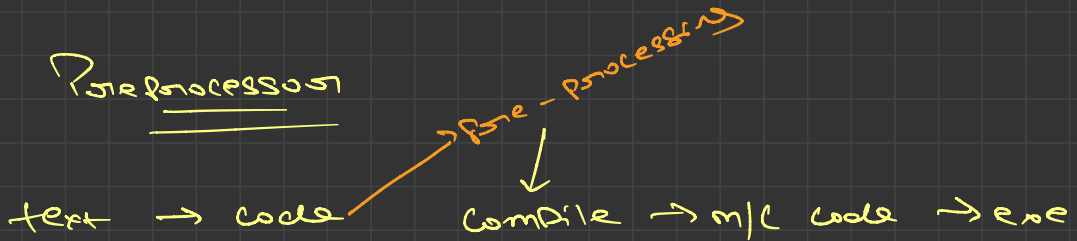
Macros

Macros are pre-processor directives that allow you to define constants, functions or code snippets that can use throughout your code.

They are typically defined using the #define directive & are evaluated by the preprocessor before the code is completed.

Macros can be used for the variety of purpose, such as defining constant or creating shorthand for commonly used expression.

Preprocessor



first compiler read the code sequentially & where they find # define

| pre-process

replace

before Compiling pre-processor change all the PI with this value

```
' #include<iostream>
using namespace std;

// create macros -> does not consume any space in memory
#define PI 3.14159465

// float circleArea(float r){
//     return 3.14 * r * r;
//     // we can make macro of 3.14
// }

float circleArea(float r){
    return PI * r * r;
}

float circleParameter(float r){
    return 2 * PI * r;
}

int main(){
    cout << circleArea(34.4) << endl;
    cout << circleParameter(3.4) << endl;
}
```

We can also define this like ternary operator

to compare the two values

```
#include<iostream>
using namespace std;

// create macros
#define MAXX(x,y) (x > y ? x : y)

void fun(){
    int x = 6;
    int y = 17;
    // int z = x > y ? x : y;
    // using this write again and again make macros
of z
    int z = MAXX(x,y);
    cout << z << endl;

}

void fun2(){
    int x = 25;
    int y = 17;
    int z = MAXX(x,y);
    cout << z << endl;
}

void fun3(){
    int x = 6;
    int y = 2;
    int z = x > y ? x : y;
    cout << z << endl;
}

int main(){
    fun();
    fun2();
    fun3();
}
```