

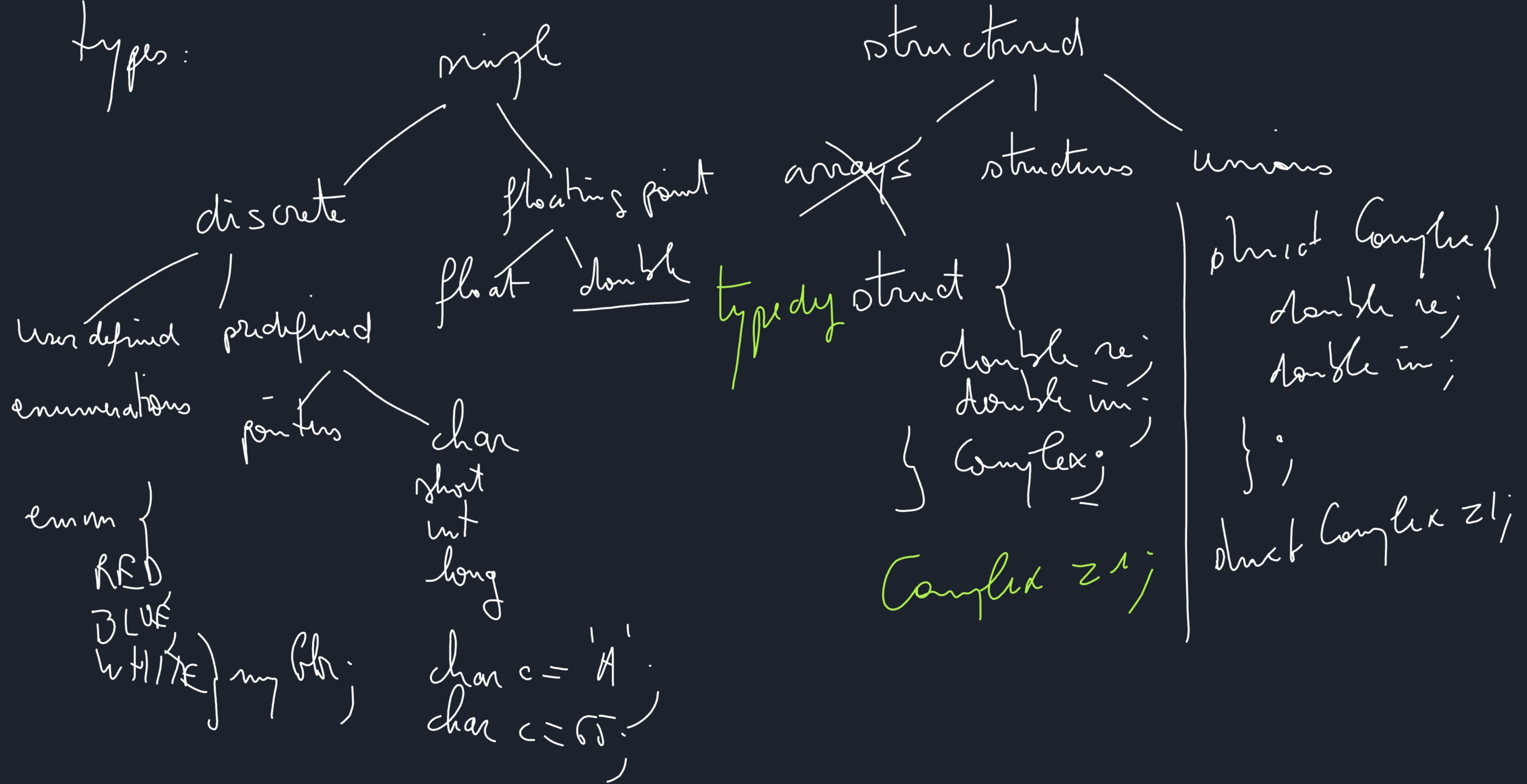
## header files .h

- function declarations
- pre-processor directives (# ...)
- user-defined types
- inclusion of other headers

## source files .c

- function definitions (main())
- 
- global variables
- inclusion of headers

types:

