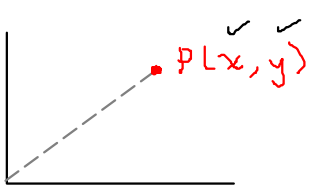
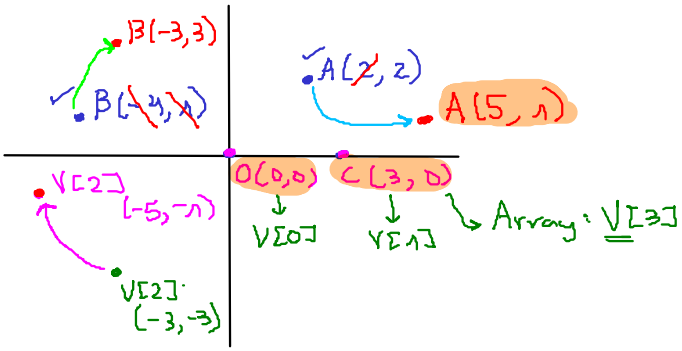
23/09/2021.

# Estructuras - Repaso

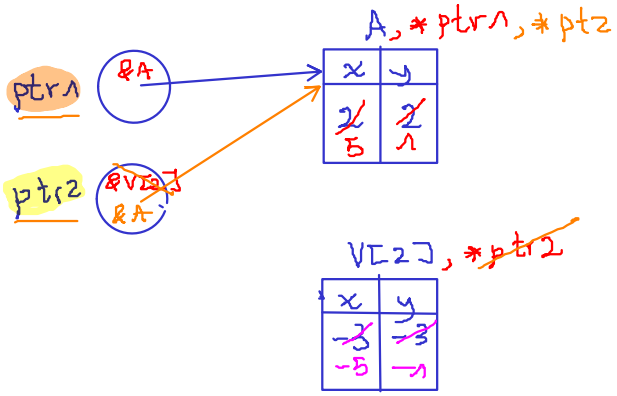



```
struct _Punto {
    double x,
    double y
};

typedef struct _Punto Punto;
```




```
Punto A = { 2, 2};
Punto B = { .x = -4
            .y = 1 };
```

```
Punto O, C ;
O.x = 0;
O.y = 0;
C.x = 3;
C.y = 0;
```




```
Punto V[3];
V[0] = O;
V[1] = C;
V[2].x = -3;
V[2].y = -3;
```

```
Punto *ptr1 = &A;
```

```
Punto *ptr2;
ptr2 = &V[2];
```

```
B.x = -3;
B.y = -3;
```
{ variable.miembro

```
ptr1->x=5; //A.x=5
ptr1->y=1; //A.y=1
```
{ apuntador -> miembro

```
V[2].x=-5;
ptr2->y=-1;
```

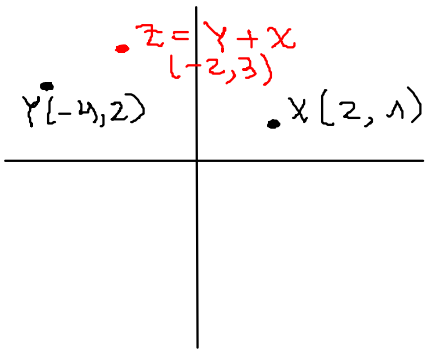
```
ptr2 = ptr1; // ptr2 = &A;
```

Funciones:

```
void imprimir_coordenadas (Punto *p) {
    printf ("(%lf , %lf)\n", p->x, p->y);
}
```

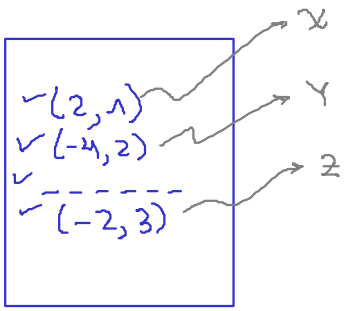
```
Punto sumar (Punto p1, Punto p2) {
    Punto s;
    s = p1 + p2
    return s;
}
```

Ejemplo usando las funciones:



