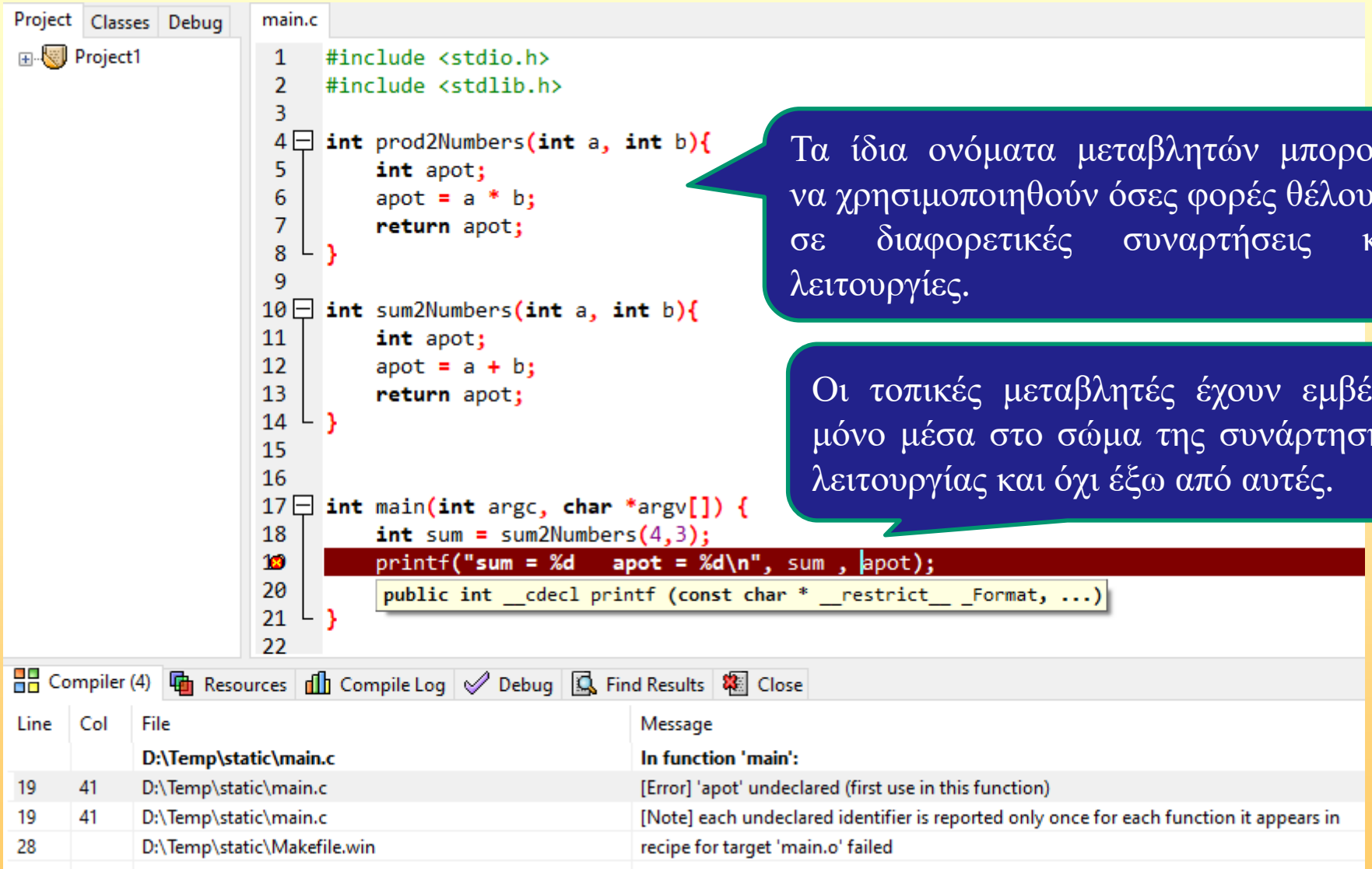

# Προγραμματισμός Υπολογιστών

## Καθολικές – Τοπικές – Στατικές Μεταβλητές
## Χρήση #define για συναρτήσεις

Νικόλαος Ζ. Ζάχαρης
Καθηγητής

Πανεπιστήμιο Δυτικής Αττικής
Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

