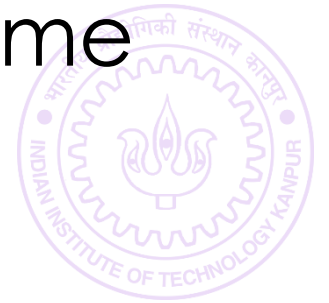
Type Hype with Mr C

ESC101: Foundations of Computing

Purushottam Kar

Announcements

- Special session on using computers
  - August 11, 2018 (coming Saturday), 5PM, NCL CC-02
  - Not a revision class – only for students who are new to computers
- Slides, tutorial sheets, lecture codes on website
  - https://tinyurl.com/esc18-19aw
  - In case the above does not work https://web.cse.iitk.ac.in/users/purushot/courses/esc/2018-19-a/
- Please bring your IITK ID card to lab – come on time



Various kinds of variables in C

Various kinds of variables in C

In math, we have several kinds of numbers
naturals, integers, rationals, reals, complex

Various kinds of variables in C

In math, we have several kinds of numbers
naturals, integers, rationals, reals, complex

We can define variables of all above kinds

Various kinds of variables in C

In math, we have several kinds of numbers
naturals, integers, rationals, reals, complex

We can define variables of all above kinds

Mr C also understands various *types* of variables

Various kinds of variables in C

In math, we have several kinds of numbers
naturals, integers, rationals, reals, complex

We can define variables of all above kinds

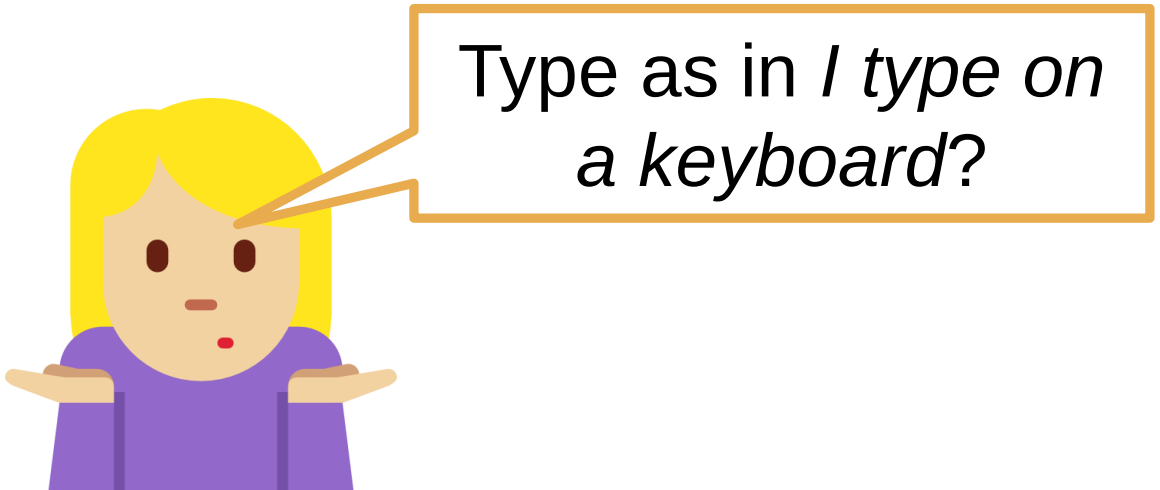
Mr C also understands various *types* of variables

Various kinds of variables in C

In math, we have several kinds of numbers
naturals, integers, rationals, reals, complex

We can define variables of all above kinds

Mr C also understands various *types* of variables

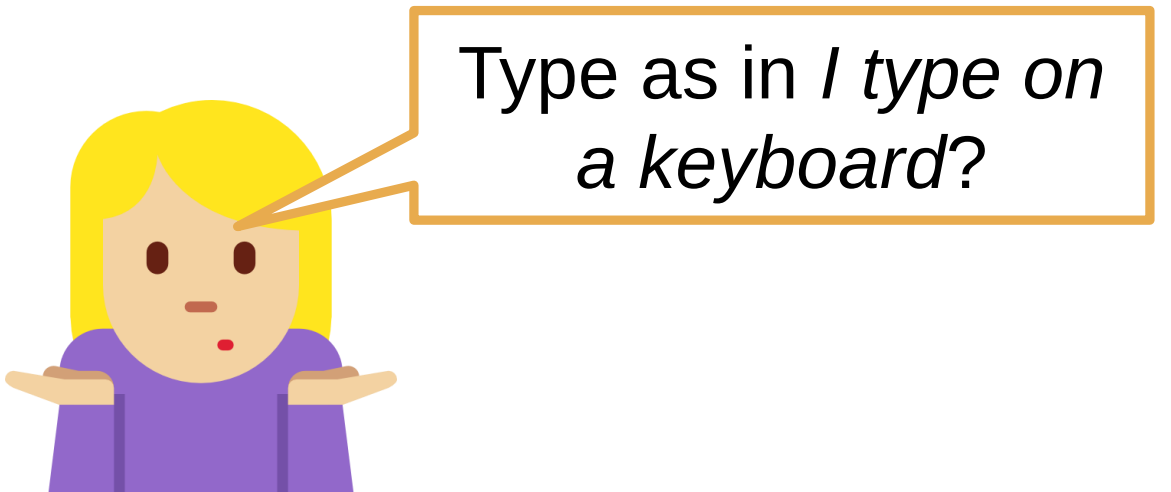


Various kinds of variables in C

In math, we have several kinds of numbers
naturals, integers, rationals, reals, complex

We can define variables of all above kinds

Mr C also understands various *types* of variables

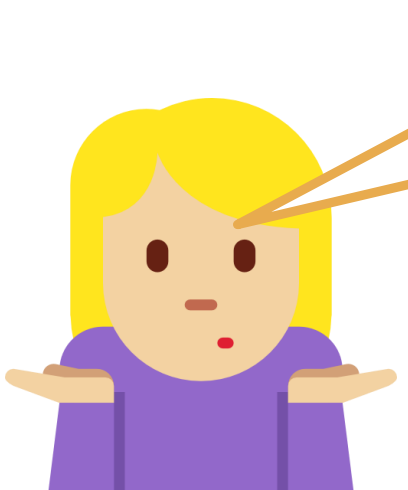


Various kinds of variables in C

In math, we have several kinds of numbers
naturals, integers, rationals, reals, complex

We can define variables of all above kinds

Mr C also understands various *types* of variables



Type as in *I type on a keyboard*?

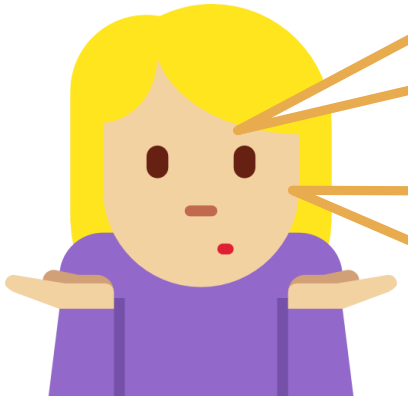
No, as in *I saw two types of mangoes in the store today*

Various kinds of variables in C

In math, we have several kinds of numbers
naturals, integers, rationals, reals, complex

We can define variables of all above kinds

Mr C also understands various *types* of variables



Type as in *I type on a keyboard*?

Ah. So *type* means different kinds of things

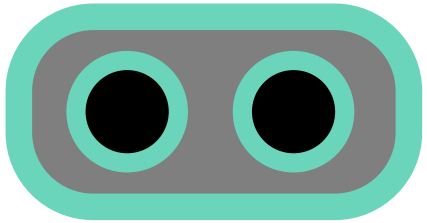
No, as in *I saw two types of mangoes in the store today*

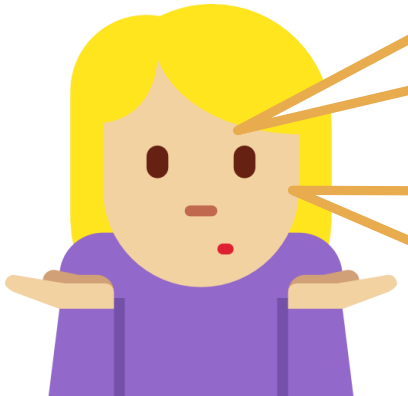
Various kinds of variables in C

In math, we have several kinds of numbers
naturals, integers, rationals, reals, complex

We can define variables of all above kinds

Mr C also understands various *types* of variables





Type as in *I type on a keyboard*?

Ah. So *type* means different kinds of things

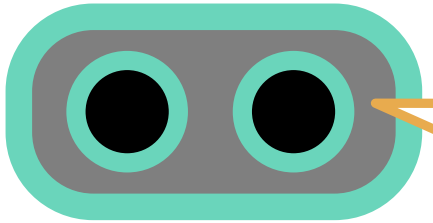
No, as in *I saw two types of mangoes in the store today*

Various kinds of variables in C

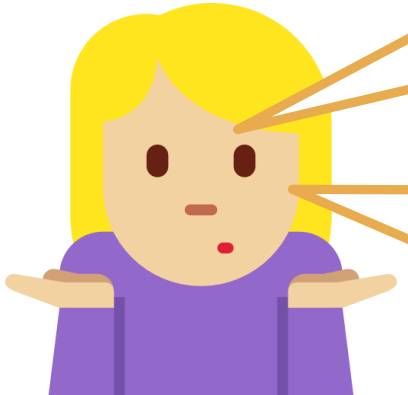
In math, we have several kinds of numbers
naturals, integers, rationals, reals, complex

We can define variables of all above kinds

Mr C also understands various *types* of variables



That's right. I can work with lots of kinds of variables



Type as in *I type on a keyboard*?

Ah. So *type* means different kinds of things

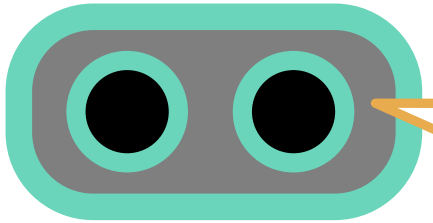
No, as in *I saw two types of mangoes in the store today*

Various kinds of variables in C

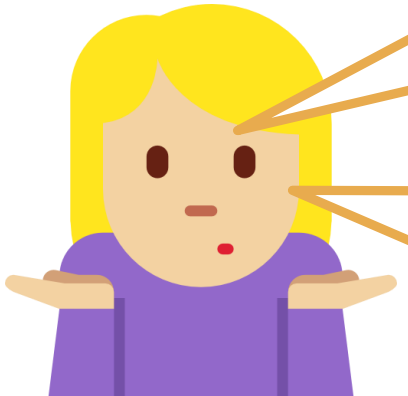
In math, we have several kinds of numbers
naturals, integers, rationals, reals, complex

We can define variables of all above kinds

Mr C also understands various *types* of variables



That's right. I can work with lots of kinds of variables



Type as in *I type on a keyboard*?

Ah. So *type* means different kinds of things

We have already seen int, long and float types in class

No, as in *I saw two types of mangoes in the store today*

Int type

भारतीय प्रौद्योगिकी संस्थान कानपुर
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

Int type

Can store integers b/w -2,147,483,648 and 2,147,483,647

Int type

Shorthand for between

Can store integers b/w -2,147,483,648 and 2,147,483,647

Int type

Shorthand for between

Can store integers b/w -2,147,483,648 and 2,147,483,647

Int type

Shorthand for between

Can store integers b/w -2,147,483,648 and 2,147,483,647

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

Int type

Shorthand for between

Can store integers b/w -2,147,483,648 and 2,147,483,647

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

Int type

Shorthand for between

Can store integers b/w -2,147,483,648 and 2,147,483,647

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

Int type

Shorthand for between

Can store integers b/w -2,147,483,648 and 2,147,483,647

```
#include <stdio.h>
int main(){
int a;
scanf("%d", &a);
```

Int type

Shorthand for between

Can store integers b/w -2,147,483,648 and 2,147,483,647

```
#include <stdio.h>
int main(){
int a;
scanf("%d", &a);
printf("My first int %d", a);
```

Int type

Shorthand for between

Can store integers b/w -2,147,483,648 and 2,147,483,647

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

Int type

Shorthand for between

Can store integers b/w -2,147,483,648 and 2,147,483,647

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

Int type

Shorthand for between

Can store integers b/w -2,147,483,648 and 2,147,483,647

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

%d

Int type

Shorthand for between

Can store integers b/w -2,147,483,648 and 2,147,483,647

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

%d

a

Int type

Shorthand for between

Can store integers b/w -2,147,483,648 and 2,147,483,647

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

%d

a

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

Int type

Shorthand for between

Can store integers b/w -2,147,483,648 and 2,147,483,647

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

%d

a

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

Have worked with them a lot so far

Long int type

Long int type

Really long – can store integers between

Long int type

Really long – can store integers between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807

Long int type

Really long – can store integers between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807

Long int type

Really long – can store integers between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807

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

Long int type

Really long – can store integers between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807

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

Long int type

Really long – can store integers between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807

```
#include <stdio.h>
int main(){
long a; //long int also
```

Long int type

Really long – can store integers between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807

```
#include <stdio.h>
int main(){
long a; //long int also
scanf("%ld", &a);
```

Long int type

Really long – can store integers between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807

```
#include <stdio.h>
int main(){
long a; //long int also
scanf("%ld", &a);
printf("My first long int %ld", a);
```

Long int type

Really long – can store integers between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807

```
#include <stdio.h>
int main(){
long a; //long int also
scanf("%ld", &a);
printf("My first long int %ld", a);
return 0;
```

Long int type

Really long – can store integers between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807

```
#include <stdio.h>
int main(){
long a; //long int also
scanf("%ld", &a);
printf("My first long int %ld", a);
return 0;
}
```

Long int type

Really long – can store integers between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807

```
#include <stdio.h>
int main(){
long a; //long int also
scanf("%ld", &a);
printf("My first long int %ld", a);
return 0;
}
```

Long int type

Really long – can store integers between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807

```
#include <stdio.h>
int main(){
long a; //long int also
scanf("%ld", &a);
printf("My first long int %ld", a);
return 0;
}
```


%ld

Long int type

Really long – can store integers between

-9,223,372,036,854,775,808 and 9,223,372,036,854,775,807

```
#include <stdio.h>
int main(){
long a; //long int also
scanf("%ld", &a);
printf("My first long int %ld", a);
return 0;
}
```


%ld

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

Long int type

Really long – can store integers between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807

```
#include <stdio.h>
int main(){
long a; //long int also
scanf("%ld", &a);
printf("My first long int %ld", a);
return 0;
}
```


%ld

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

Try them out on Prutor

Long int type

Really long – can store integers between

-9,223,372,036,854,775,808 and 9,223,372,036,854,775,807

```
#include <stdio.h>
int main(){
long a; //long int also
scanf("%ld", &a);
printf("My first long int %ld", a);
return 0;
}
```


a

%ld

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

Try them out on Prutor

How does long work with int int + long, int * long?

