PF

- Add .0 at the end of any number when only integers are being computed and a floating answer is expected in an expression or sub-expression. e.g: (2/3) * 4 ⟶ (2/3.0) * 4

- Always use operators ⟶ p(p-a)(p-b) ✗ // p*(p-a)*(p-b) ✓

- remember semi-colon and braces
- Use single quotes when comparing char
- Data type precedency: In any calculation, answer is given in the data type that has the largest precedency. Float being the highest.
- /* ... */ (Multi-line Comments)
- \t : Tab character

  \" : double quote character
- For scientific numbers: int float f1 = 35e3; // e/E means "x 10^sth"

  // f2 = 36E2;

Data Types

- short int, int, long int
- float, double
- char

e.g: int 35.234e4; gives 352340

int 35.234e2; gives 3523

Assignment operators

- +=
- -=
- /=
- *=
- %=
- &= → Performs and operation
- |= → // or operation
- ^= → XOR operation
- >>= → Bitwise right
- <<= → Bitwise left

These apply to the variable itself.

These are used like n+=5 → n=n+5

n*=5 → n=n*5

Note: cout<< (5>3);

will give output 0 which is a boolean value.

- First expression is evaluated then output is given

- non-zero values considered as true (1). That's why cout << (5 & 4); is 1.

Logical Operators (if else wali)

- && (and)
- || (or)
- ! (not) e.g: !(x>5 && x<10)
- !=
- ==

Cmath Functions

- sqrt(num) $\sqrt{}$
- pow(base, exponent)
- log(num) natural log
- exp(num) e^{num}

Arrays

- Selection sort

- `int arr = { 1,2,3,4 };` (Size detected automatically)
  - num of elements
- `int arr[4] = { 1,2,3,4 };`
- `int arr[4];` (A garbage value will be stored in this case)

2-D

- `int arr[2][3] = { {1,2,3}, {4,5,6} };`
  - row
  - columns

Functions

return datatype

parameter datatype

- `int sum(int num) {`
  - by ref "`&num`"
  - if no value of parameter (`int num = 5`) passed

`}`

- `int num(int arr[])`

- `int num( int arr[][5])`
  - ↳ col. num is a must but row num is optional

sorting Algos