# Τοπικές μεταβλητές



# Καθολικές μεταβλητές

main.c

```c
#include <stdio.h>
#include <stdlib.h>

int sum = 0;

void testA() {
    printf("TestA sum = %d\n", sum);
    sum += 5;
}

void testB() {
    printf("TestB sum = %d\n", sum);
    sum += 10;
}

void testC() {
    int sum = 7;
    printf("TestC sum = %d\n", sum);
    sum += 5;
}

int main(int argc, char *argv[]) {
    sum = 5;
    testA();
    testB();
    testC();
    printf("sum = %d\n", sum);
    return 0;
}
```

```
D:\Temp\static\Project1.exe
TestA sum = 5
TestB sum = 10
TestC sum = 7
sum = 20
```

Όλες οι μεταβλητές που ορίζονται εκτός του σώματος οποιασδήποτε συνάρτησης ή λειτουργίας ονομάζονται καθολικές μεταβλητές και έχουν εμβέλεια σε όλο το πρόγραμμα.

Επιτρέπεται μέσα στο σώμα μιας συνάρτησης ή λειτουργίας, μία τοπική μεταβλητή να έχει το ίδιο όνομα με μία καθολική μεταβλητή. Τότε στην περίπτωση που εκτελείται αυτή η συνάρτηση ή λειτουργία, θεωρούμε ότι όλες οι αλλαγές στη τιμή συμβαίνουν στην τοπική μεταβλητή.

# Στατικές μεταβλητές

[*] main.c

```c
#include <stdio.h>
#include <stdlib.h>

void DemoStatic(int x) {
    static int k = 0;

    if(x == 0){
       printf("k = %d\n", k);
    }
    k++;
}

int main(int argc, char *argv[]) {
    DemoStatic(1);
    DemoStatic(1);
    DemoStatic(1);
    DemoStatic(1);
    DemoStatic(0);
    return 0;
}
```

D:\Temp\static\Project1.exe

```
k = 4
```

Οι μεταβλητές που δηλώνονται μέσα στο σώμα μιας λειτουργίας ή συνάρτησης σαν static διατηρούν τη τιμή ανάμεσα στις διαδοχικές κλήσεις τους

Κατά αυτό τον τρόπο μπορούμε να γνωρίζουμε πόσες φορές έγινε κλήση της συνάρτησης.

# Χρήση του #define για συναρτήσεις

main.c

```
Min = 2
Sum = 5
Pwr = 9
Pwr = 11
```

```c
#include <stdio.h>
#include <stdlib.h>

#define Min(a, b) (a < b) ? a : b
#define Sum(a, b) a + b
#define Pwr2(a)  a * a

int main(int argc, char *argv[]) {

    printf("Min = %d\n", Min(2,3));
    printf("Sum = %d\n", Sum(2,3));
    printf("Pwr = %d\n", Pwr2(3));
    printf("Pwr = %d\n", Pwr2(3+2));

    return 0;
}
```

Μπορούμε να χρησιμοποιήσουμε τη δήλωση #define για τον ορισμό μιας συνάρτησης

Προσοχή όμως στα ορίσματα γιατί το #define αντικαθιστά το σημείο της κλήσης με τις εντολές που έχουμε ορίσει και μπορεί να προκύψουν απρόσμενα αποτελέσματα.

Για παράδειγμα στη γραμμη 12 το αποτέλεσμα είναι 9 και είναι σωστό. Όμως στην επόμενη γραμμή το αποτέλεσμα είναι 11 αλλά δεν είναι το αναμενόμενο. Ο λόγος είναι ότι το σημείο κλήσης Pwr2(3+2) έγινε αντικατάσταση με τις εντολές :

3+2 * 3 + 2 = 3 + 6 + 2 = 11

Για να πάρουμε σαν αποτέλεσμα το 25 θα πρέπει να ξαναγράψουμε τη Pwr2 όπως παρακάτω :

**#define Pwr2(a)  (a) * (a)**