data → how do we store data in program.

data types ⇒

char → character

int → integer

float → real number

double ↗

bool → boolean ⇒ true / false

data type var. name;

char ch;

$\boxed{ch = \underbrace{'a'}_{\text{character constant}}; \approx ch = 97;}$

assignment

var. name = value

$\Uparrow$

store value in var. name

92 → integer constant
101 →

2.973 → real number constant
1.0e+10 →

Initialization

int i = 10;

New way to initialize

$\boxed{\begin{array}{l}\text{int i (10);}\\ \text{int i = \{10\};}\end{array}}$

Initialization $\Rightarrow$ Happens only once per variable, at time of declaration.

Assignment $\Rightarrow$ Can happen multiple times. After variable is declared.

Constant variables $\Rightarrow$ Named constants.

i = 92;   i = YEAR_OF_BIRTH;

const int YEAR_OF_BIRTH = 92;

$\uparrow$

Keyword declares a constant variable.

YEAR_OF_BIRTH = 100; $\times$ ← as YEAR_OF_BIRTH is constant

int x; ← un initialized variable

const int A; $\times$ ← const must be initialized.

↑

const must be initialised & their values can not be changed later.

Operate on data $\Rightarrow$ expression

$$\begin{tikzcd}
\text{expression} \arrow[d] & & \\
\text{data} \arrow[r] \arrow[dr] & \text{variable} & \\
& \text{constant} & \text{operands} \\
\text{operation} \arrow[r] & \text{operator} &
\end{tikzcd}$$

Arithmetic Operators

+   −   *   /

} binary operators

$\Downarrow$

work on two operands

$\boxed{2 + 5} \rightarrow$ expression

$\Downarrow$ result

value & type of value

$$\begin{tikzcd}
2 + 1.5 \arrow[r, Rightarrow] & 3.5 \arrow[d] \\
& \text{double}
\end{tikzcd}$$

2 ↑ int

1.5 ↑ ~~float~~ double

1.5 → double constant

1.5f → float constant

Implicit type conversion / upcasting.

$$\begin{tikzcd}
\text{char} \arrow[r] & \text{int} \arrow[r] & \text{float} \arrow[r] & \text{double}
\end{tikzcd}$$

$3 / 2 \rightarrow 1\cancel{.5}$ ← quotent

3 ↑ integer, 2 ↑ integer

1 ↑ integer

% ← modulus

5 % 2 → 1 ← Remainder of division

5 % 2.5 $\cancel{\rightarrow}$ Syntax error as % can only work on integer data.

3 / 2.5 → 1.5

result

val = no1 / no2;

→ / has higher precedence than =

val = result;

value of right operand will be stored in left operand

value of this expression ?

$\Downarrow$

value that is stored in left operand.

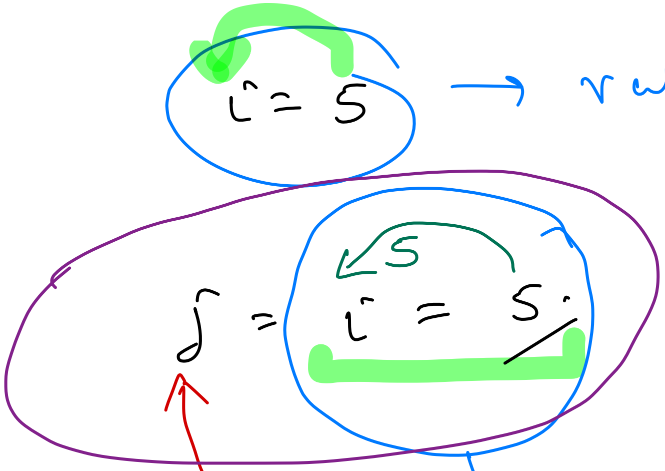
$i = 5$ $\rightarrow$ value of expression 5

$j = i = 5;$ $\rightarrow$ = is right to left associative

5 value assigned to j

5

value of expression = 5



$x = 10 + (\cancel{2 = 3})$

$\uparrow$

2 is constant & can not be changed.

$\underline{\underline{\text{var}}} = \text{value}$

$\uparrow$

left operand must be a variable

OR

left operand must have L-value.

Relational operators

<   >   <=   >=

==   !=

binary operators

op1 < op2 → result of type bool

value: true / false

if marks less than 40 then
Fail
else
Pass

Conditional statement

Conditional statement in C++

Keywords: if else

if condition → something that gives true/false

statement / statement block

[else

statement / statement block]

```
if (marks < 40)
    std:: cout << "Fail";

else
    std:: cout << "Pass";
```

```
if (marks < 40) {
        std:: cout << " ... ";
        std:: cout << " ... ";
}
else
        std:: cout << " ... ";
```

statement block

if marks are greater than or equal to 70
then
Pass with Dist.

else if marks greater than or equal to 60

then
        Pass with First Class

else if marks greater than or eq to 40
    then
            Pass

    else
            fail

⇓

```
if ( marks >= 70 )
        std::cout << " Pass with Dist";
else if ( marks >= 60)
        std::cout << " Pass with FC";
else if ( marks >= 40)
```

```
    std::cout << "Pass";
else
    std::cout << "Fail";
```

```
if (x < y)
    if (a < b)
        std::cout < "1";
else
    std::cout << "2";
```

```
if (x < y) {
        if (a < b)
            std::cout << "1";
}
else
        std::cout << "2";
```

Logical operators

&&	||	!
and	or	not

$$\underbrace{\text{and} \qquad \text{or}}_{\text{binary}} \qquad \underbrace{\text{not}}_{\text{unary}}$$

$$\underbrace{(\text{Condition 1})}_{\text{left operand}} \;\&\&\; \underbrace{(\text{condition2})}_{\text{right operand}}$$

&& → true if both operands are true else false

|| → true if any one operand is true else false

! operand → true if operand is false else false