Long int type

Really long – can store integers between

-9,223,372,036,854,775,808 and 9,223,372,036,854,775,807

```
#include <stdio.h>
int main(){
long a; //long int also
scanf("%ld", &a);
printf("My first long int %ld", a);
return 0;
}
```


a

%ld

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

Try them out on Prutor

How does long work with int int + long, int * long?

Will see today

Float type

भारतीय प्रौद्योगिकी संस्थान कानपुर
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

Float type

int, long allow us to store, do math formulae with integers

Float type

int, long allow us to store, do math formulae with integers

float allows us to store, do math formulae with reals

Float type

int, long allow us to store, do math formulae with integers

float allows us to store, do math formulae with reals

Float type

int, long allow us to store, do math formulae with integers

float allows us to store, do math formulae with reals

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

Float type

int, long allow us to store, do math formulae with integers

float allows us to store, do math formulae with reals

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

Float type

int, long allow us to store, do math formulae with integers

float allows us to store, do math formulae with reals

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

Float type

int, long allow us to store, do math formulae with integers

float allows us to store, do math formulae with reals

```
#include <stdio.h>
int main(){
float a;
scanf("%f", &a);
```

Float type

int, long allow us to store, do math formulae with integers

float allows us to store, do math formulae with reals

```
#include <stdio.h>
int main(){
float a;
scanf("%f", &a);
printf("My first real %f", a);
```

Float type

int, long allow us to store, do math formulae with integers

float allows us to store, do math formulae with reals

```
#include <stdio.h>
int main(){
float a;
scanf("%f", &a);
printf("My first real %f", a);
return 0;
```

Float type

int, long allow us to store, do math formulae with integers

float allows us to store, do math formulae with reals

```
#include <stdio.h>
int main(){
float a;
scanf("%f", &a);
printf("My first real %f", a);
return 0;
}
```

Float type

int, long allow us to store, do math formulae with integers

float allows us to store, do math formulae with reals

```
#include <stdio.h>
int main(){
float a;
scanf("%f", &a);
printf("My first real %f", a);
return 0;
}
```

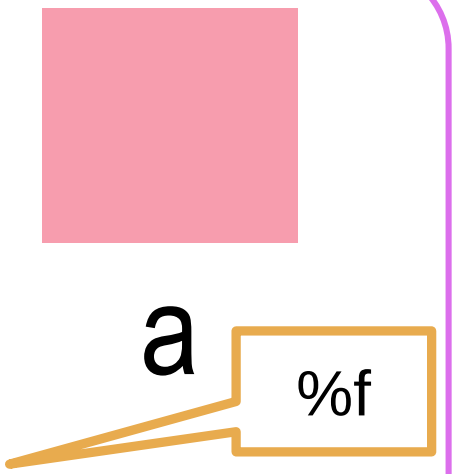
a

Float type

int, long allow us to store, do math formulae with integers

float allows us to store, do math formulae with reals

```
#include <stdio.h>
int main(){
float a;
scanf("%f", &a);
printf("My first real %f", a);
return 0;
}
```

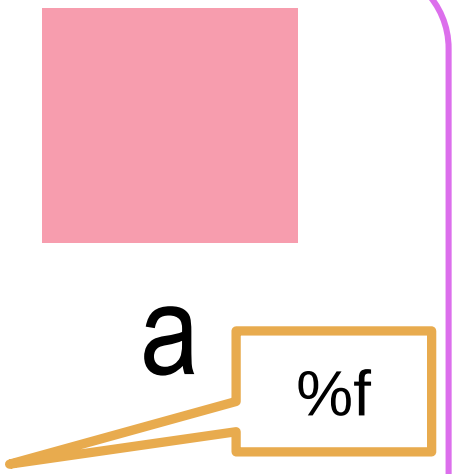


Float type

int, long allow us to store, do math formulae with integers

float allows us to store, do math formulae with reals

```
#include <stdio.h>
int main(){
float a;
scanf("%f", &a);
printf("My first real %f", a);
return 0;
}
```



Very large range ± 3.4e+38

Float type

int, long allow us to store, do math formulae with integers

float allows us to store, do math formulae with reals

```
#include <stdio.h>
int main(){
float a;
scanf("%f", &a);
printf("My first real %f", a);
return 0;
}
```

a

%f

Very large range ± 3.4e+38

Arithmetic operations apply to float as well +, -, /, *, ()

Float type

int, long allow us to store, do math formulae with integers

float allows us to store, do math formulae with reals

```
#include <stdio.h>
int main(){
float a;
scanf("%f", &a);
printf("My first real %f", a);
return 0;
}
```

a

%f

Very large range ± 3.4e+38

Arithmetic operations apply to float as well +, -, /, *, ()

Float type

int, long allow us to store, do math formulae with integers

float allows us to store, do math formulae with reals

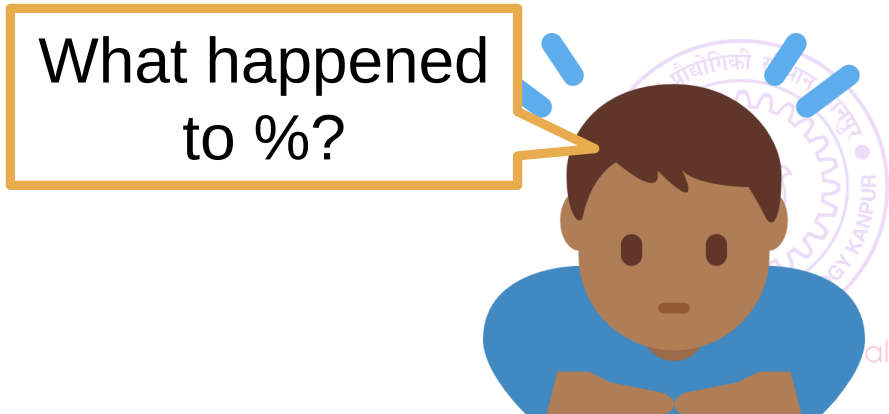
```
#include <stdio.h>
int main(){
float a;
scanf("%f", &a);
printf("My first real %f", a);
return 0;
}
```

a

%f

Very large range ± 3.4e+38

Arithmetic operations apply to float as well +, -, /, *, ()



Float type

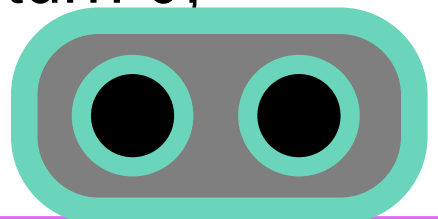
int, long allow us to store, do math formulae with integers

float allows us to store, do math formulae with reals

```
#include <stdio.h>
int main(){
float a;
scanf("%f", &a);
printf("My first real %f", a);
return 0;
}
```

a

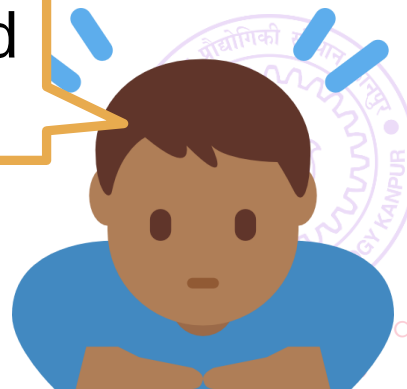
%f



Very large range ± 3.4e+38

Arithmetic operations apply to float as well +, -, /, *, ()

What happened to %?



Float type

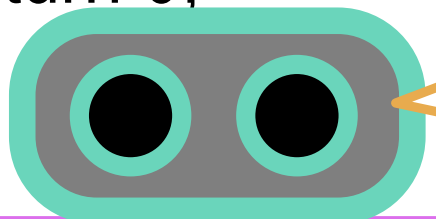
int, long allow us to store, do math formulae with integers

float allows us to store, do math formulae with reals

```
#include <stdio.h>
int main(){
float a;
scanf("%f", &a);
printf("My first real %f", a);
return 0;
}
```

a

%f

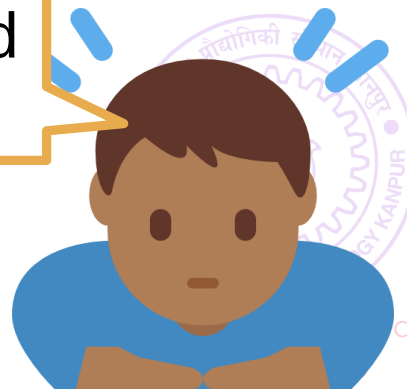


☺ Did you ever do remainders with real numbers in school?

Very large range ± 3.4e+38

Arithmetic operations apply to float as well +, -, /, *, ()

What happened to %?



Float type

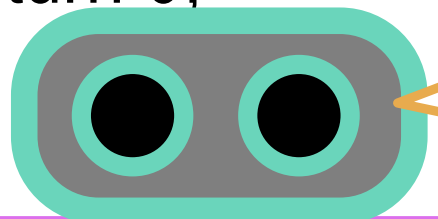
int, long allow us to store, do math formulae with integers

float allows us to store, do math formulae with reals

```
#include <stdio.h>
int main(){
float a;
scanf("%f", &a);
printf("My first real %f", a);
return 0;
}
```

a

%f



☺ Did you ever do remainders with real numbers in school?

Very large range ± 3.4e+38

Arithmetic operations apply to float as well +, -, /, *, ()

Float type

int, long allow us to store, do math formulae with integers

float allows us to store, do math formulae with reals

```
#include <stdio.h>
int main(){
float a;
scanf("%f", &a);
printf("My first real %f", a);
return 0;
}
```

a

%f

☺ Did you ever do remainders with real numbers in school?

Very large range ± 3.4e+38

Arithmetic operations apply to float as well +, -, /, *, ()

I remember. Remainders make sense for integers, not for real numbers

Float type

int, long allow us to store, do math formulae with integers

float allows us to store, do math formulae with reals

```
#include <stdio.h>
int main(){
float a;
scanf("%f", &a);
printf("My first real %f", a);
return 0;
}
```

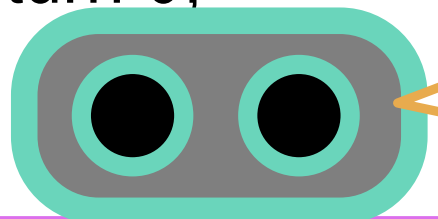
a

%f

Very large range ± 3.4e+38

Arithmetic operations apply to float as well +, -, /, *, ()

Try them out on Prutor



☺ Did you ever do remainders with real numbers in school?

I remember. Remainders make sense for integers, not for real numbers

Ints and Longs

Ints and Longs

Very good friends since both store integers

Ints and Longs

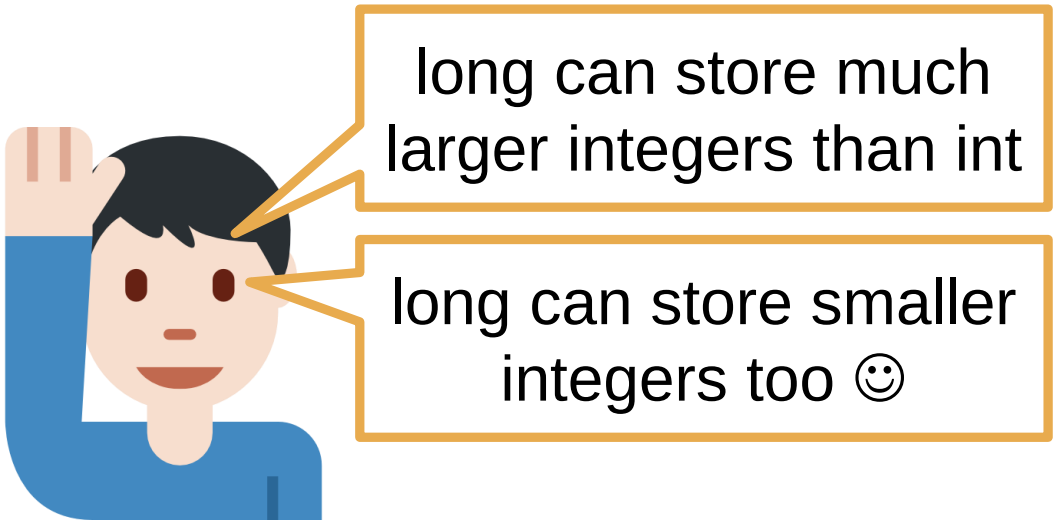
Very good friends since both store integers

Ints and Longs

Very good friends since both store integers

Ints and Longs

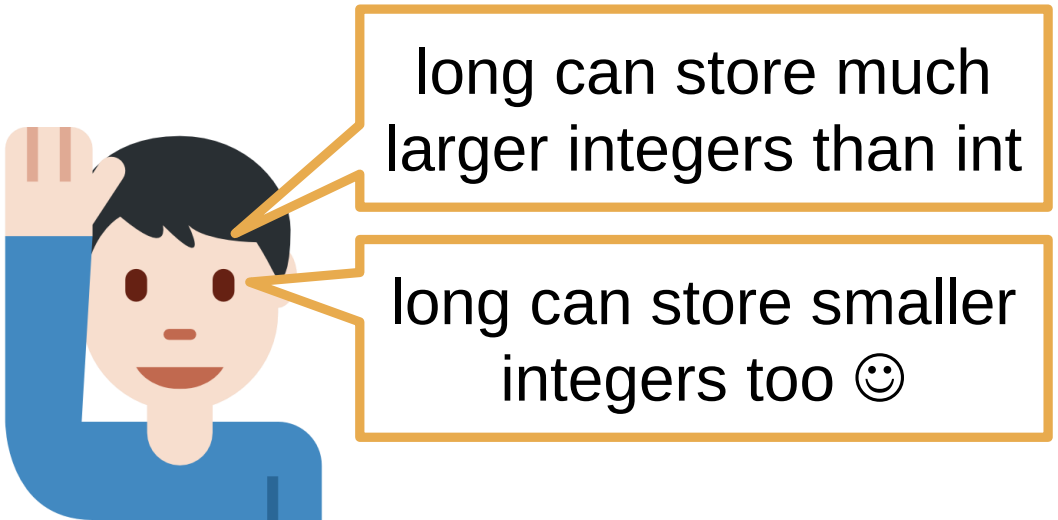
Very good friends since both store integers



Ints and Longs

Very good friends since both store integers

Can add/subtract/multiply/divide/remainder two ints, two longs, as well as an int and a long

