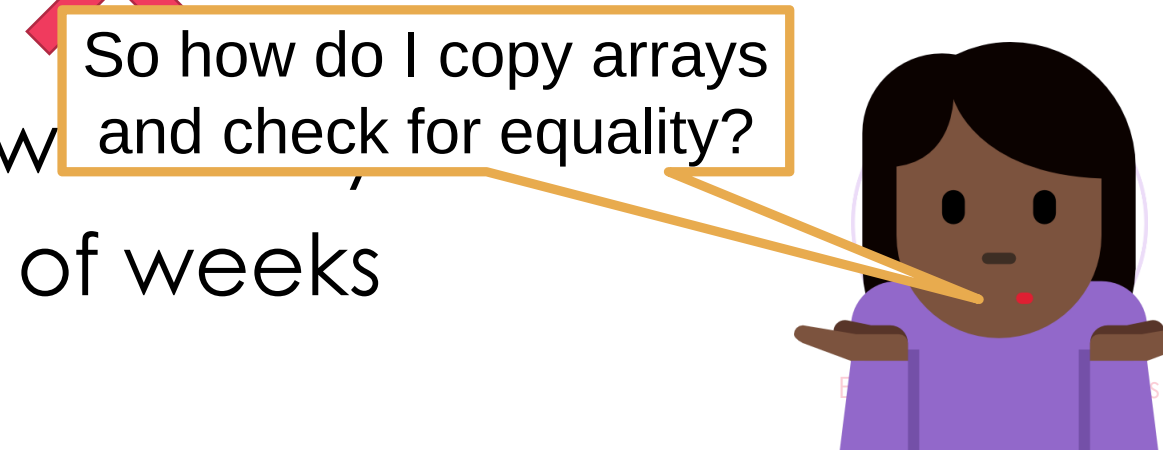
Will result in an error

Arrays can also not be checked for equality directly

```
int a[3] = {1,2,3}, b[3] = {4,5,6};
if(b == a) printf("Equal");
```

Will not result in error but b == a w

Reason will be clear in a couple of weeks



More about arrays

Arrays can not be copied like normal variables

```
int a[3] = {1,2,3}, b[3];
b = a;
```

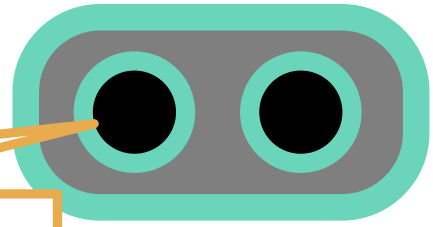
Will result in an error

Arrays can also not be checked for equality directly

```
int a[3] = {1,2,3}, b[3] = {4,5,6};
if(b == a) printf("Equal");
```

Will not result in error but b == a w

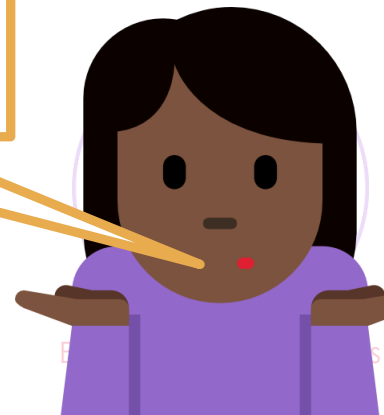
Reason will be clear in a couple of weeks



For strings (basically char arrays), nice library functions exist to do assignment, equality checks

Do it yourself, element by element, using a for loop

So how do I copy arrays and check for equality?



More about arrays

Arrays can not be copied like normal variables

```
int a[3] = {1,2,3}, b[3];
b = a;
```

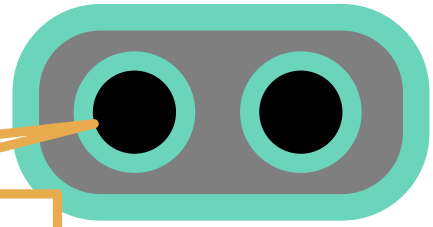
Will result in an error

Arrays can also not be checked for equality directly

```
int a[3] = {1,2,3}, b[3] = {4,5,6};
if(b == a) printf("Equal");
```

Will not result in error but b == a w

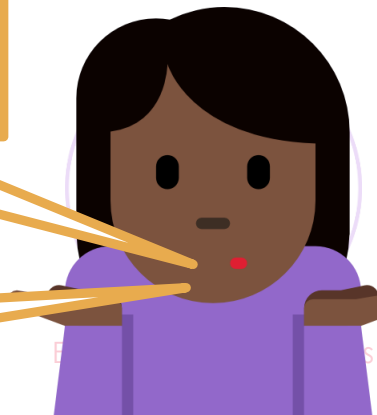
Reason will be clear in a couple of weeks



For strings (basically char arrays), nice library functions exist to do assignment, equality checks

Do it yourself, element by element, using a for loop

So how do I copy arrays and check for equality?