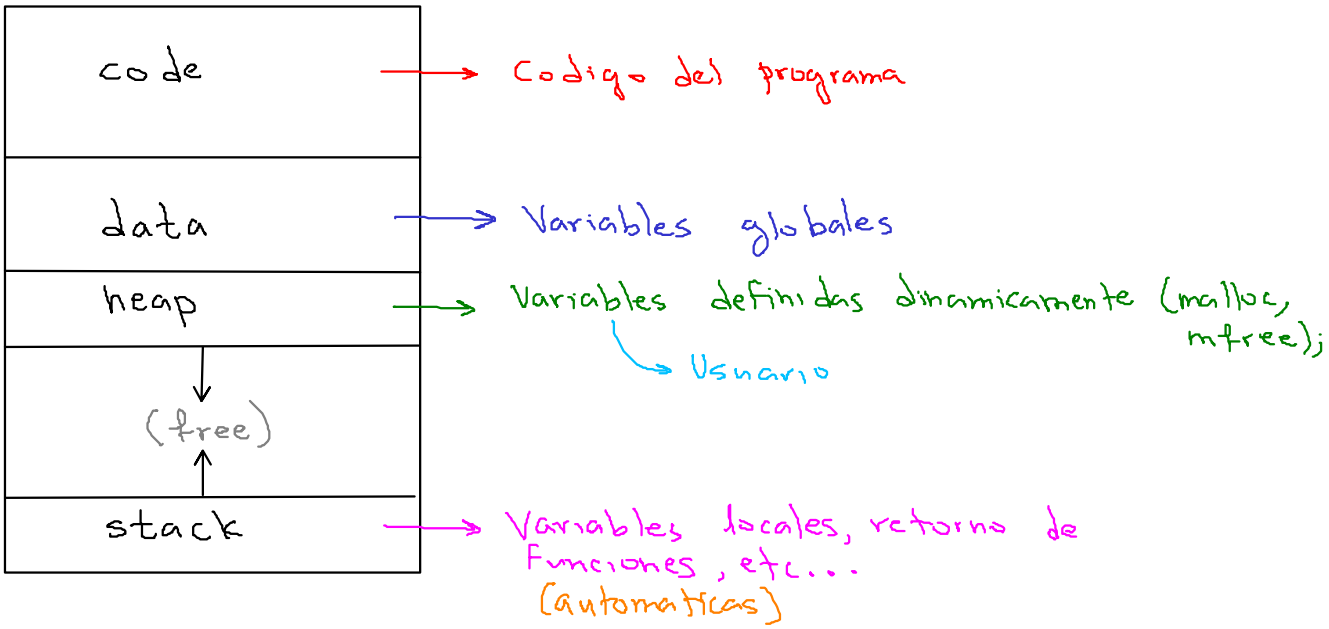

```
Punto X = {-4,2}, Y = {2,1}, Z;
Z = sumar (p1, p2);
```

```
imprimir_coordenadas (&X);
imprimir_coordenadas (&Y);
printF ("- - - - - - - \n");
imprimir_coordenadas (&Z);
```

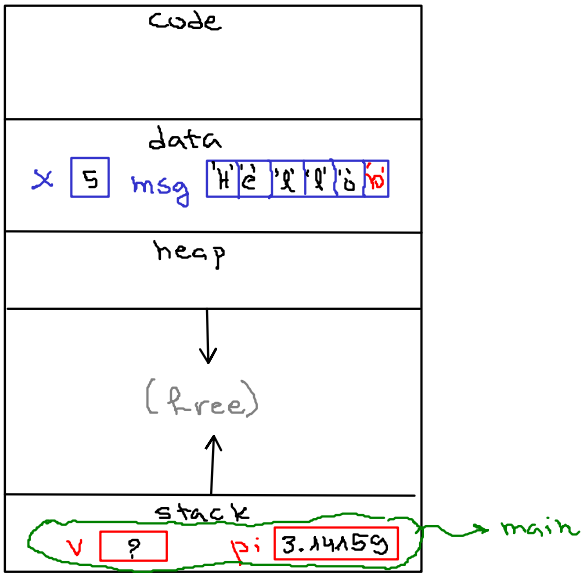



## Reserva dinamica de memoria.

### Mapa de memoria



```c
#include <stdio.h>

int x=5;      //global
char msg[] = "Hello";  //global

int main(int argc, const char* argv[]) {
  int v;    //local
  float pi = 3.14159;  //local
  printf("%d\n",x);
  printf("%f\n",pi);
  return 0;
}
```