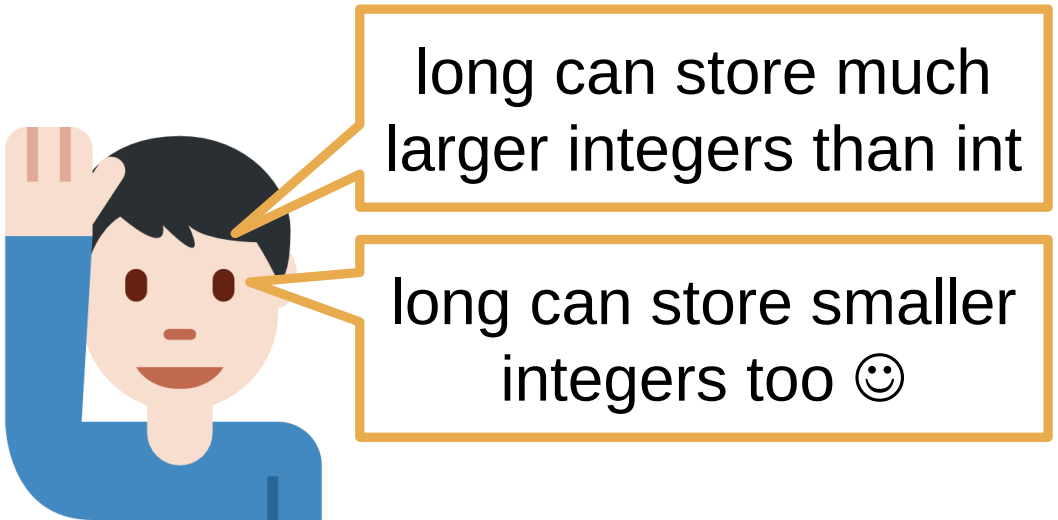


Ints and Longs

Very good friends since both store integers

Can add/subtract/multiply/divide/remainder two ints, two longs, as well as an int and a long

In fact, even if we try to print an int using %ld or print a long using %d, Prutor will only warn us, not throw an error

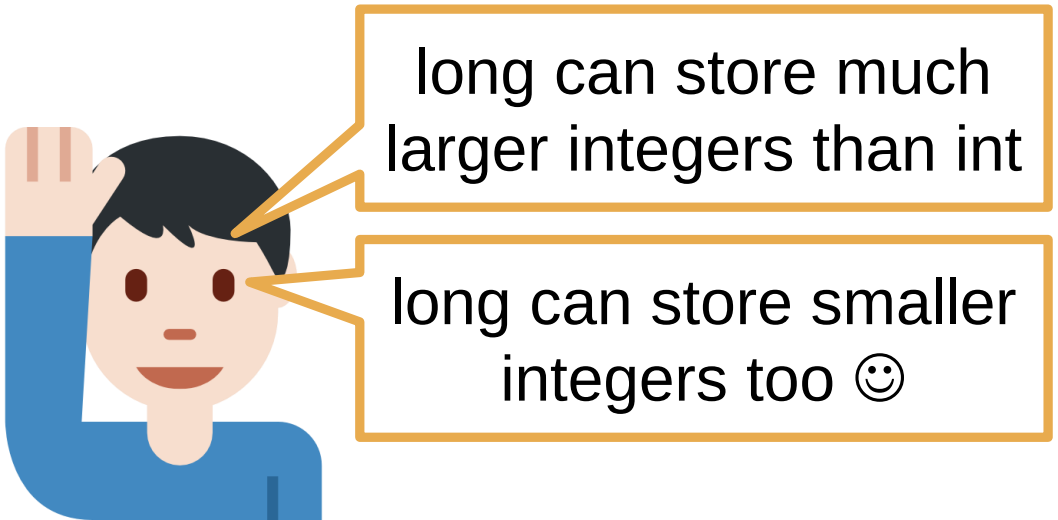


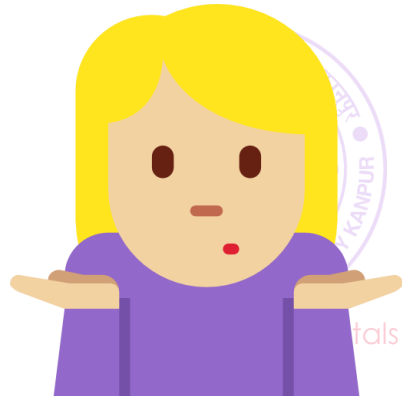
Ints and Longs

Very good friends since both store integers

Can add/subtract/multiply/divide/remainder two ints, two longs, as well as an int and a long

In fact, even if we try to print an int using %ld or print a long using %d, Prutor will only warn us, not throw an error



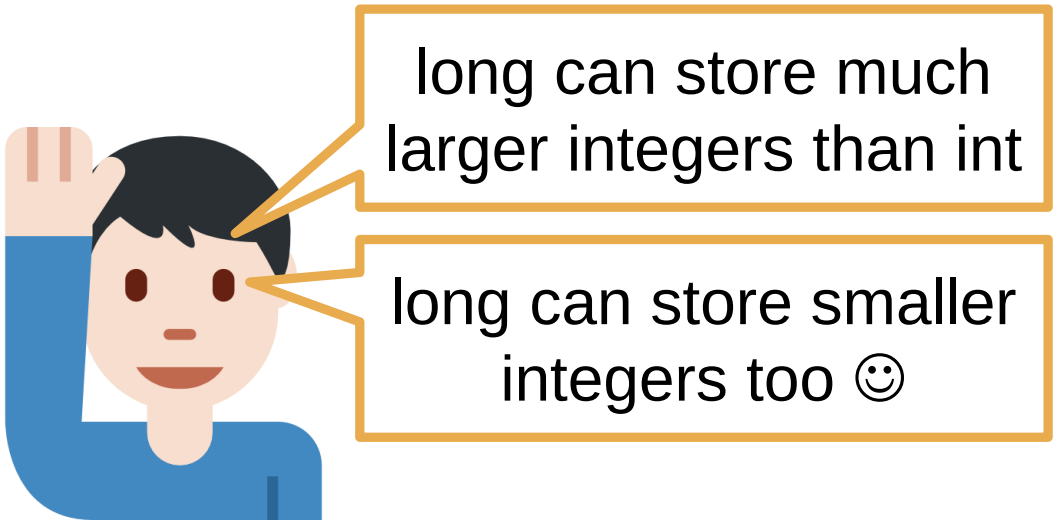


Ints and Longs

Very good friends since both store integers

Can add/subtract/multiply/divide/remainder two ints, two longs, as well as an int and a long

In fact, even if we try to print an int using %ld or print a long using %d, Prutor will only warn us, not throw an error



Ints and Longs

Ints and Longs

Ints and Longs


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

Ints and Longs


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

Ints and Longs


```
#include <stdio.h>
int main(){
    int a = 2000000000;
```

Ints and Longs


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
```

Ints and Longs


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
```

Ints and Longs


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```

Ints and Longs


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```


a

Ints and Longs


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```

```
20000
00000
```

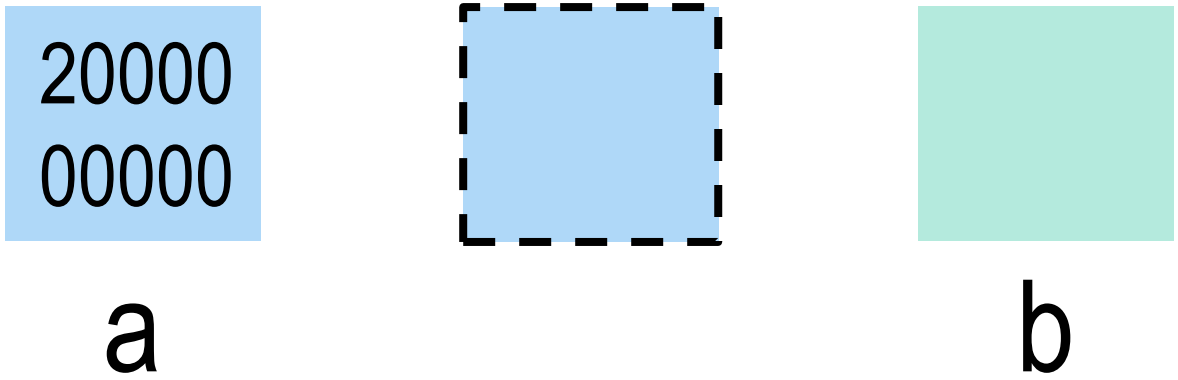
a

Ints and Longs


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```

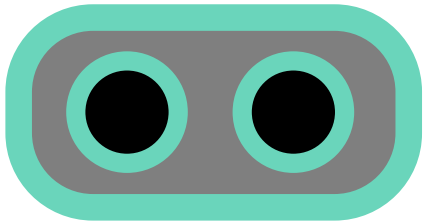
Ints and Longs

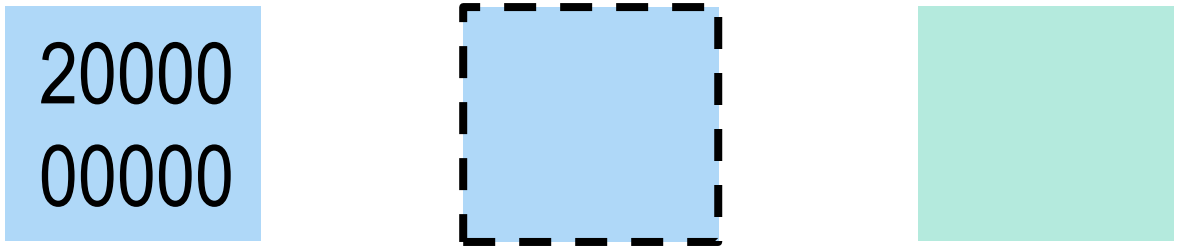

```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```



Ints and Longs


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```

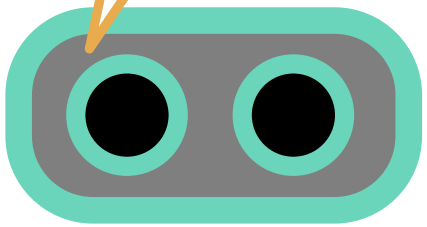


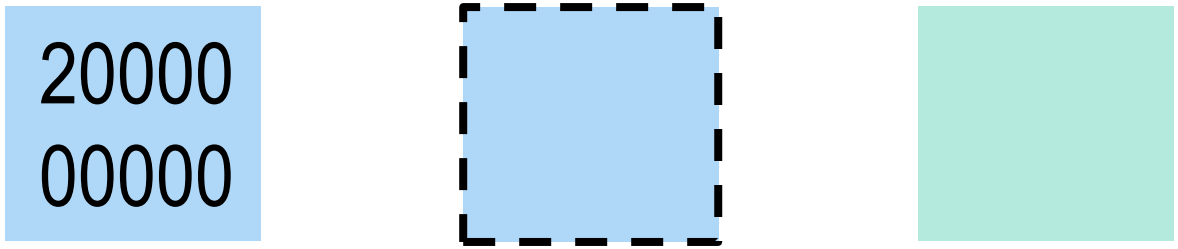


Ints and Longs


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```

Dotted line means I create these variables myself



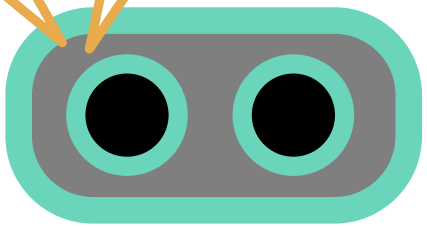


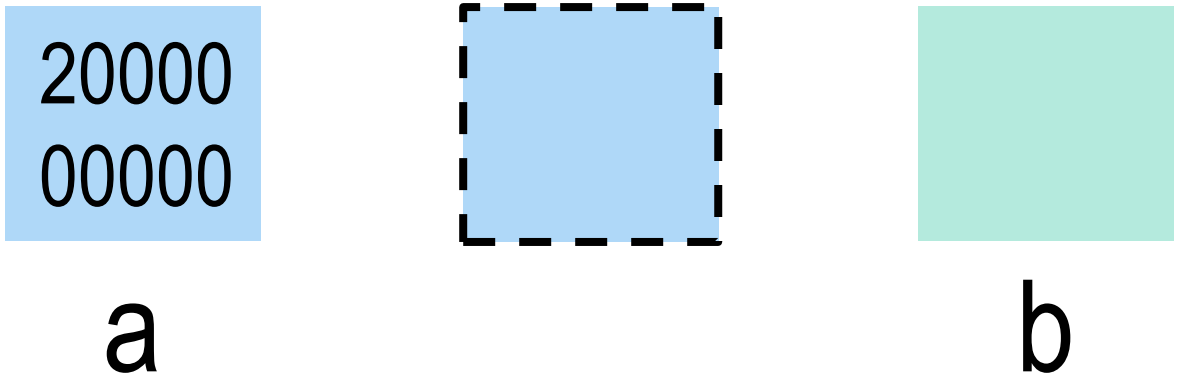
Ints and Longs