What is a char?

Char: new datatype

Char: new datatype

Close cousin of the int and long datatypes

Char: new datatype

Close cousin of the int and long datatypes

Internally stored as an integer between 0 and 127

Char: new datatype

Close cousin of the int and long datatypes

Internally stored as an integer between 0 and 127

Char: new datatype

Close cousin of the int and long datatypes

Internally stored as an integer between 0 and 127

```
#include <stdio.h>
```

Char: new datatype

Close cousin of the int and long datatypes

Internally stored as an integer between 0 and 127

```
#include <stdio.h>
int main(){
```

Char: new datatype

Close cousin of the int and long datatypes

Internally stored as an integer between 0 and 127

```
#include <stdio.h>
int main(){
char a = ‘p’;
```

Char: new datatype

Close cousin of the int and long datatypes

Internally stored as an integer between 0 and 127

```
#include <stdio.h>
int main(){
char a = ‘p’;
scanf(“%c”, &a);
```

Char: new datatype

Close cousin of the int and long datatypes

Internally stored as an integer between 0 and 127

```
#include <stdio.h>
int main(){
char a = ‘p’;
scanf(“%c”, &a);
printf(“My first char %c”, a);
```

Char: new datatype

Close cousin of the int and long datatypes

Internally stored as an integer between 0 and 127

```
#include <stdio.h>
int main(){
char a = 'p';
scanf("%c", &a);
printf("My first char %c", a);
return 0;
```

Char: new datatype

Close cousin of the int and long datatypes

Internally stored as an integer between 0 and 127

```
#include <stdio.h>
int main(){
char a = ‘p’;
scanf(“%c”, &a);
printf(“My first char %c”, a);
return 0;
}
```

Char: new datatype

Close cousin of the int and long datatypes

Internally stored as an integer between 0 and 127

```
#include <stdio.h>
int main(){
char a = 'p';
scanf("%c", &a);
printf("My first char %c", a);
return 0;
}
```


a

Char: new datatype

Close cousin of the int and long datatypes

Internally stored as an integer between 0 and 127

```
#include <stdio.h>
int main(){
char a = 'p';
scanf("%c", &a);
printf("My first char %c", a);
return 0;
}
```

p

a

Char: new datatype

Close cousin of the int and long datatypes

Internally stored as an integer between 0 and 127

```
#include <stdio.h>
int main(){
char a = 'p';
scanf("%c", &a);
printf("My first char %c", a);
return 0;
}
```

p

a

%c

Char: new datatype

Close cousin of the int and long datatypes

Internally stored as an integer between 0 and 127

```
#include <stdio.h>
int main(){
char a = 'p';
scanf("%c", &a);
printf("My first char %c", a);
return 0;
}
```

p

a

%c

Char constants enclosed in ' '

Char: new datatype

Close cousin of the int and long datatypes

Internally stored as an integer between 0 and 127

```
#include <stdio.h>
int main(){
char a = 'p';
scanf("%c", &a);
printf("My first char %c", a);
return 0;
}
```

p

a

%c

Char constants enclosed in ' '

Integer arithmetic applies to char as well +, -, /, *, %, ()

Char: new datatype

Close cousin of the int and long datatypes

Internally stored as an integer between 0 and 127

```
#include <stdio.h>
int main(){
char a = 'p';
scanf("%c", &a);
printf("My first char %c", a);
return 0;
}
```

p

a

%c

Char constants enclosed in ' '

Integer arithmetic applies to char as well +, -, /, *, %, ()

Can use it for nice tricks but be careful

Char: new datatype

Close cousin of the int and long datatypes

Internally stored as an integer between 0 and 127

```
#include <stdio.h>
int main(){
char a = 'p';
scanf("%c", &a);
printf("My first char %c", a);
return 0;
}
```

p

a

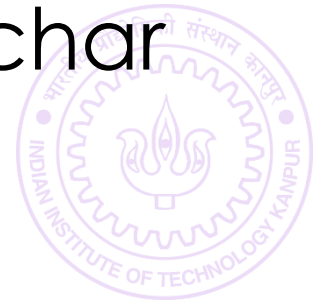
%c

Char constants enclosed in ' '