C (gcc 4.8, C11) [unsupported]

```c
#include <stdio.h>

int x=5;
char msg[] = "Hello";

int main(int argc, const char* argv[]) {
  int v;
  float pi = 3.14159;
  printf("%d\n",x);
  printf("%f\n",pi);
  return 0;
}
```

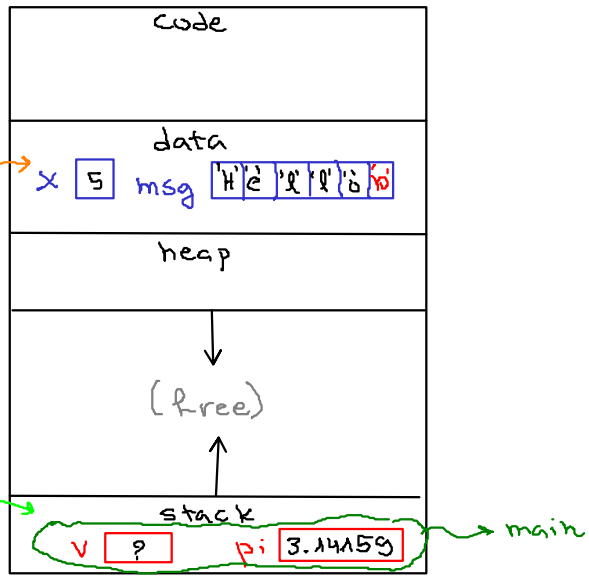
Print output (drag lower right corner to resize)

```
5
3.141590
```

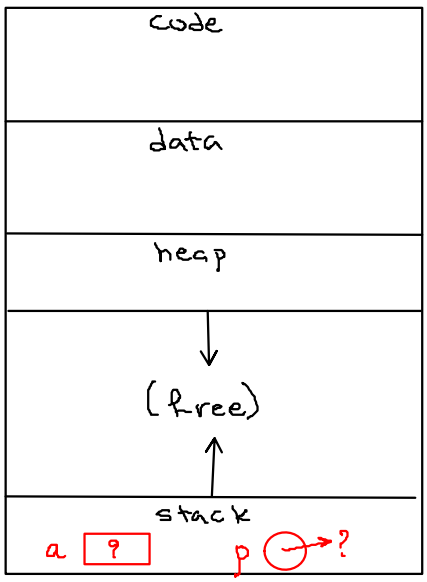
Stack Heap

Global variables
x int 5
msg array [0] char 'H' [1] char 'e' [2] char 'l' [3] char 'l' [4] char 'o' [5] char '\0'

main
argc int 1
argv pointer
pi float 3.14159




```c
%%tutor -l c -k
#include <stdio.h>

int main() {
  int a;  //local
  int *p = &a; //local
  *p = 5;
  printf("*p = %d\n", *p);
  return 0;
}
```

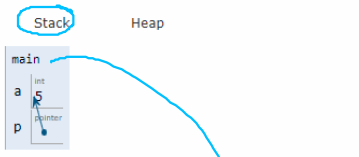


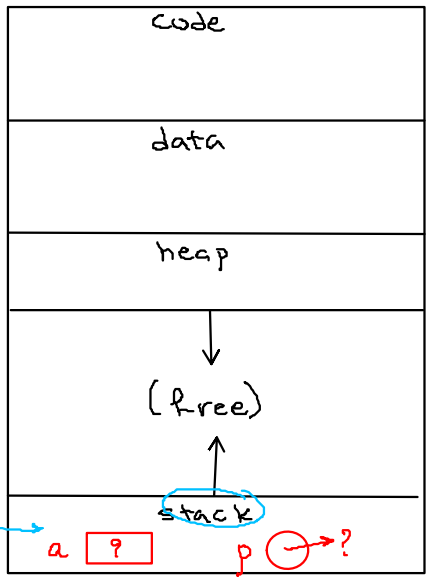
C (gcc 4.8, C11) [unsupported]

```c
#include <stdio.h>

int main() {
  int a;
  int *p = &a;
  *p = 5;
  printf("*p = %d\n", *p);
  return 0;
}
```

