

The "main" function

→ every C++ will have a main function.

→ this is the function executed first when program is run.

• for main() it will always be int.

→ the function main() always return 0 in case of successful completion.

To print new line

→ you can use 'std::endl' or you can use '\n'

* A variable is a memory

address where data can be stored

& changed. → specify both name & data type.

DATA TYPES

int. (for integers)

char

float & double

* important.

sizeof(variable_name or data type)

int var_name;

int var_name = 3;

int var_name {3};

} ways to declare

→ you can define variable anywhere in the middle of program.

int Type

→ generally occupies 4 bytes (32 bits) of memory

↳ long occupies 4 bytes

↳ short occupies 2 bytes

↳ int accessed faster than short. So use short if you're low on space.

char Type

→ occupies 1 byte

→ use of single quotes

→ double quotes are used for writing strings.

Date: _____

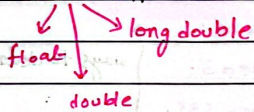
M	T	W	T	F	S	S
---	---	---	---	---	---	---

Escape Sequences

- `\n` → new line
- `\t` → tab
- `\\` → backslash
- `\"` → double quotes
- etc
- `\xdd` → hexadecimal notation.

} represents single character.

float type



- ↳ also takes up 4 bytes like int
- ↳ precision is 7 digits.

`float pi {3.14159f};`

`float light_speed = 2.99E8f;` // Exponentials.

`float absolute_zero = -273.15f;`

both capital F, ←

2. lowercase f works!

① type double

↳ size : 8 bytes

↳ Range: $1.7E-308$ to $1.7E308$

↳ Precision: 15 digits.

② type long double

↳ compiler dependent

Note:- ① Avoid doing comparisons b/w 2 floating point numbers / variables if you can

↳ especially if values are closer together

② There are rounding and/or precision errors that may result in unexpected outputs.

bool Type

- ↳ 1 byte of storage
- ↳ only two possible values - true or false.

unsigned Data Types

- ↳ useful when working with positive-only data

keyword	bytes	Comments
unsigned char	1	→ // single line comment.
unsigned short	2	→ /* multi-line comment */
unsigned int	4	
unsigned long	4	

Type Conversions

- lower gets converted to higher
- automatically done for us by compiler. → * when the compiler doesn't do for us we use **CASTING**.

long double highest
double
float
long
int
short
char lowest

Date: _____

M	T	W	T	F	S	S
---	---	---	---	---	---	---

Add (+)

+= item

Subtract (-)

-= item

Divide (/)

*= item

Multiply (*)

/= item

Remainder (%) → only works with integers (char, short, int, long). %= item.

↓

% has higher precedence than insertion
operator (<<)

Total ++ ; (postfix variant)

++ Total ; (prefix variant) } can only add 1 using increment operator.

Date: _____

M	T	W	T	F	S	S
---	---	---	---	---	---	---

`totalWeight = avgWeight * ++count;` // prefix

→ (Prefix Case) `count` is incremented first and then the multiplication is performed

→ (Postfix Case) `→ (count++)`, multiplication would've been performed first & then increment.

* if you want to keep a value constant, use `const`
`const double PI {3.14159};`

`std::cout` an object of class `ostream`

→ used to print content on screen

→ used alongside an insertion operator (`<<`)

`std::cin` an object of class `istream`.

→ used to take input from the user

→ used alongside an extraction operator (`>>`)

LOGICAL OPERATORS

`&&` Logical AND

`||` Logical OR

`!` Logical NOT

`include`: directive copies that file into your program.

`namespace`: a collection of names & their definitions. Allows different namespaces to use the same names without confusion.