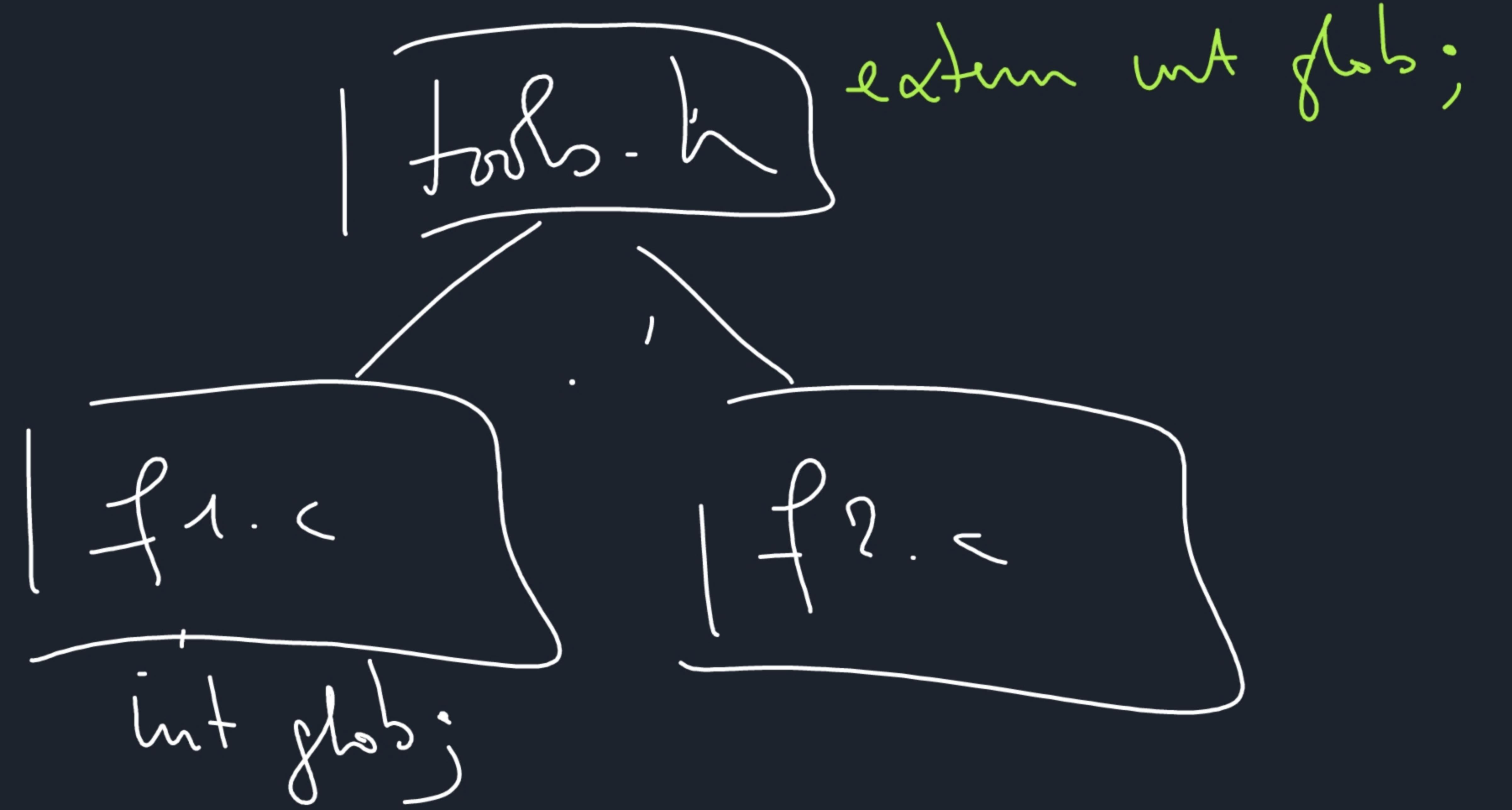


Variable <u>storage</u> :		automatic, static, extern, register	
Scope		auto	
life duration		static	
	- in {} : the {}	- in {} : the {}	
	- otherwise : global	- otherwise : the file	
	- in {} : exec'd {}	Whole prog exec	
	- otherwise : whole prog exec		

```

int nTimes() {
    static int n=1;
    :
    return n++;
}