```
#include <stdio.h>
int a = 20000 00000;
long b = a + a;
printf("%ld",b);
}
```

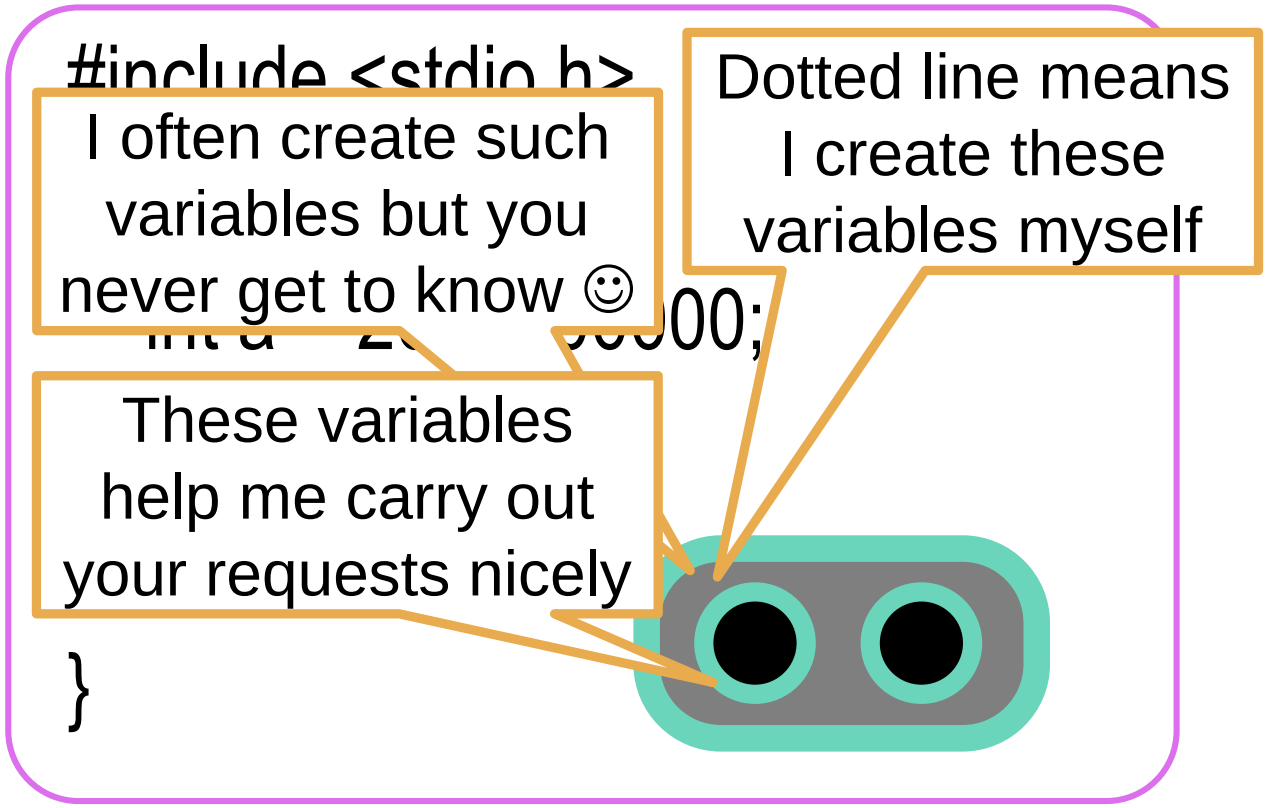
I often create such variables but you never get to know ☺

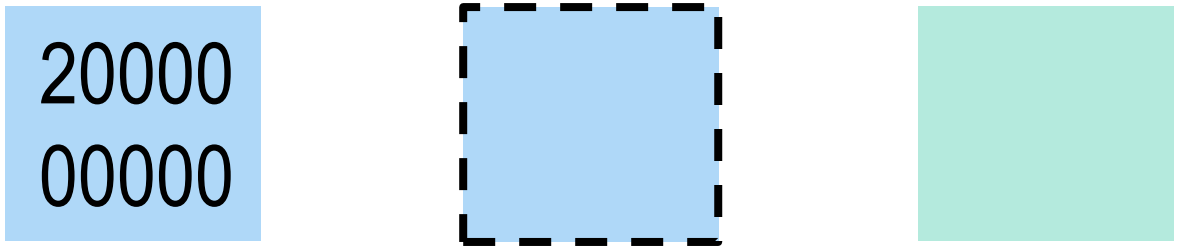
Dotted line means I create these variables myself





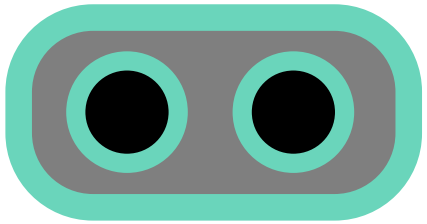
Ints and Longs

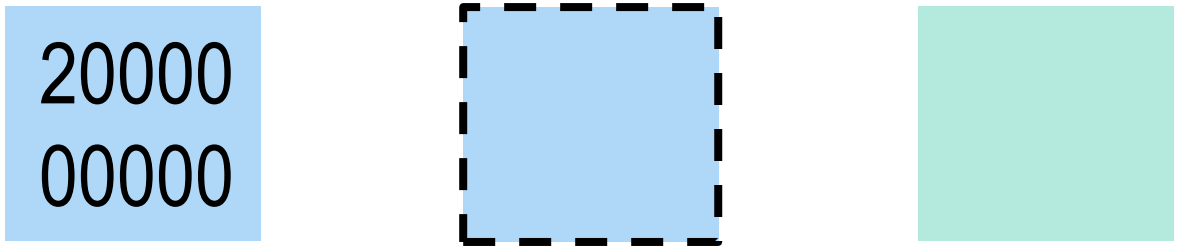




Ints and Longs

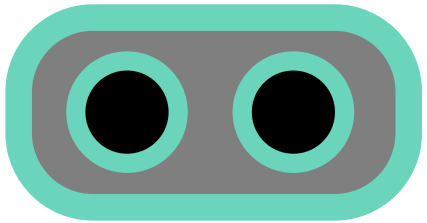

```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```

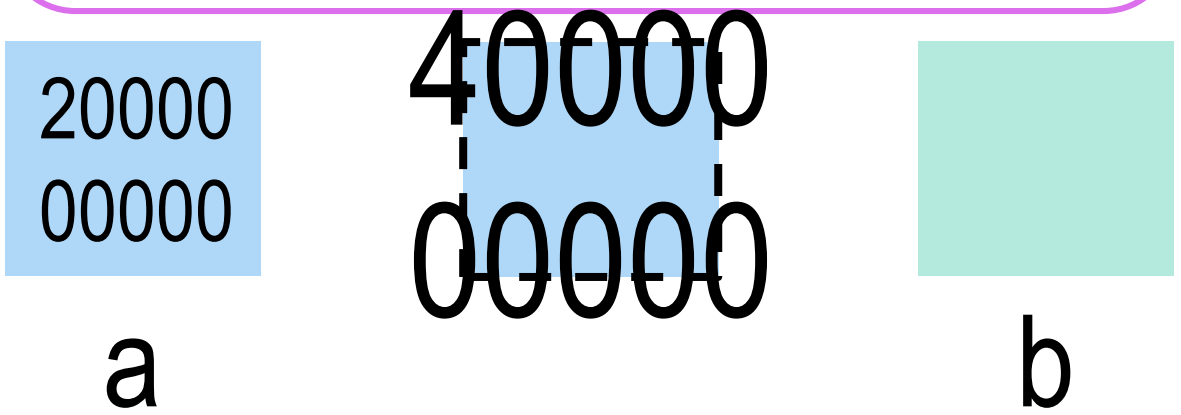




Ints and Longs

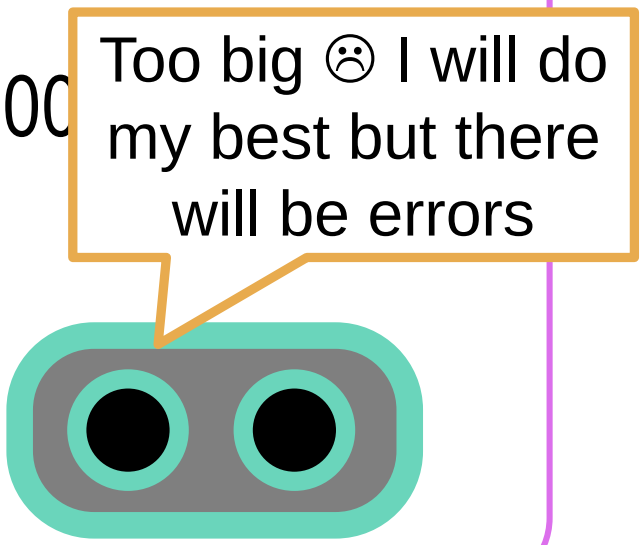

```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```

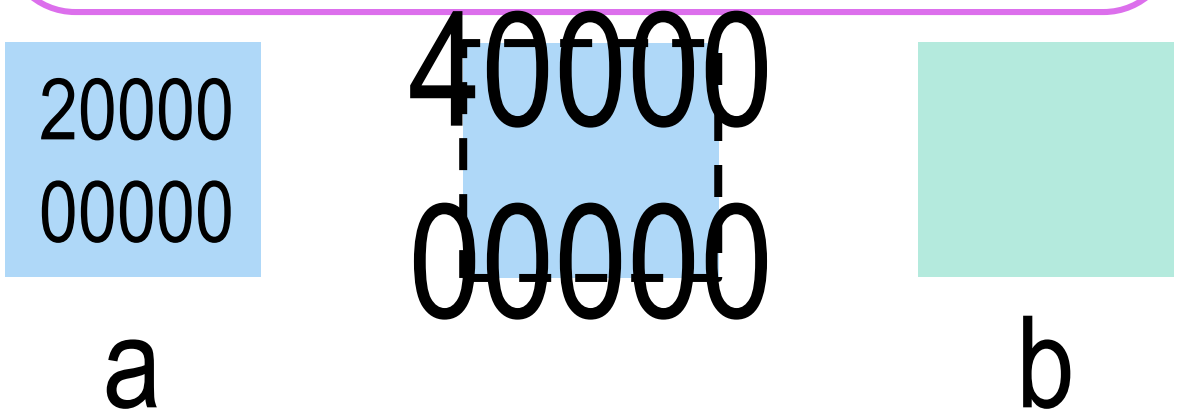




Ints and Longs

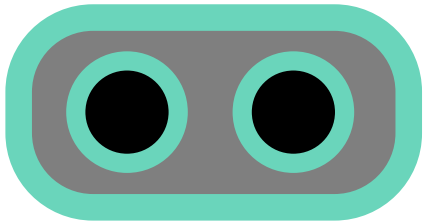

```
#include <stdio.h>
int main(){
    int a = 200000000
    long b = a + a;
    printf("%ld",b);
}
```

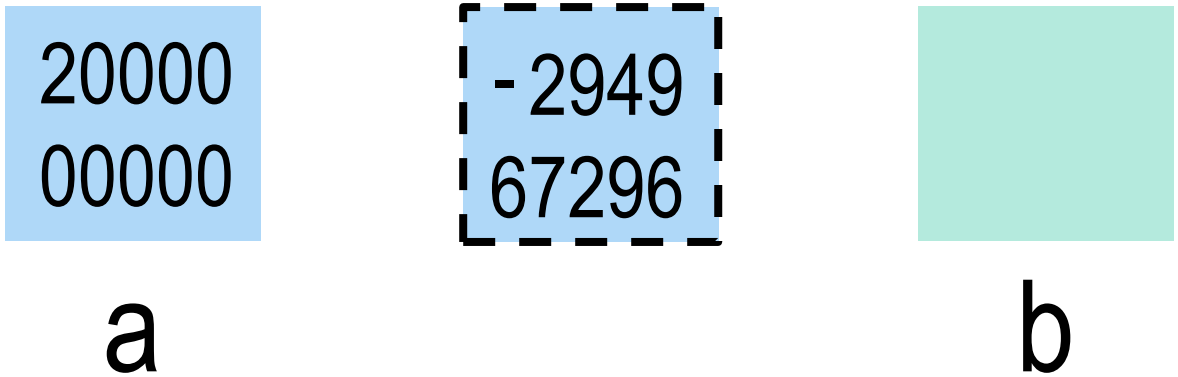




Ints and Longs

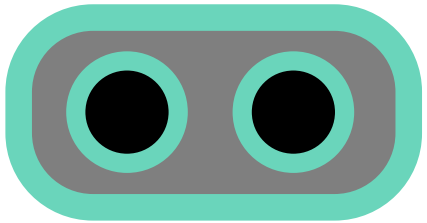

```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```





Ints and Longs

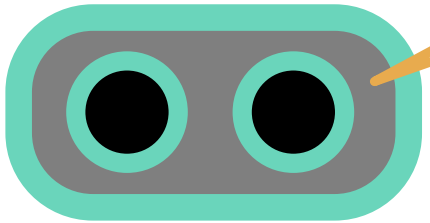

```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```



20000 00000	-2949 67296	-2949 67296
a		b

Ints and Longs


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```

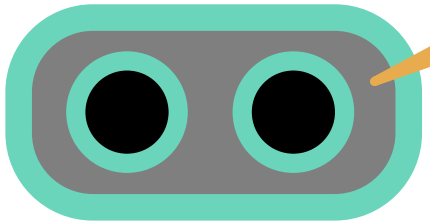


-294967296

20000 00000	-2949 67296	-2949 67296
a		b

Ints and Longs - Typecasting


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```

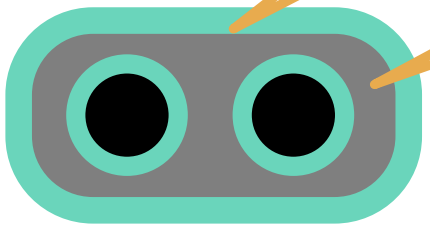


-294967296

20000 00000	-2949 67296	-2949 67296
a		b

Ints and Longs - Typecasting


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```



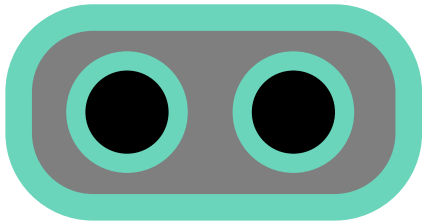
Typecasting is an instruction to me to interpret an int as a long

-294967296

20000 00000	-2949 67296	-2949 67296
a		b

Ints and Longs - Typecasting

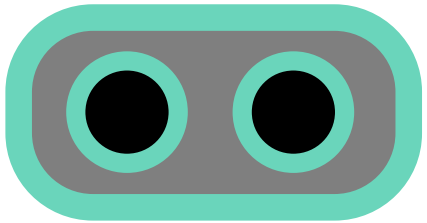

```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```



20000 00000	-2949 67296	-2949 67296
a		b

Ints and Longs - Typecasting


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```



20000
00000

a

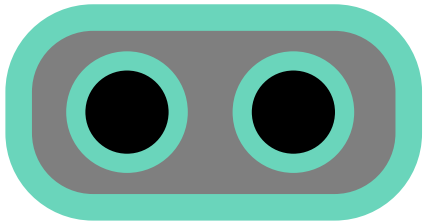
-2949
67296

-2949
67296

b

Ints and Longs - Typecasting


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```

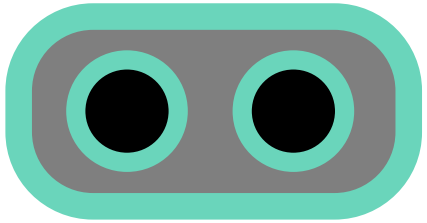


```
#include <stdio.h>
```

20000 00000	-2949 67296	-2949 67296
a		b

Ints and Longs - Typecasting


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```



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

20000
00000

a

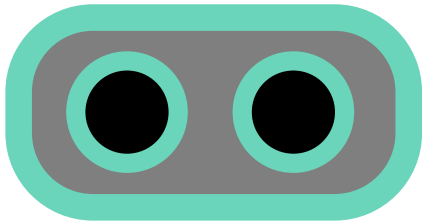
-2949
67296

-2949
67296

b

Ints and Longs - Typecasting


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```



```
#include <stdio.h>
int main(){
    int a = 2000000000;
```