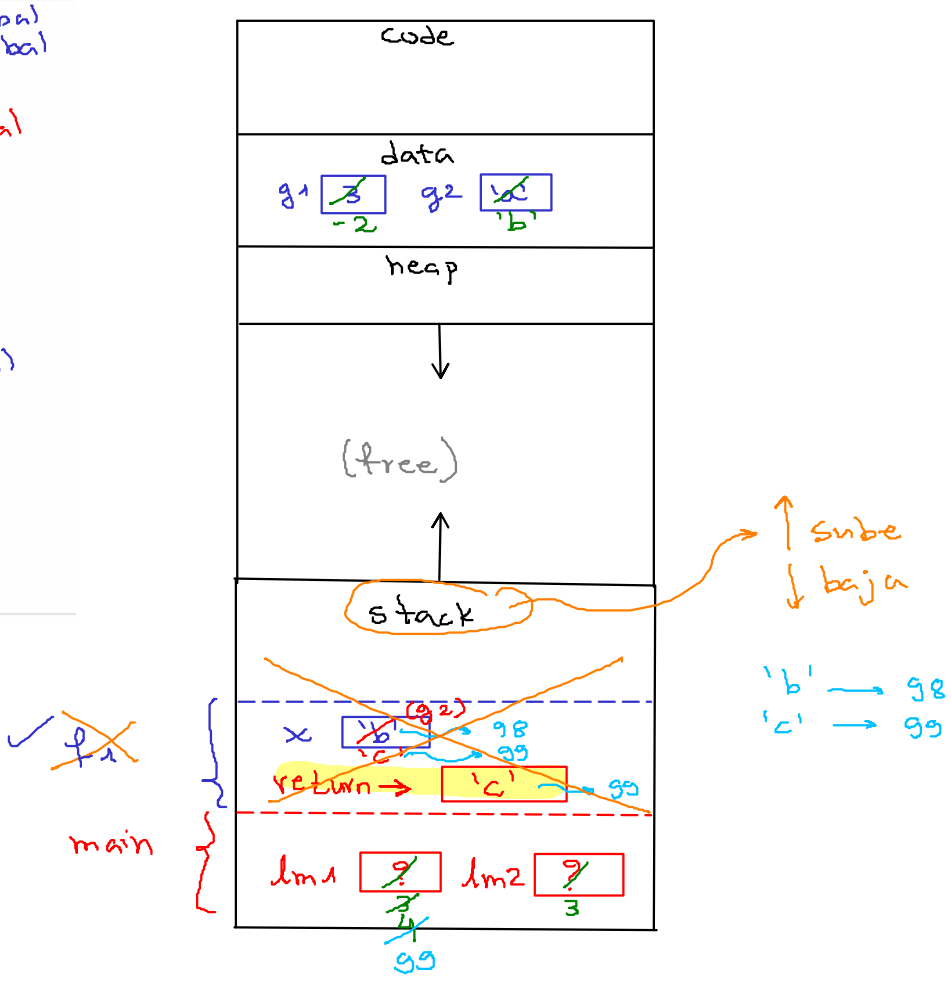
Print output (drag lower right corner to resize)

```
*p = 5
```