```



register int i;  
for ( $i = 0; i < N; i++$ )  
 doSomething( $i$ );

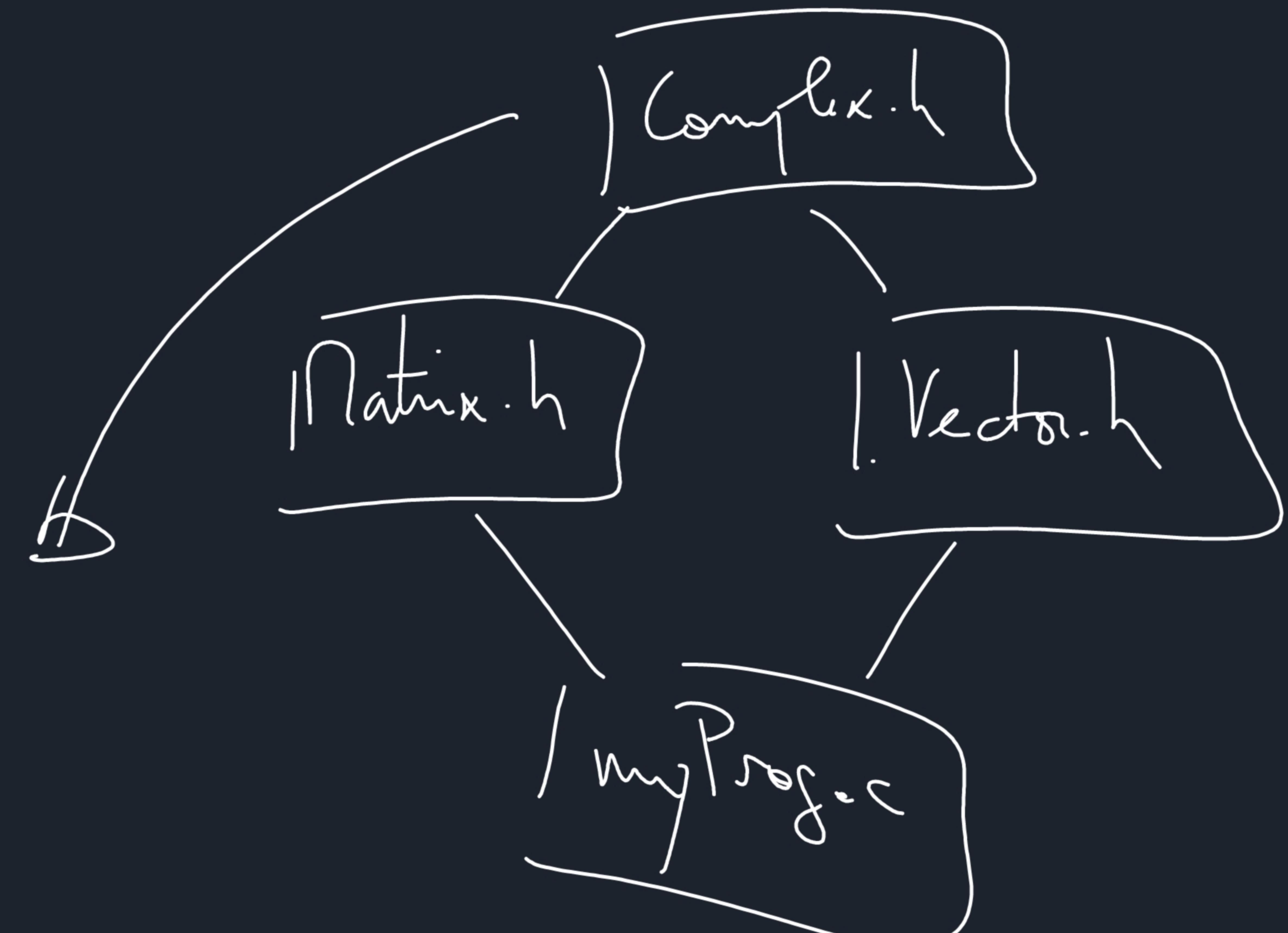
return type name (params);

double avg (double x, double y) { return (x + y) / 2; }

void

## Non pre-processor tasks

```
#include <stdio.h>
#include "myType.h"
#ifndef __COMPLEX__
#define __COMPLEX__
typedef struct {
    :
    :
} Complex;
#endif
```