20000
00000

a

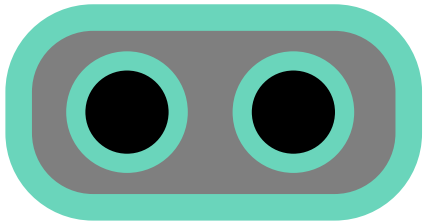
-2949
67296

-2949
67296

b

Ints and Longs - Typecasting


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```

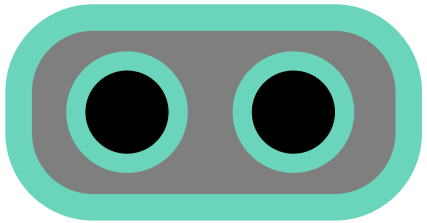


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
```

20000 00000	-2949 67296	-2949 67296
a		b

Ints and Longs - Typecasting


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```



```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
    printf("%ld",b);
```

20000
00000

a

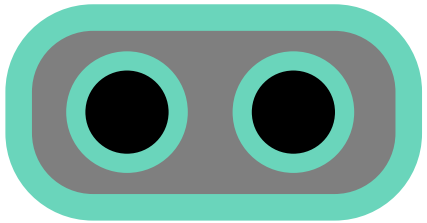
-2949
67296

-2949
67296

b

Ints and Longs - Typecasting


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```

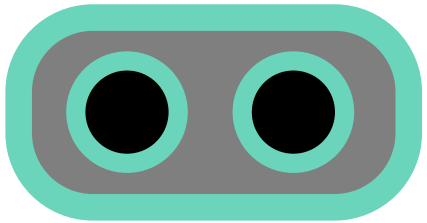


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
    printf("%ld",b);
}
```

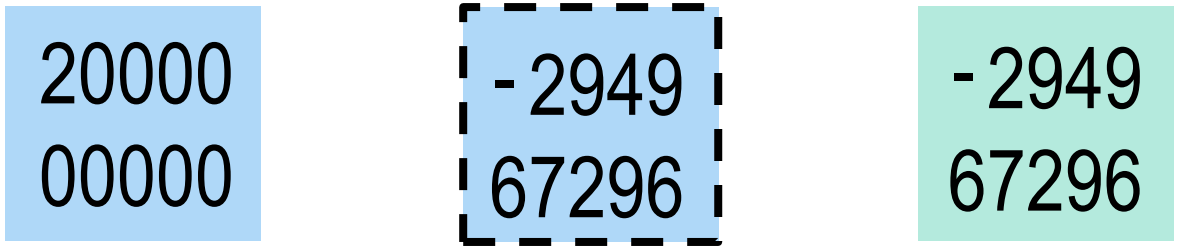
a		b
20000 00000	-2949 67296	-2949 67296

Ints and Longs - Typecasting


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```

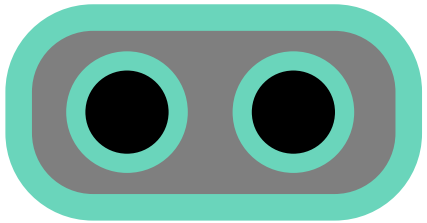


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
    printf("%ld",b);
}
```



Ints and Longs - Typecasting


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```

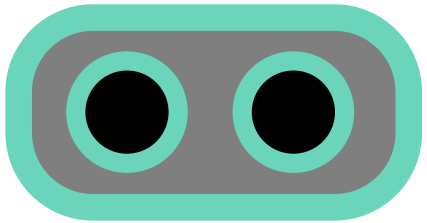


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
    printf("%ld",b);
}
```

20000 00000	-2949 67296	-2949 67296	20000 00000
a		b	a

Ints and Longs - Typecasting


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```



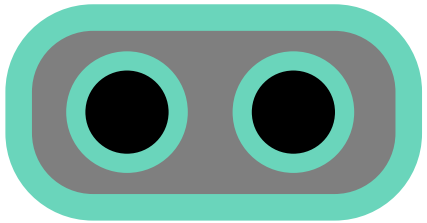
```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
    printf("%ld",b);
}
```

20000 00000	-2949 67296	-2949 67296	20000 00000
a		b	a

b

Ints and Longs - Typecasting


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```



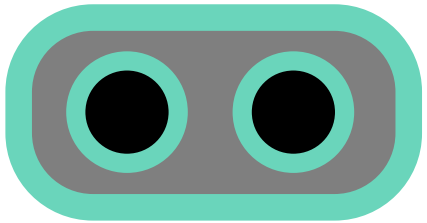
a		b
20000 00000	-2949 67296	-2949 67296

```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
    printf("%ld",b);
}
```

a		b
20000 00000		

Ints and Longs - Typecasting


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```



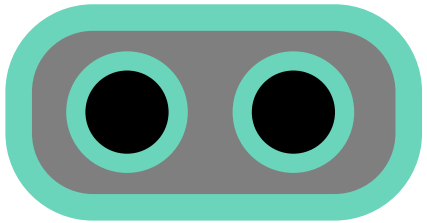
20000 00000	-2949 67296	-2949 67296
a		b

```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
    printf("%ld",b);
}
```

20000 00000	20000 00000
a	b

Ints and Longs - Typecasting


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```



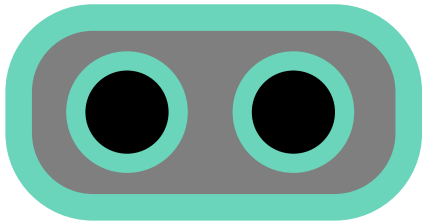
20000 00000	-2949 67296	-2949 67296
a		b

```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
    printf("%ld",b);
}
```

20000 00000	20000 00000	
a		b

Ints and Longs - Typecasting


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```



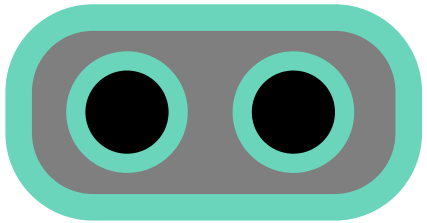
20000 00000	-2949 67296	-2949 67296
a		b

```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
    printf("%ld",b);
}
```

20000 00000	20000 00000	20000 00000
a		b

Ints and Longs - Typecasting


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```



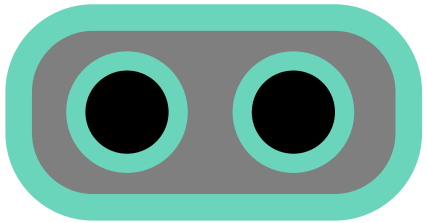
20000 00000	-2949 67296	-2949 67296
a		b

```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
    printf("%ld",b);
}
```

20000 00000	20000 00000	20000 00000
a		b

Ints and Longs - Typecasting


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```



a		b
20000 00000	-2949 67296	-2949 67296

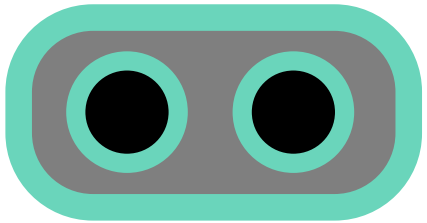
```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
    printf("%ld",b);
}
```

40000
00000

a			b
20000 00000	20000 00000	20000 00000	

Ints and Longs - Typecasting


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```



20000
00000

a

-2949
67296

-2949
67296

b

```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
    printf("%ld",b);
}
```

40000
00000

20000
00000

a

20000
00000

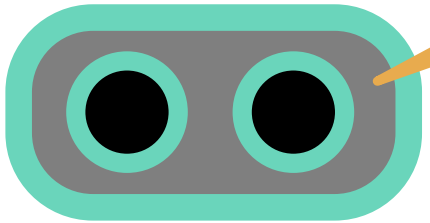
20000
00000

40000
00000

b

Ints and Longs - Typecasting


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```



4000000000

```
#include <stdio.h>
int main(){
    int a = 2000000000;
    b = (long)a + (long)a;
    printf("%ld",b);
}
```

40000
00000

20000
00000

-2949
67296

-2949
67296

20000
00000

20000
00000

20000
00000

40000
00000

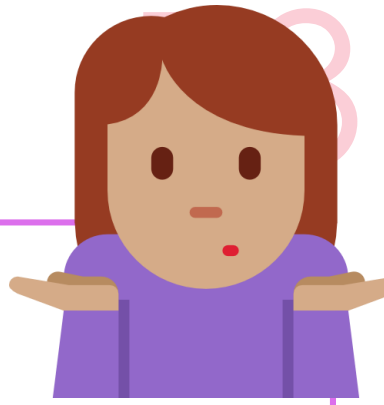
a

b

a

b

Ints and Longs - Typecasting



```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```

4000000000

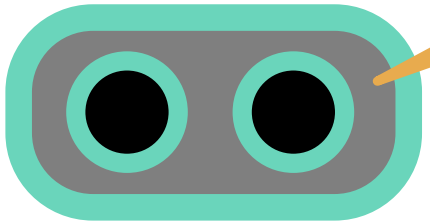
```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
    printf("%ld",b);
}
```

a		b
20000 00000	-2949 67296	-2949 67296

a			b
20000 00000	20000 00000	20000 00000	40000 00000

Ints and Longs - Typec

```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = a + a;
    printf("%ld",b);
}
```



4000000000

20000 00000 — a

-2949 67296

-2949 67296 — b

Why not just define a long variable? No need for typecasting!

```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
    printf("%ld",b);
}
```

40000 00000

20000 00000 — a

20000 00000

20000 00000 — b

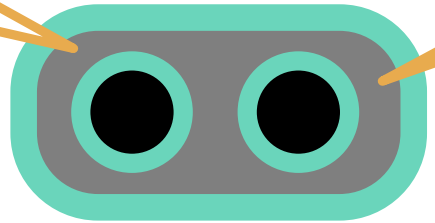
40000 00000

Ints and Longs - Typec

Why not just define a long variable? No need for typecasting!

```
#include <stdio.h>
...00;
    printf("%ld",b);
}
```

Often, you don't have control over the kind of data you receive. Typecasting helps convert data to a form your like to work with



4000000000

```
#include <stdio.h>
int main(){
    int a = 2000000000;
    ... b = (long)a + (long)a;
    printf("%ld",b);
}
```

40000
00000

20000 00000	-2949 67296	-2949 67296	20000 00000	20000 00000	20000 00000	40000 00000
a		b	a			b

A handy shorthand

A handy shorthand

A handy shorthand


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

A handy shorthand


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

A handy shorthand


```
#include <stdio.h>
int main(){
    int a = 2000000000;
```

A handy shorthand


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
```

A handy shorthand


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
    printf("%ld",b);
```

A handy shorthand


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
    printf("%ld",b);
}
```

A handy shorthand


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
    printf("%ld",b);
}
```

A handy shorthand


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
    printf("%ld",b);
}
```

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

A handy shorthand


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
    printf("%ld",b);
}
```

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

A handy shorthand


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
    printf("%ld",b);
}
```

```
#include <stdio.h>
int main(){
    int a = 2000000000;
```

A handy shorthand


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
    printf("%ld",b);
}
```

```
#include <stdio.h>
int main(){
    int a = 2000000000;
    printf("%ld", (long)a + (long)a);
```

A handy shorthand

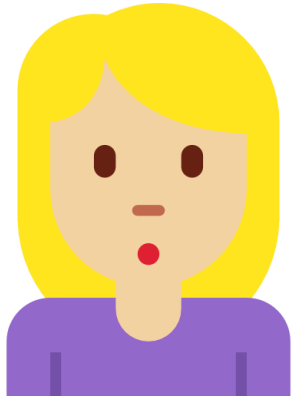

```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
    printf("%ld",b);
}
```

```
#include <stdio.h>
int main(){
    int a = 2000000000;
    printf("%ld", (long)a + (long)a);
}
```

A handy shorthand


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
    printf("%ld",b);
}
```

```
#include <stdio.h>
int main(){
    int a = 2000000000;
    printf("%ld", (long)a + (long)a);
}
```



A handy shorthand

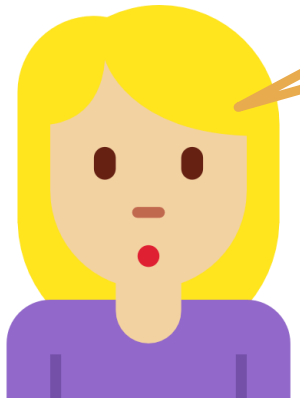

```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
    printf("%ld",b);
}
```

```
#include <stdio.h>
int main(){
    int a = 2000000000;
    printf("%ld", (long)a + (long)a);
}
```

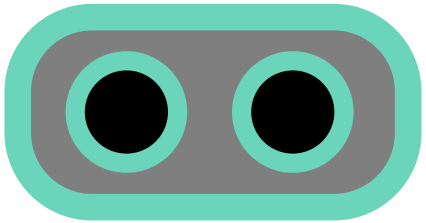
A handy shorthand


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
    printf("%ld",b);
}
```

That means I really have to be careful



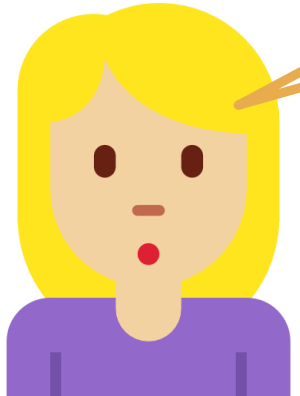
```
#include <stdio.h>
int main(){
    int a = 2000000000;
    printf("%ld", (long)a + (long)a);
}
```



A handy shorthand

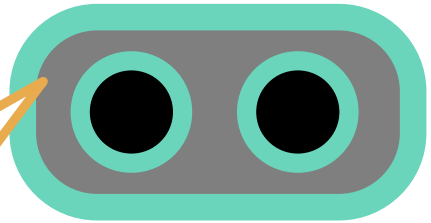

```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
    printf("%ld",b);
}
```

That means I really have to be careful



```
#include <stdio.h>
int main(){
    int a = 2000000000;
    printf("%ld", (long)a + (long)a);
}
```

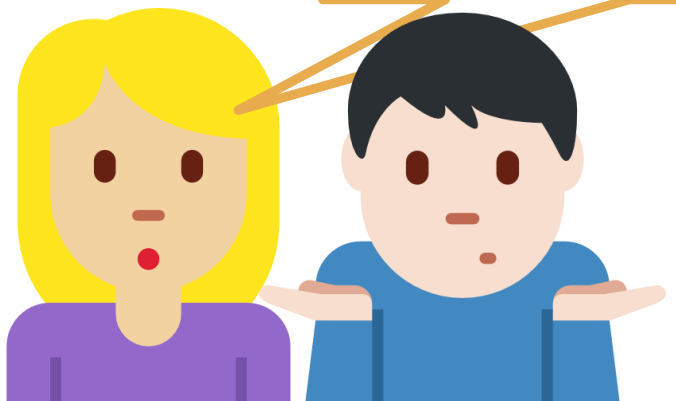
Yes, when in doubt, try typecasting to see if error vanishes



A handy shorthand

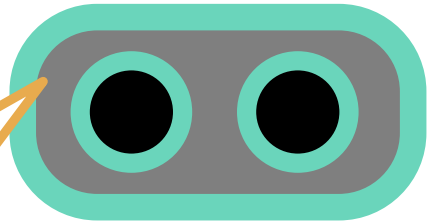

