

Macros



Can you recognize what the following program does?

```
main(){
  int i,n,s,N;
  Reset(i);
  Reset(s);
  Input_an_Integer(N);
  forever{
    Input_an_Integer(n);
    Add_with(s,n);
    Increment(i);
    Stop_if_Done(i,N);
  }
  Display(s);
}
```

Macros in C

- C preprocessor → a separate first step in compilation.
- `#define` is used to replace a token by arbitrary sequence of characters.

Macro substitution

#define *name replacement_text*

Example:

```
#define PI 3.14
```

```
main(){
```

```
.....
```

```
double a = PI*r*r;    → double a = 3.14*r*r;
```

```
.....
```

```
}
```

Macro substitution

#define *name replacement_text*

Example:

```
#define forever for( ; ; )
```

```
main(){
```

```
.....
```

```
    forever{ → for( ; ; ) {  
        printf("Enter a positive number: ");  
        scanf("%d",&n);  
        if(n >= 0) break;  
    }
```

```
.....
```

```
}
```

Macros with arguments

#define *name replacement_text*

Example:

```
#define increment(x) x++
```

```
main(){
```

```
.....
```

```
    k = 1;
```

```
    while(k <= n){
```

```
        printf("%d", k);
```

```
        increment(k); → k++;
```

```
    }
```

```
    increment(n); → n++;
```

```
}
```

Macro with arguments (benefit: works with all data types)

```
#define square(x) (x)*(x)
```

```
main(){
```

```
    int x = 5;
```

```
    double y = 5.4;
```

```
    int p = square(x);
```

→ int p = (x)*(x);

```
    double z = square(y);
```

→ double z = (y)*(y);

```
}
```

Need for parentheses

```
#define square(x) x*x
```

```
main(){
```

```
    int x = 5;
```

```
    double y = 5.4;
```

```
    int p = square(x+1);    → int p = x+1*x+1;
```

```
    double z = square(y+1); → double z = y+1*y+1;
```

```
}
```


Macros with multiple lines

```
#define swap(t,x,y)  t = x; x = y; y = t;
```

```
main(){
```

```
    int v, p = 10,q = 30;
```

```
    float a = 3.0,b = 10.3, c;
```

```
    ...
```

```
    swap(v, p, q);    → v = p; p = q; q = v;
```

```
    swap(c, a, b);    → c = a; a = b; b = c;
```

```
    .....
```

```
}
```

Works with any
data type

Macros with multiple lines: problem when used in if-else

```
#define swap(t,x,y)  t = x; x = y; y = t;
```

```
main(){
```

```
....
```

```
if (p < q) swap(v, p, q);    → if (p < q) v = p; p = q; q = v;;
```

```
}
```

Macros with multiple lines: problem when used in if-else

```
#define swap(t,x,y)  {t = x; x = y; y = t;}
```

```
main(){
```

```
....
```

```
if (p < q) swap(v, p, q);    → if (p < q) {v = p; p = q; q = v;};
```

```
}
```

Macros with multiple lines: solution to the problem when used in if-else

```
#define swap(t,x,y)  do{ t = x; x = y; y = t;} while (0)
```

```
main(){
```

```
....
```

```
if (p < q) swap(v, p, q);    →if (p < q)
                             do {v = p; p = q; q = v;} while (0);

}
```

Macros side effect: use with caution

```
#define max(a,b) ((a > b)? (a) : (b))
```

```
main(){
```

```
....
```

```
r = max(p,q);           → r = ((p > q) ? (p) : (q));
```

```
r = max(p++,q++);      → r = ((p++ > q++) ? (p++) : (q++));
```

```
}
```

Does it
make sense
now?

```
#define forever for(;;)
#define Input_an_Integer(N) scanf("%d",&N)
#define Add_with(s,n) s = s+n
#define Increment(i) i++
#define Stop_if_Done(i,N) if(i>=N) break;
#define Display(a) printf("%d\n",a)
#define Reset(i) i = 0

main(){
    int i,n,s,N;
    Reset(i);
    Reset(s);
    Input_an_Integer(N);
    forever{
        Input_an_Integer(n);
        Add_with(s,n);
        Increment(i);
        Stop_if_Done(i,N);
    }
    Display(s);
}
```