#define BSIZE 1024

#define max(a,b) a < b ? b : a

int i=2, j=3, k;

k = max(i++, j++);      k = i++ < j++ ? j++ : i++

expected: i=3, j=4, k=3

got: i=3, j=5, k=4

i: 3  
j: 4  
k: 5

## I/O functions.

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

```
int printf (format, ...);
```

- strings ↗ "long int"

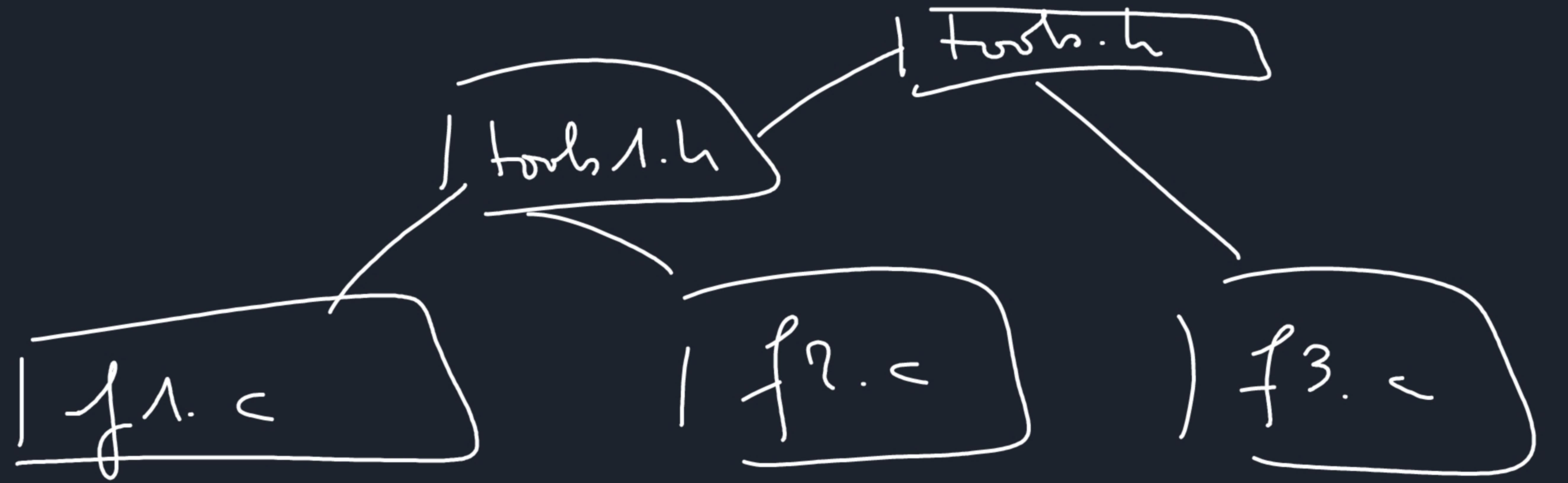
- special chars \t, \n, - -

- format markers %d      integers  
          f      float  
          s      strings  
          c      char

printf ("avg of %lf and %lf is %lf\n",

x, y, avg(x,y));

scanf ("%lf %lf", &x, &y);



## makefile

$$OBJ = f1.o \quad f2.o \quad f3.o$$

*myProg : S(OBJ)*

*cc -o myProg \$ (OBJ)*

*f1.o : f1.c tools1.h tools.h*

*cc -c f1.c*

*f1.o : f1.c tools1.h tools.h*

depend :

```
sed    '/^#DEPEND/q' makefile > mk-hmp
cc     -n *.
```

#DEPEND : don't delete this line !



# Pointers

Why ?