```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
    printf("%ld",b);
}
```

That means I really have to be careful



```
#include <stdio.h>
int main(){
    int a = 2000000000;
    printf("%ld", (long)a + (long)a);
}
```

Yes, when in doubt, try typecasting to see if error vanishes

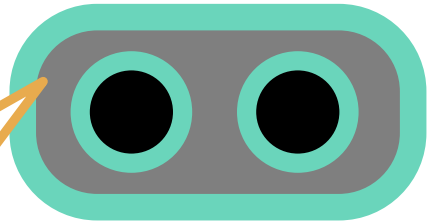


A handy shorthand

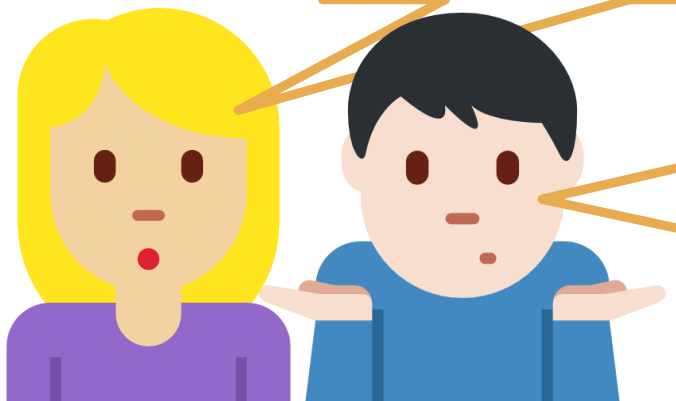

```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
    printf("%ld",b);
}
```

That means I really have to be careful

```
#include <stdio.h>
int main(){
    int a = 2000000000;
    printf("%ld", (long)a + (long)a);
}
```



Yes, when in doubt, try typecasting to see if error vanishes

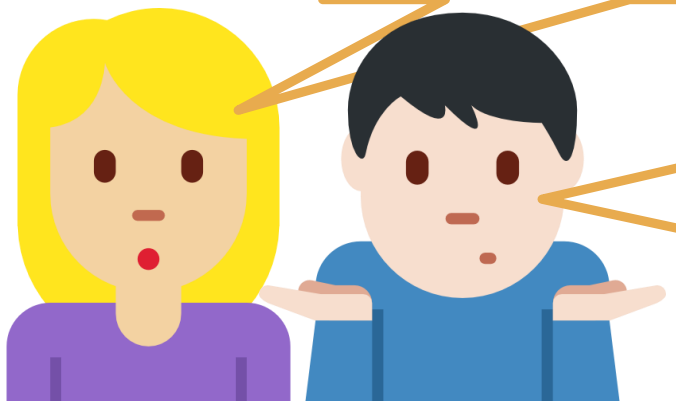


But Mr. C, why didn't you tell us about these errors when we compiled?

A handy shorthand

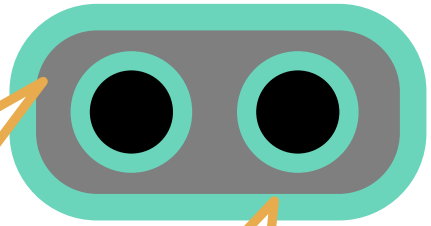

```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
    printf("%ld",b);
}
```

```
#include <stdio.h>
int main(){
    int a = 2000000000;
    printf("%ld", (long)a + (long)a);
}
```



That means I really have to be careful

But Mr. C, why didn't you tell us about these errors when we compiled?



Yes, when in doubt, try typecasting to see if error vanishes

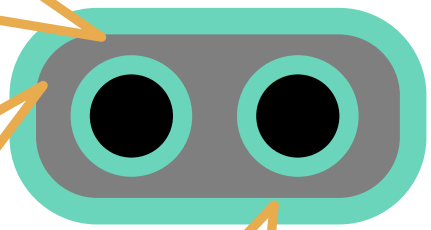
During compilation I only check if your grammar is correct

A handy shorthand


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    long b = (long)a + (long)a;
    printf("%ld",b);
}
```

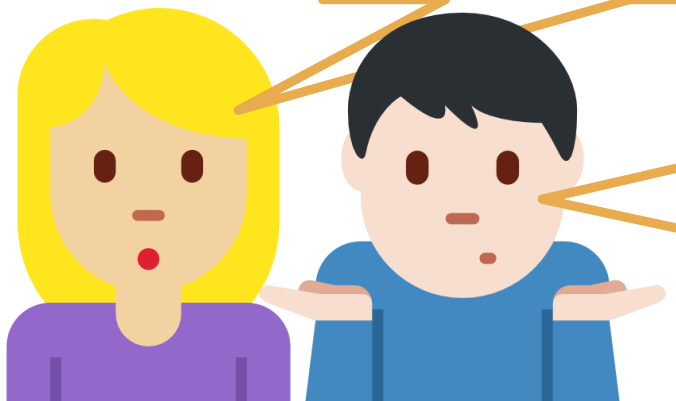
```
#include <stdio.h>
int main(){
                                00;
                               )a + (long)a);
```

The errors we just saw are called *type errors*. These are a kind of *runtime error*



Yes, when in doubt, try typecasting to see if error vanishes

That means I really have to be careful



But Mr. C, why didn't you tell us about these errors when we compiled?

During compilation I only check if your grammar is correct

A handy shorthand


```
#include <stdio.h>
int main(){
    int a = 2000000000;
    lo                  ;
    p
}
```

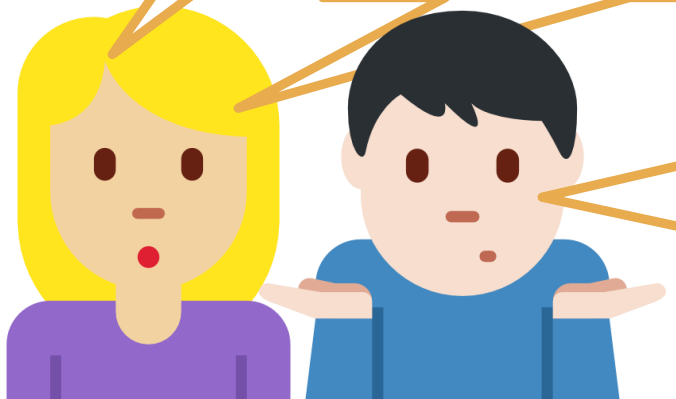
```
#include <stdio.h>
int main(){
                        00;
                        )a + (long)a);
```

Hmm. We see runtime errors only when we execute our code

That means I really have to be careful

The errors we just saw are called *type errors*. These are a kind of *runtime error*

Yes, when in doubt, try typecasting to see if error vanishes



But Mr. C, why didn't you tell us about these errors when we compiled?

During compilation I only check if your grammar is correct

Mixed type operations

Mixed type operations


```
int a = 2;
long c, b = 5;
```

Mixed type operations

What if we have mixed types in a formula?

```
int a = 2;
long c, b = 5;
```

Mixed type operations

What if we have mixed types in a formula?

```
c = a * b;
```

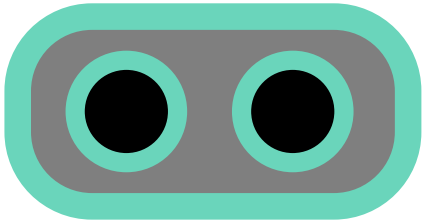
```
int a = 2;
long c, b = 5;
```

Mixed type operations

What if we have mixed types in a formula?

```
c = a * b;
```

```
int a = 2;
long c, b = 5;
```



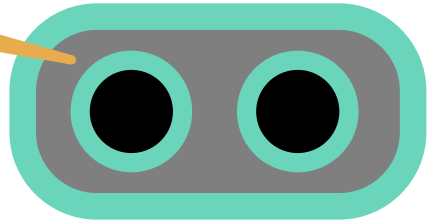
Mixed type operations

What if we ha

```
c = a * b;
```

Hmm … An int being multiplied to a long. Let me take care to convert the int to a long before performing the operation ☺

```
int a = 2;
long c, b = 5;
```



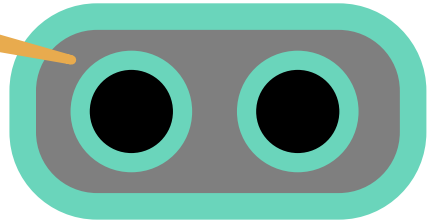
Mixed type operations

What if we ha

```
c = a * b;
```

Hmm … An int being multiplied to a long. Let me take care to convert the int to a long before performing the operation ☺

```
int a = 2;
long c, b = 5;
```



2

a

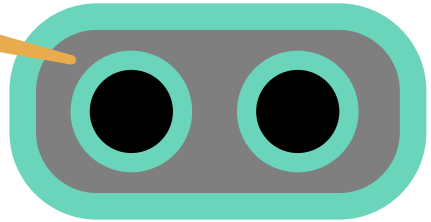
Mixed type operations

What if we ha

```
c = a * b;
```

Hmm … An int being multiplied to a long. Let me take care to convert the int to a long before performing the operation ☺

```
int a = 2;
long c, b = 5;
```



2

a

c

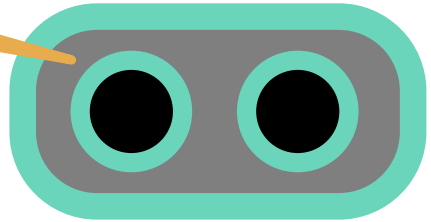
Mixed type operations

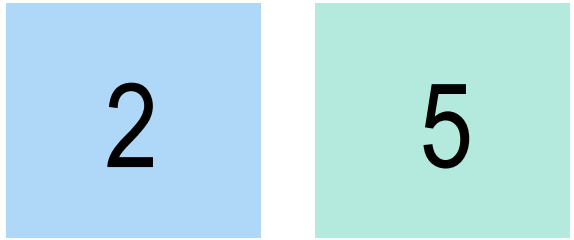
What if we ha

```
c = a * b;
```

Hmm … An int being multiplied to a long. Let me take care to convert the int to a long before performing the operation ☺

```
int a = 2;
long c, b = 5;
```





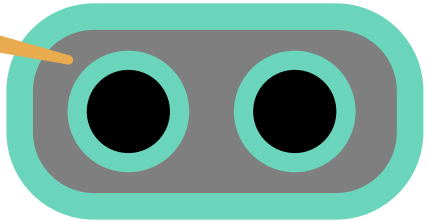
Mixed type operations

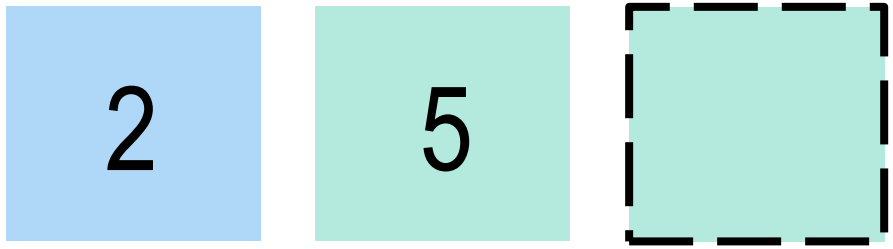
What if we ha

```
c = a * b;
```

Hmm … An int being multiplied to a long. Let me take care to convert the int to a long before performing the operation ☺

```
int a = 2;
long c, b = 5;
```





c

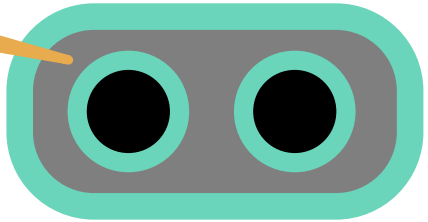
Mixed type operations

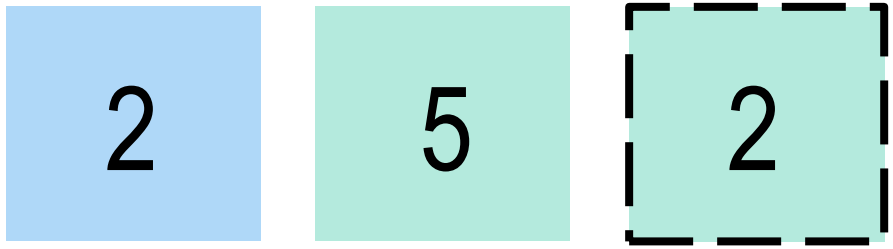
What if we ha

```
c = a * b;
```

Hmm … An int being multiplied to a long. Let me take care to convert the int to a long before performing the operation ☺

```
int a = 2;
long c, b = 5;
```





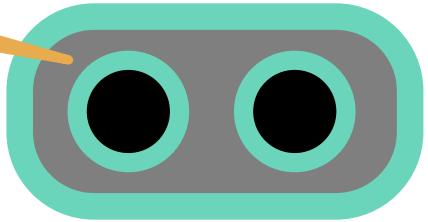
Mixed type operations

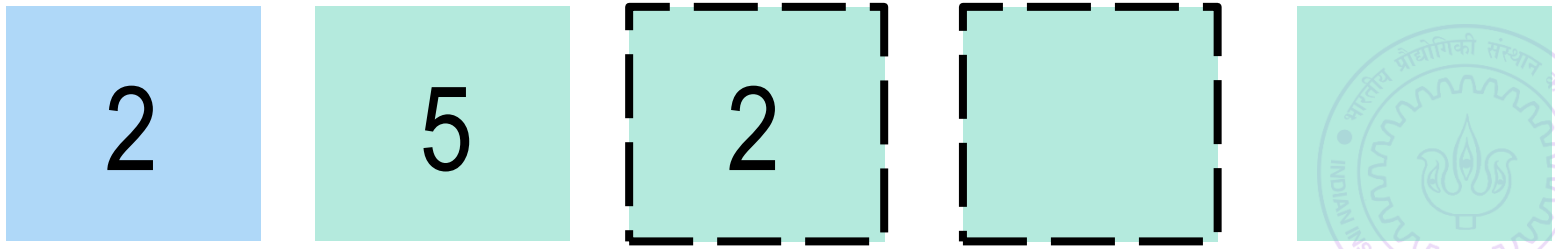
What if we ha

```
c = a * b;
```

Hmm … An int being multiplied to a long. Let me take care to convert the int to a long before performing the operation ☺

```
int a = 2;
long c, b = 5;
```