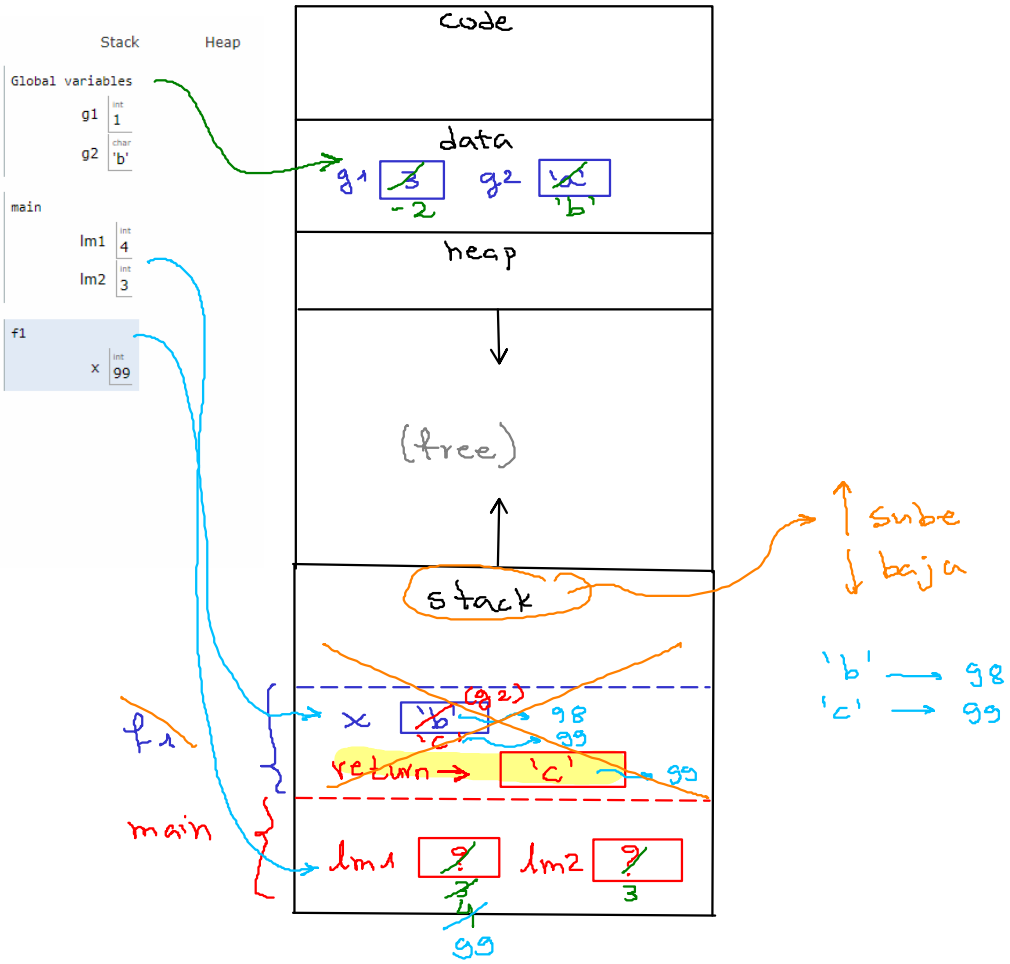

```c
int g1 = 3;          //global
char g2 = 'a';       //global

int main() {
  int lm1, lm2;      //local
  lm1 = 3,
  lm2 = lm1++;
  g2++;
  g1 -= 2;
  lm1 = f1(g2);

  return 0;
}

int f1(int x) {      //local
  return x++;
}


int f2() {
  g1 *= 2;
}
```



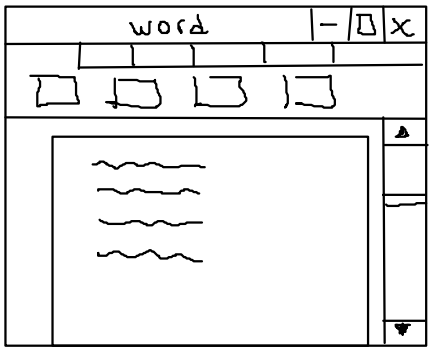
C (gcc 9.3, C17 + GNU extensions)
(known limitations)

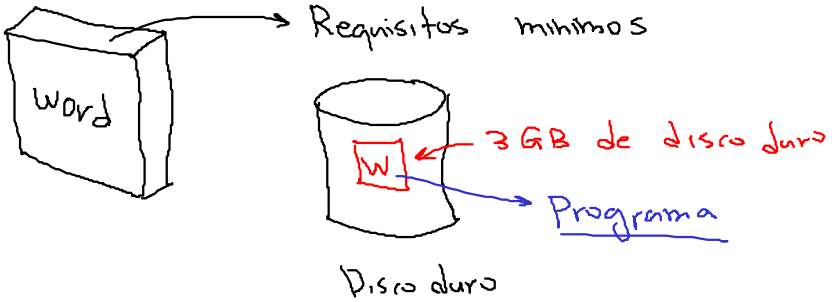
```
1  int g1 = 3;
2  char g2 = 'a';
3
4  int main() {
5    int lm1, lm2;
6    lm1 = 3,
7    lm2 = lm1++;
8    g2++;
9    g1 -= 2;
10   lm1 = f1(g2);
11
12   return 0;
13 }
14
15 int f1(int x) {
16   return x++;
17 }
18
19
20 int f2() {
```