Integer arithmetic applies to char as well +, -, /, *, %, ()

Can use it for nice tricks but be careful

Can typecast to/from char



Char: new datatype

Close cousin of the int and long datatypes

Internally stored as an integer between 0 and 127

```
#include <stdio.h>
int main(){
char a = 'p';
scanf("%c", &a);
printf("My first char %c", a);
return 0;
}
```

p

a

%c

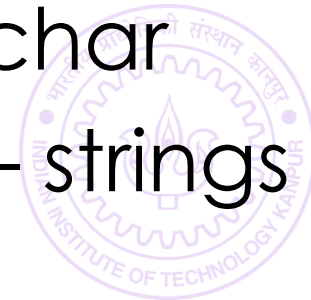
Char constants enclosed in ' '

Integer arithmetic applies to char as well +, -, /, *, %, ()

Can use it for nice tricks but be careful

Can typecast to/from char

Can have char arrays – strings



Char: new datatype

Close cousin of the int and long datatypes

Internally stored as an integer between 0 and 127

```
#include <stdio.h>
int main(){
char a = 'p';
scanf("%c", &a);
printf("My first char %c", a);
return 0;
}
```

p

a

%c

Char constants enclosed in ' '

Integer arithmetic applies to char as well +, -, /, *, %, ()

Can use it for nice tricks but be careful

Can typecast to/from char

Can have char arrays – strings

Case sensitive 'a', 'A' different

IMAGE COURTESY WIKIPEDIA.ORG

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

IMAGE COURTESY WIKIPEDIA.ORG

ASCII TABLE

American Standard Code for Information Interchange

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

IMAGE COURTESY WIKIPEDIA.ORG

ASCII TABLE

American Standard Code for Information Interchange

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

\t

IMAGE COURTESY WIKIPEDIA.ORG

ASCII TABLE

American Standard Code for Information Interchange

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

\t

\n

IMAGE COURTESY WIKIPEDIA.ORG

Take care with char

Take care with char

`char a = p;` Mr C will search for a variable named p.

Take care with char

`char a = p;` Mr C will search for a variable named p.

To assign character constant ‘p’ to a, `char a = ‘p’;`

Take care with char

`char a = p;` Mr C will search for a variable named p.

To assign character constant ‘p’ to a, `char a = ‘p’;`

Note that ‘5’ and 5 are different according to Mr C

Take care with char

`char a = p;` Mr C will search for a variable named p.

To assign character constant ‘p’ to a, `char a = ‘p’;`

Note that ‘5’ and 5 are different according to Mr C

- ‘5’ is a character constant stored internally as the integer 53

Take care with char

`char a = p;` Mr C will search for a variable named p.

To assign character constant ‘p’ to a, `char a = ‘p’;`

Note that ‘5’ and 5 are different according to Mr C

- ‘5’ is a character constant stored internally as the integer 53
- 5 is an integer constant stored internally as the integer 5 itself

Take care with char

`char a = p;` Mr C will search for a variable named p.

To assign character constant ‘p’ to a, `char a = ‘p’;`

Note that ‘5’ and 5 are different according to Mr C

- ‘5’ is a character constant stored internally as the integer 53
- 5 is an integer constant stored internally as the integer 5 itself

Be very careful if mixing %c with other format specifiers like %d, %ld, %f, %lf in scanf and printf statements

Take care with char

`char a = p;` Mr C will search for a variable named p.

To assign character constant ‘p’ to a, `char a = ‘p’;`

Note that ‘5’ and 5 are different according to Mr C

- ‘5’ is a character constant stored internally as the integer 53
- 5 is an integer constant stored internally as the integer 5 itself

Be very careful if mixing %c with other format specifiers like %d, %ld, %f, %lf in scanf and printf statements

getchar(), putchar() shortcuts to read/print single char

Take care with char

`char a = p;` Mr C will search for a variable named p.

To assign character constant ‘p’ to a, `char a = ‘p’;`

Note that ‘5’ and 5 are different according to Mr C

- ‘5’ is a character constant stored internally as the integer 53
- 5 is an integer constant stored internally as the integer 5 itself

Be very careful if mixing %c with other format specifiers like %d, %ld, %f, %lf in scanf and printf statements

getchar(), putchar() shortcuts to read/print single char

When using characters in arithmetic, relational, logical expressions, integer (ASCII) value of character gets used