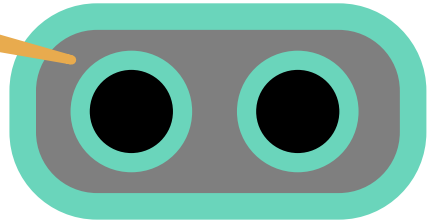
Mixed type operations

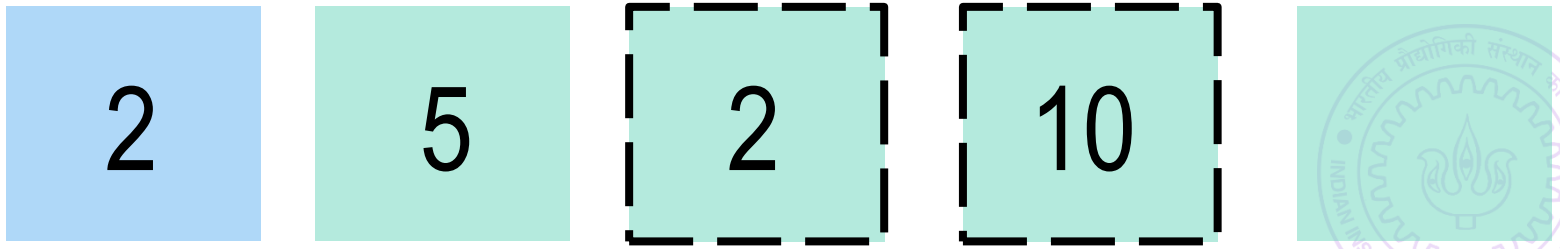
What if we ha

```
c = a * b;
```

Hmm … An int being multiplied to a long. Let me take care to convert the int to a long before performing the operation ☺

```
int a = 2;
long c, b = 5;
```





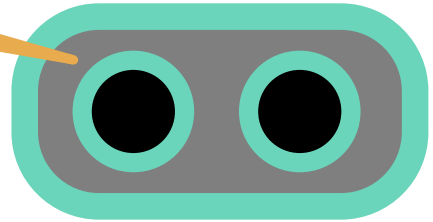
Mixed type operations

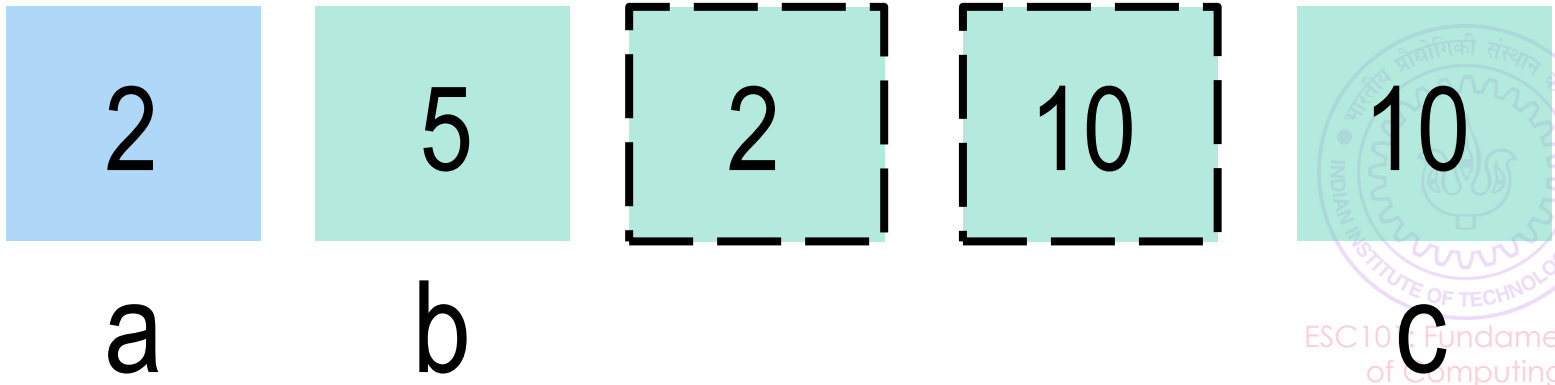
What if we ha

```
c = a * b;
```

Hmm … An int being multiplied to a long. Let me take care to convert the int to a long before performing the operation ☺

```
int a = 2;
long c, b = 5;
```





Mixed type operations

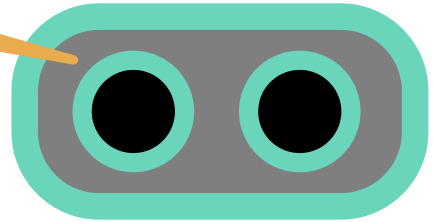
What if we ha

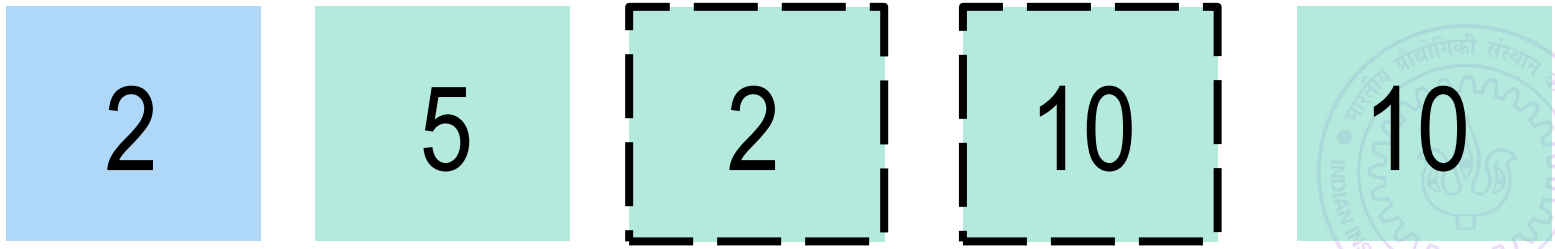
```
c = a * b;
```

Can typecast int to long

Hmm … An int being multiplied to a long. Let me take care to convert the int to a long before performing the operation ☺

```
int a = 2;
long c, b = 5;
```





Mixed type operations

What if we ha

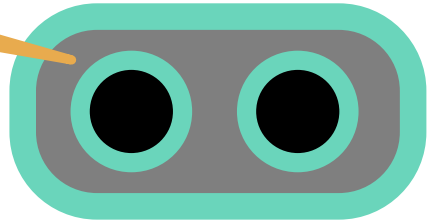
```
c = a * b;
```

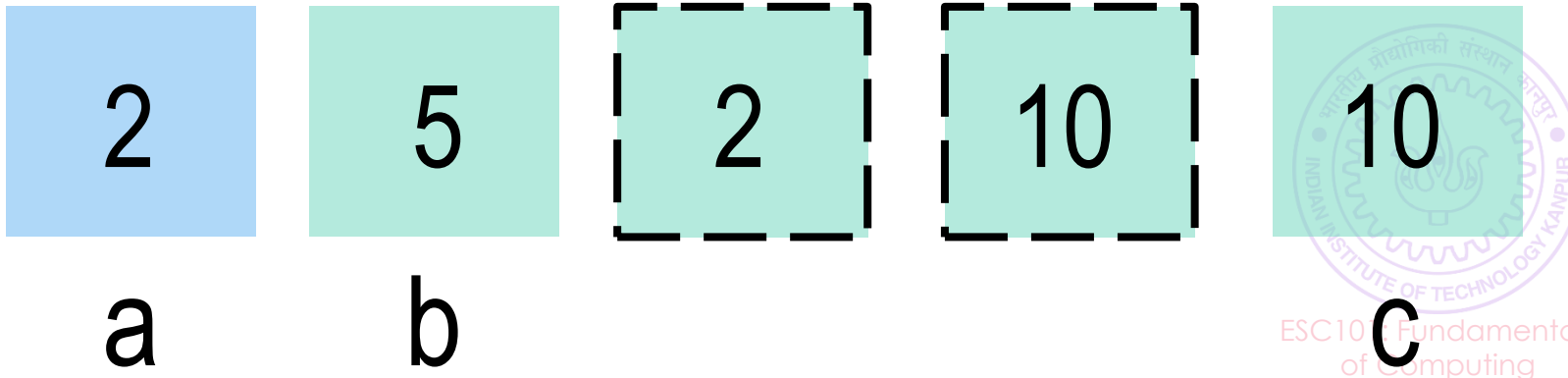
Can typecast int to long

```
b = (long) a;
```

Hmm … An int being multiplied to a long. Let me take care to convert the int to a long before performing the operation ☺

```
int a = 2;
long c, b = 5;
```





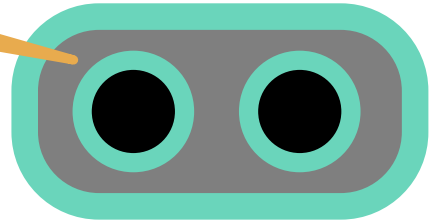
Mixed type operations

What if we ha

```
c = a * b;
```

Hmm … An int being multiplied to a long. Let me take care to convert the int to a long before performing the operation ☺

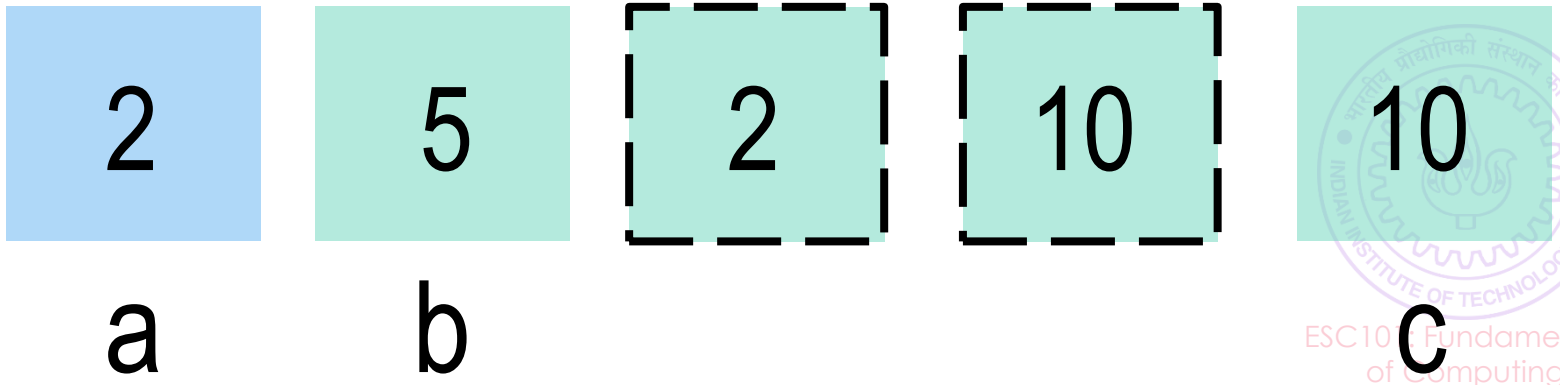
```
int a = 2;
long c, b = 5;
```



Can typecast int to long

```
b = (long) a;
```

Can typecast long to int



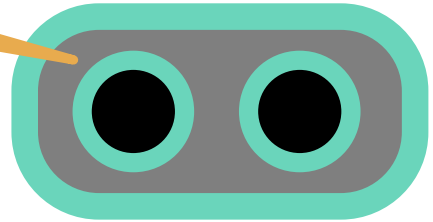
Mixed type operations

What if we ha

```
c = a * b;
```

Hmm … An int being multiplied to a long. Let me take care to convert the int to a long before performing the operation ☺

```
int a = 2;
long c, b = 5;
```

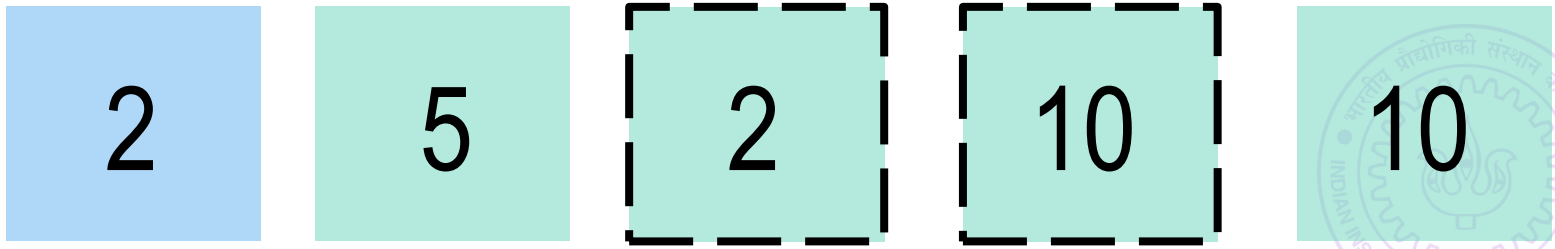


Can typecast int to long

```
b = (long) a;
```

Can typecast long to int

```
a = (int) b;
```



Mixed type operations

What if we ha

```
c = a * b;
```

Hmm … An int being multiplied to a long. Let me take care to convert the int to a long before performing the operation ☺

```
int a = 2;
long c, b = 5;
```

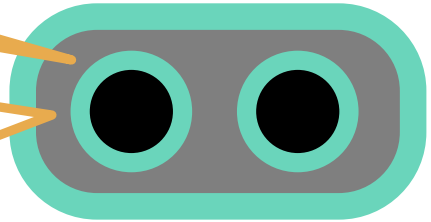
Can typecast int to long

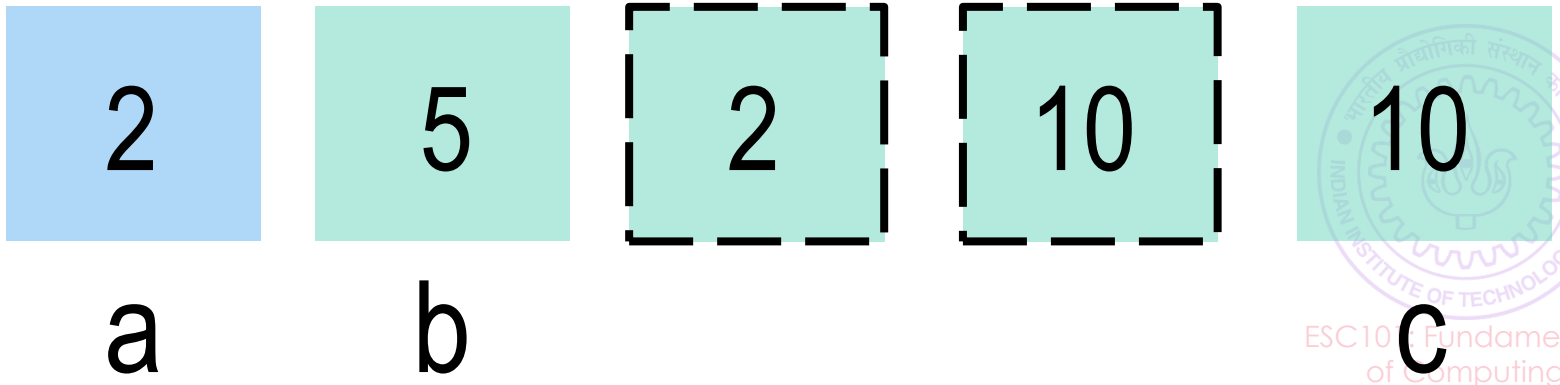
```
b = (long) a;
```

Can typecast long to int

```
a = (int) b;
```

Be careful! If b was storing a very large integer that won't fit into int, this typecast will cause errors