Edit this code

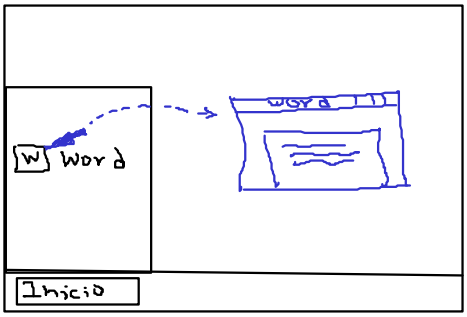


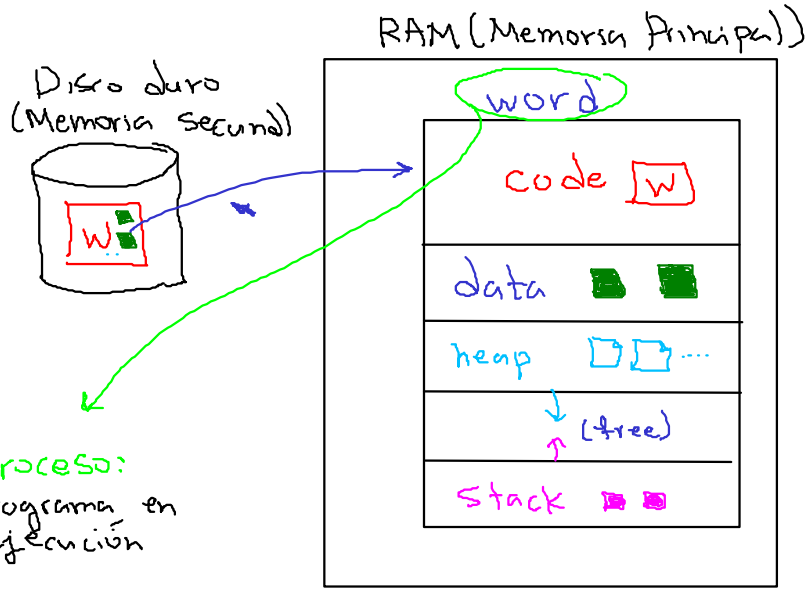

Por que es importante reservar dinamicamente memoria ?



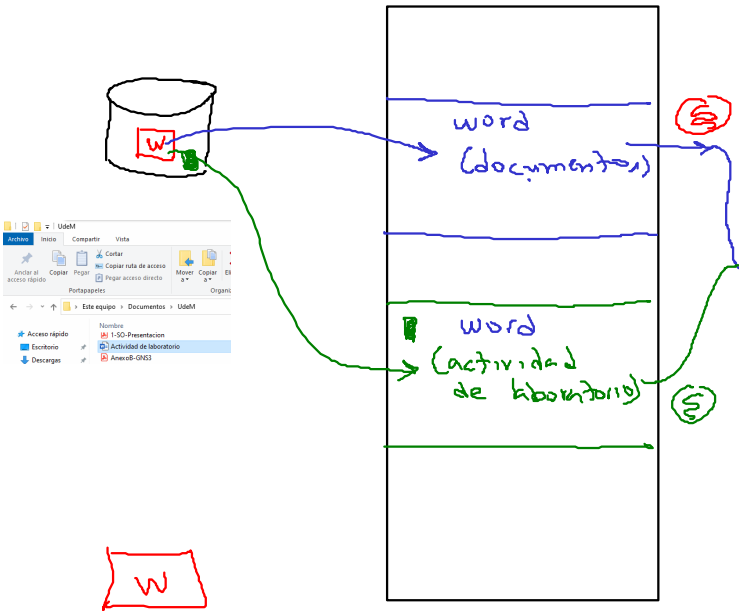



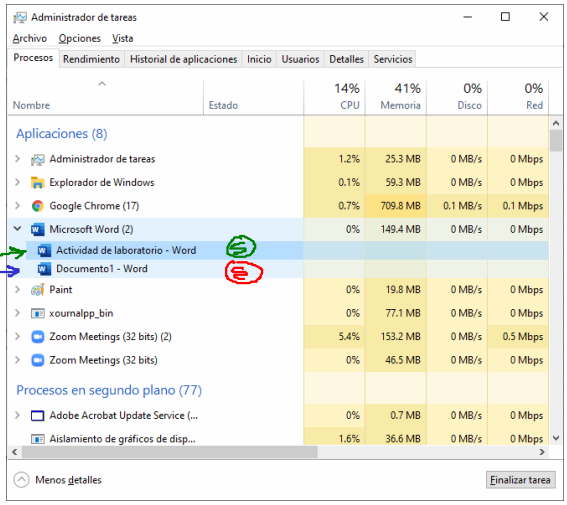

Programa ya instalado




Proceso:
Programa en ejecución








Hemos cargado "2 procesos".