

Type casting

CHAPTER 25



SURESH TECHS

C PROGRAMMING COURSE

Casting?

Hindu -> Christian

char -> int

int -> float

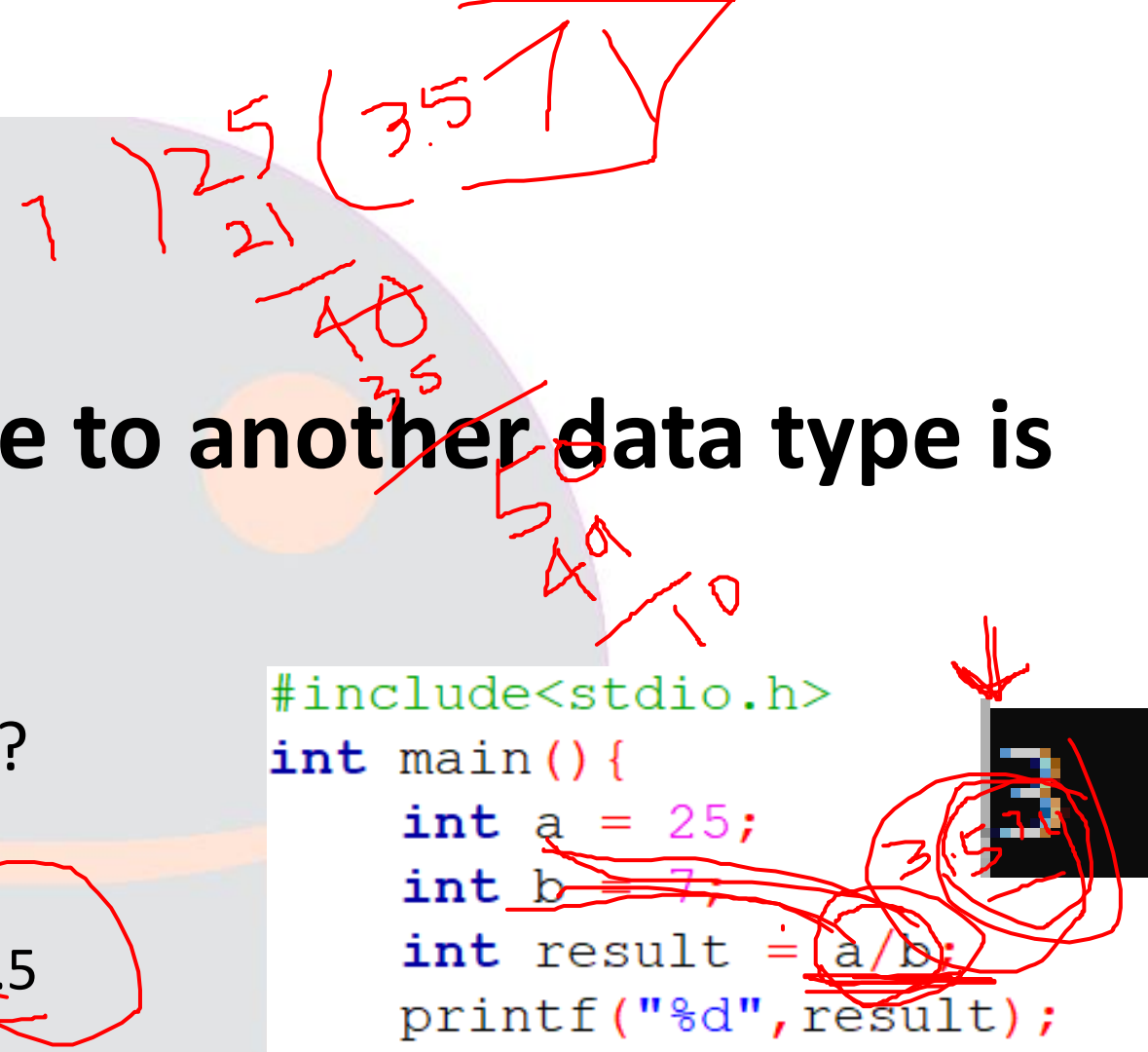
Type casting

Converting one data type to another data type is called type casting

What is the output of this program?

What if I want in fractional points? Ex: 3.5

```
#include<stdio.h>
int main(){
    int a = 25;
    int b = 7;
    int result = a/b;
    printf("%d", result);
    return 0;
}
```



Handwritten red annotations include a large circle around the number 3.5 in the example text, a circle around the variable 'result' in the code, and a circle around the expression 'a/b'. A red arrow points from the '3.5' in the example text to the 'a/b' expression. A screenshot of a terminal window shows the output '3' with a red arrow pointing to it. Other red annotations include a large 'X' over the code line 'int result = a/b;' and various numbers and symbols in the top right corner.

Type casting from int to float

```
#include<stdio.h>
int main(){
    int a = 25;
    int b = 7;
    int result = a/b;
    printf("%d",result);
    return 0;
}
```

```
#include<stdio.h>
int main(){
    int a = 25;
    int b = 7;
    float result = (float)a/b;
    printf("%f",result);
    return 0;
}
```

3.571429

Types of casting

- Implicit typecasting
- Explicit typecasting



Implicit type casting

- This type of type casting is done by the **compiler itself without any notification from the user**, and thus it is also known as **automatic type casting**.
- In this case **all of the different data types are converted to the largest data type** according to the following order:

```
#include<stdio.h>
int main() {
    int a = 25;
    int b = 7;
    int result = a/b;
    printf("%d", result);
    return 0;
}
```

char -> short int -> int -> unsigned int -> long -> unsigned long -> long long -> float -> double -> long double

Implicit conversion based on promotions

Also known type promotion

```
#include<stdio.h>
int main(){
    int a = 55;
    char b = 'c';
    a = a+b; //ASCII value of c is 99
    float res = a + 4.23;
    printf("a = %d, res = %f",a,res);
    return 0;
}
```

```
a = 154, res = 158.229996
```

Explicit type casting

- User defined
- In this conversion **user can define the type to which the expression is to be converted to**, it can be a larger or smaller data type.

```
#include<stdio.h>
int main() {
    int a = 55;
    char b = 'c';
    a = a+b; //ASCII value of c is 99
    float res = a + 4.23;
    printf("a = %d, res = %f",a,res);
    return 0;
}
```

```
#include<stdio.h>
int main() {
    int a = 55;
    char b = 'c';
    a = a+b; //ASCII value of c is 99
    int res = (int) (a + 4.23);
    printf("a = %d, res = %d",a,res);
    return 0;
}
```

```
a = 154, res = 158
```


Cast operator

(data-type)expression;

```
#include<stdio.h>
int main() {
    int a = 55;
    char b = 'c';
    a = a+b; //ASCII value of c is 99
    int res = (int) (a + 4.23);
    printf("a = %d, res = %d", a, res);
    return 0;
}
```

a = 154, res = 158

int is converted to float implicitly

```
#include<stdio.h>
int main() {
    //Type casting
    int a = 35;
    float b;
    b = a;
    printf ("b is: %f", b);
    return 0;
}
```

A black rectangular box containing the text 'b is: 35.000000' in a yellow, monospaced font, representing the output of the C program. The background of the slide features a large, faint, stylized smiley face with orange and grey segments.

b is: 35.000000

Conversion from int to char explicitly

```
#include<stdio.h>
int main() {
    int x = 105;
    char c;
    //character with ASCII value 105 will be stored in c
    c = (char) (x);
    printf("ASIIC value of 105: %c", c);
}
```

ASIIC value of 105: i

What next?

- Decision making and branching