Mixed type operations

What if we ha

```
c = a * b;
```

Hmm … An int being multiplied to a long. Let me take care to convert the int to a long before performing the operation ☺

```
int a = 2;
long c, b = 5;
```

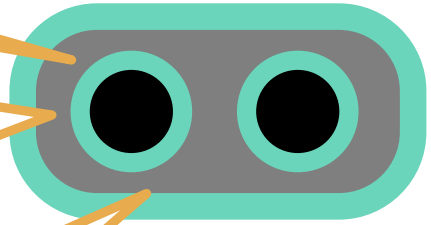
Can typecast int to long

```
b = (long) a;
```

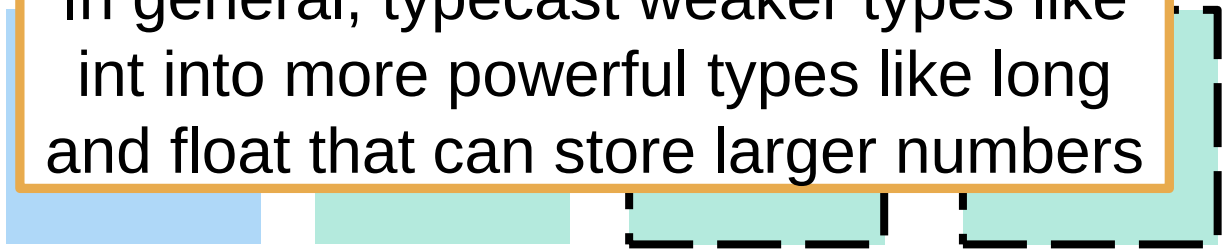
Can typecast long to int

```
a = (int) b;
```

Be careful! If b was storing a very large integer that won't fit into int, this typecast will cause errors



In general, typecast weaker types like int into more powerful types like long and float that can store larger numbers



Mixed type operations

What if we ha

```
c = a * b;
```

Hmm … An int being multiplied to a long. Let me take care to convert the int to a long before performing the operation ☺

```
int a = 2;
long c, b = 5;
```

Can typecast int to long

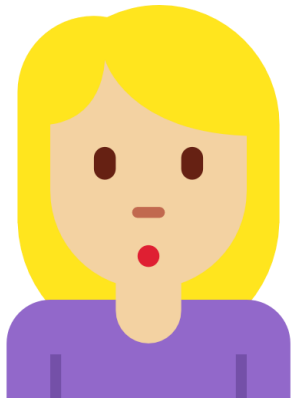
```
b = (long) a;
```

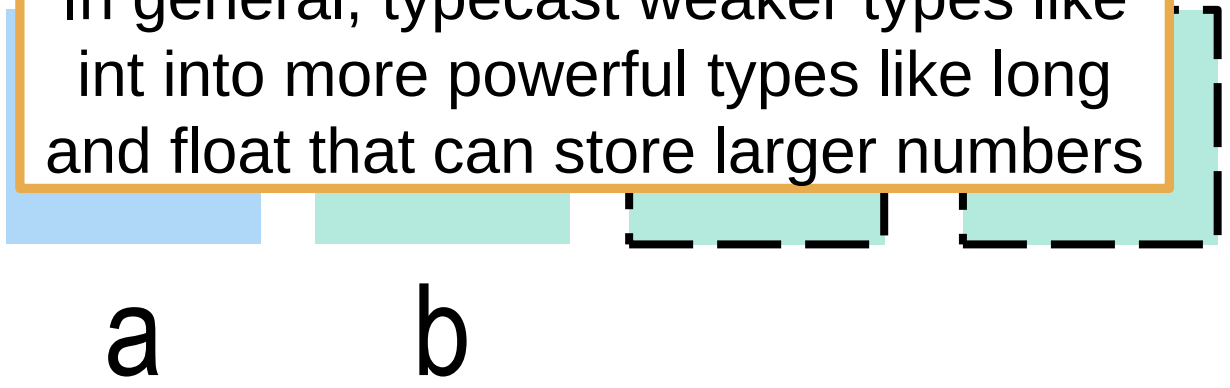
Can typecast long to int

```
a = (int) b;
```

Be careful! If b was storing a very large integer that won't fit into int, this typecast will cause errors

In general, typecast weaker types like int into more powerful types like long and float that can store larger numbers





Mixed type operations

What if we ha

```
c = a * b;
```

Hmm … An int being multiplied to a long. Let me take care to convert the int to a long before performing the operation ☺

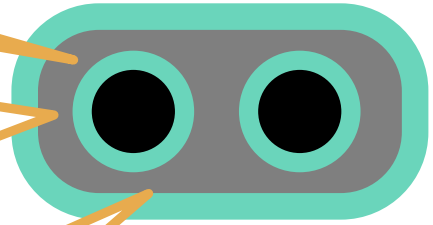
```
int a = 2;
long c, b = 5;
```

Can typecast int to long

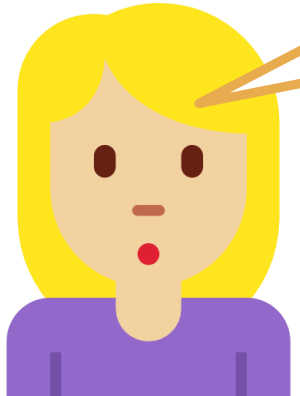
```
b = (long) a;
```

Can typecast long to int

```
a = (int) b;
```

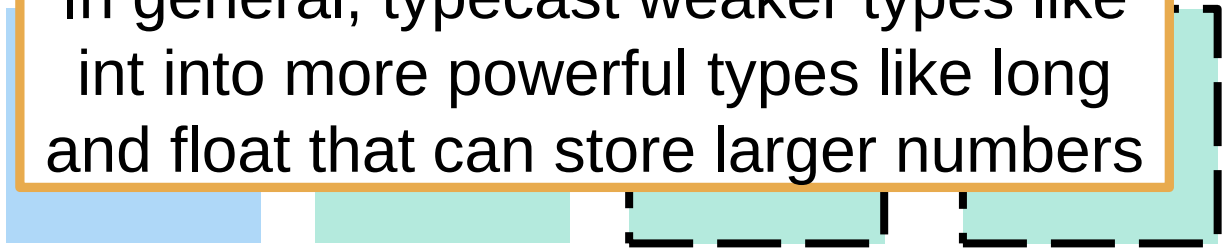


Be careful! If b was storing a very large integer that won't fit into int, this typecast will cause errors



We can typecast int to float too?

In general, typecast weaker types like int into more powerful types like long and float that can store larger numbers



a b

c

Mixed type operations

What if we ha

```
c = a * b;
```

Hmm … An int being multiplied to a long. Let me take care to convert the int to a long before performing the operation ☺

```
int a = 2;
long c, b = 5;
```

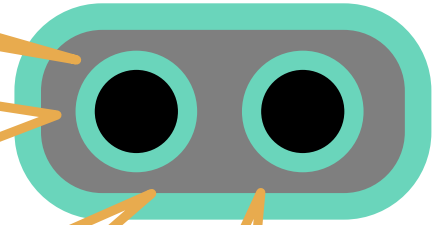
Can typecast int to long

```
b = (long) a;
```

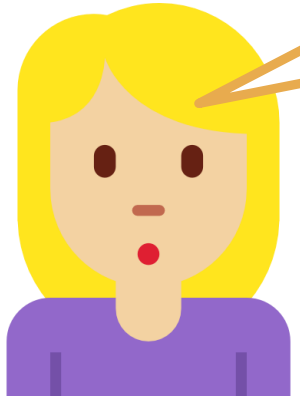
Can typecast long to int

```
a = (int) b;
```

Be careful! If b was storing a very large integer that won't fit into int, this typecast will cause errors

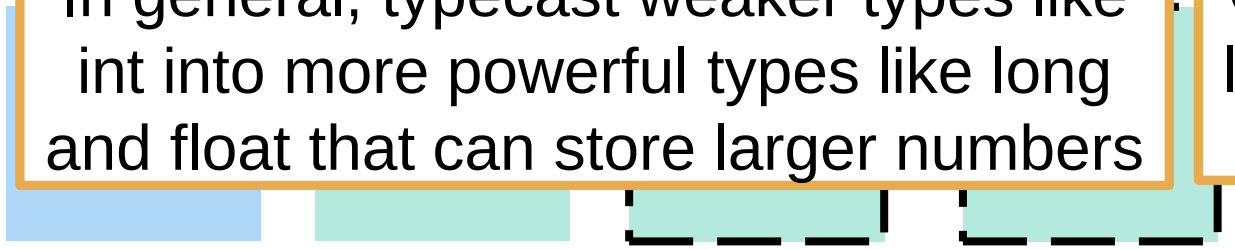


We can typecast int to float too?



In general, typecast weaker types like int into more powerful types like long and float that can store larger numbers

Of course – let me show you ☺



Is 2.0 the same as 2? (JEE 2018)

Is 2.0 the same as 2? (JEE 2018)


```
#include <stdio.h>
int main(){
    float a = 2.5;
    printf("%f", a);
    return 0;
}
```

Is 2.0 the same as 2? (JEE 2018)


```
#include <stdio.h>
int main(){
    float a = 2.5;
    printf("%f", a);
    return 0;
}
```

```
#include <stdio.h>
int main(){
    float a = 2.5;
    printf("%d", a);
    return 0;
}
```

Is 2.0 the same as 2? (JEE 2018)


```
#include <stdio.h>
int main(){
    float a = 2.5;
    printf("%f", a);
    return 0;
}
```

```
#include <stdio.h>
int main(){
    float a = 2.5;
    printf("%d", (int) a);
    return 0;
}
```

Is 2.0 the same as 2? (JEE 2018)


```
#include <stdio.h>
int main(){
    float a = 2.5;
    printf("%f", a);
    return 0;
}
```

```
#include <stdio.h>
int main(){
    float a = 2.5;
    printf("%d", (int) a);
    return 0;
}
```

```
float c = 2/3;
```

Is 2.0 the same as 2? (JEE 2018)


```
#include <stdio.h>
int main(){
    float a = 2.5;
    printf("%f", a);
    return 0;
}
```

```
#include <stdio.h>
int main(){
    float a = 2.5;
    printf("%d", (int) a);
    return 0;
}
```

float c = 2/3;

float c = 2/3.0;

Is 2.0 the same as 2? (JEE 2018)


```
#include <stdio.h>
int main(){
    float a = 2.5;
    printf("%f", a);
    return 0;
}
```

```
#include <stdio.h>
int main(){
    float a = 2.5;
    printf("%d", (int) a);
    return 0;
}
```

```
float c = 2/3;
```

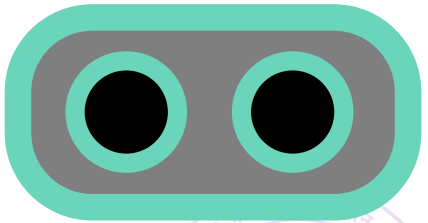
```
float c = 2/3.0;
```

```
float c = 2/(float)3;
```

Is 2.0 the same as 2? (JEE 2018)


```
#include <stdio.h>
int main(){
    float a = 2.5;
    printf("%f", a);
    return 0;
}
```

```
#include <stdio.h>
int main(){
    float a = 2.5;
    printf("%d", (int) a);
    return 0;
}
```



```
float c = 2/3;
```

```
float c = 2/3.0;
```

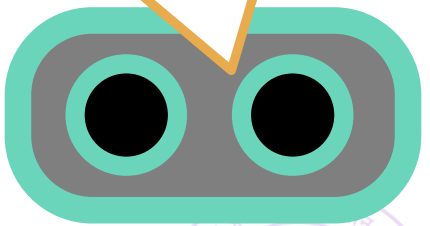
```
float c = 2/(float)3;
```

Is 2.0 the same as 2? (JEE 2018)


```
#include <stdio.h>
int main(){
    float a = 2.5;
    printf("%f", a);
    return 0;
}
```

```
#include <stdio.h>
int main(){
    float a = 2.5;
    printf("%d", (int) a);
    return 0;
}
```

If float and int are present in a formula together, I will take care to convert the int to float before performing the operation since float is more powerful



```
float c = 2/3;
```

```
float c = 2/3.0;
```

```
float c = 2/(float)3;
```

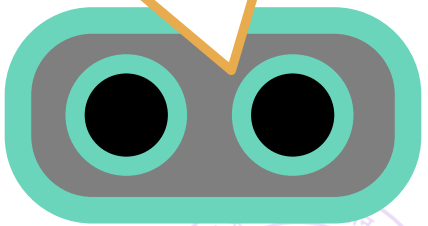
Is 2.0 the same as 2? (JEE 2018)


```
#include <stdio.h>
int main(){
    float a = 2.5;
    printf("%d", (int) a);
    return 0;
}
```

```
#include <stdio.h>
int main(){
    float a = 2.5;
    printf("%d", (int) a);
    return 0;
}
```

However, in this case, 2 and 3 are both are int so I will not bother converting them to float since int divided by int is just int

If float and int are present in a formula together, I will take care to convert the int to float before performing the operation since float is more powerful



```
float c = 2/3;
```

```
float c = 2/3.0;
```

```
float c = 2/(float)3;
```

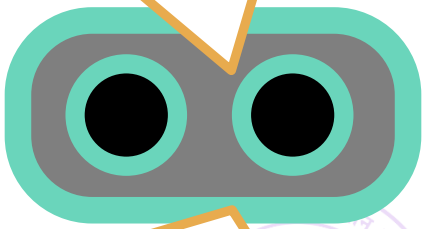
Is 2.0 the same as 2? (JEE 2018)


```
#include <stdio.h>
int main(){
    float a = 2.5;
    printf("%d", (int) a);
    return 0;
}
```

```
#include <stdio.h>
int main(){
    float a = 2.5;
    printf("%d", (int) a);
    return 0;
}
```

However, in this case, 2 and 3 are both are int so I will not bother converting them to float since int divided by int is just int

If float and int are present in a formula together, I will take care to convert the int to float before performing the operation since float is more powerful



```
float c = 2/3;
```

```
float c = 2/3.0;
```

```
float c = 2/(float)3;
```

Be careful, float can store much larger numbers (± 3.4e+38) than int can (± 2.1e+9)