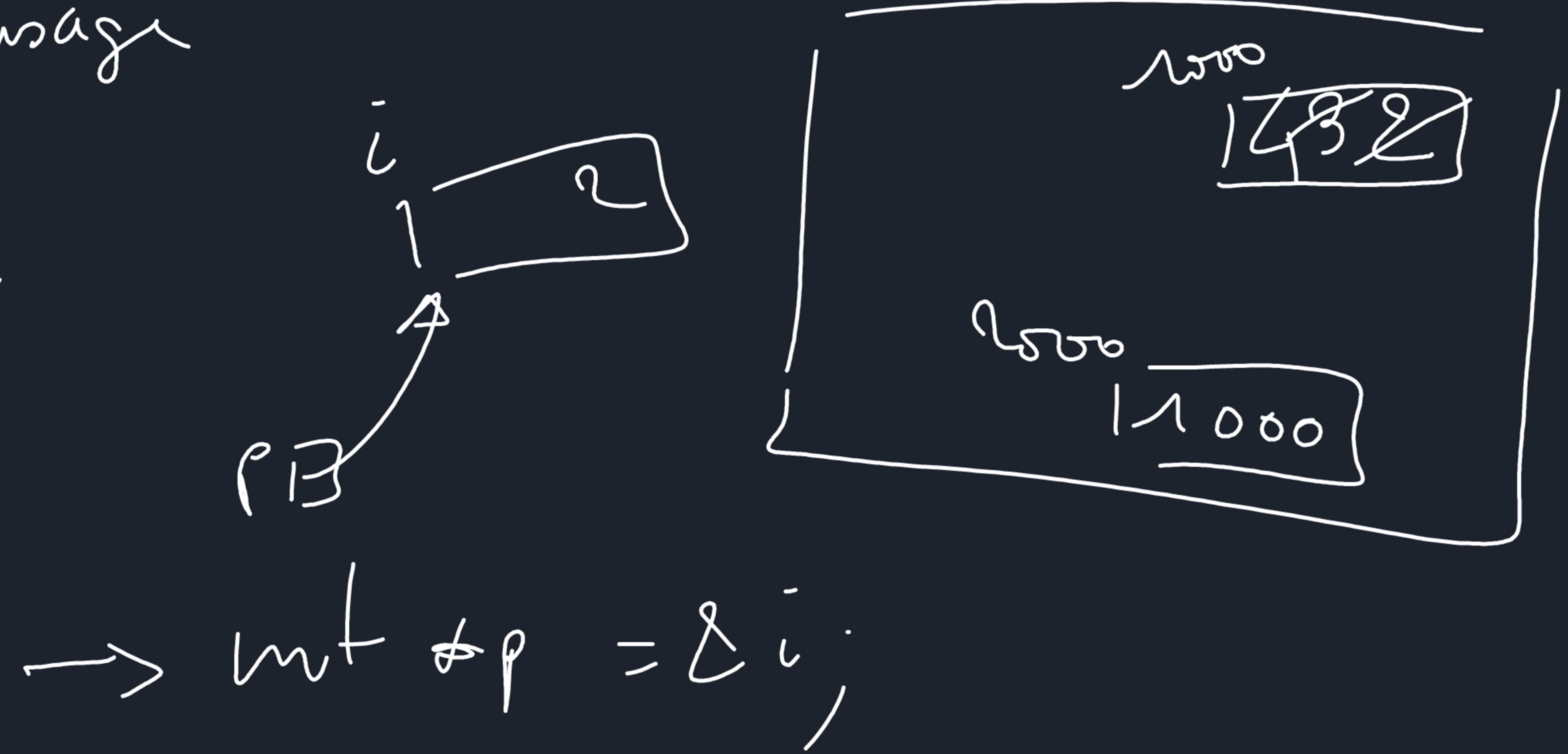
- ① create dynamic structures
- ② enable side effect in functions
- ③ optimize parameter passing in function

def: a pointer is a variable whose value is a memory address  
2 operators:  
    Unary \* : "object pointed to by"  
    Unary & : "address-of"

# definition and usage

int i = 2;

int \*p;  
p = &i;  
i = 3;  
\*p = {;



stack

ST	
name	value
i	1000
p	8000
a	11000

## Dynamic memory allocation

```
Bag * createBag (int capa) {  
    Article b[capa];  
    .  
    :  
    return theBag;  
}  
mid * malloc (int nBytes);  
mid free (mid * p);
```

functions:

```
{ int i; { int & f  
= (int &)  
malloc (10);  
;  
}  
}
```

## Side effects in functions

Pb: pass\_by\_value is the only mechanism in C

```

void triple(int*a) {
    (*a) = 3 * (*a);
}
    
```

```

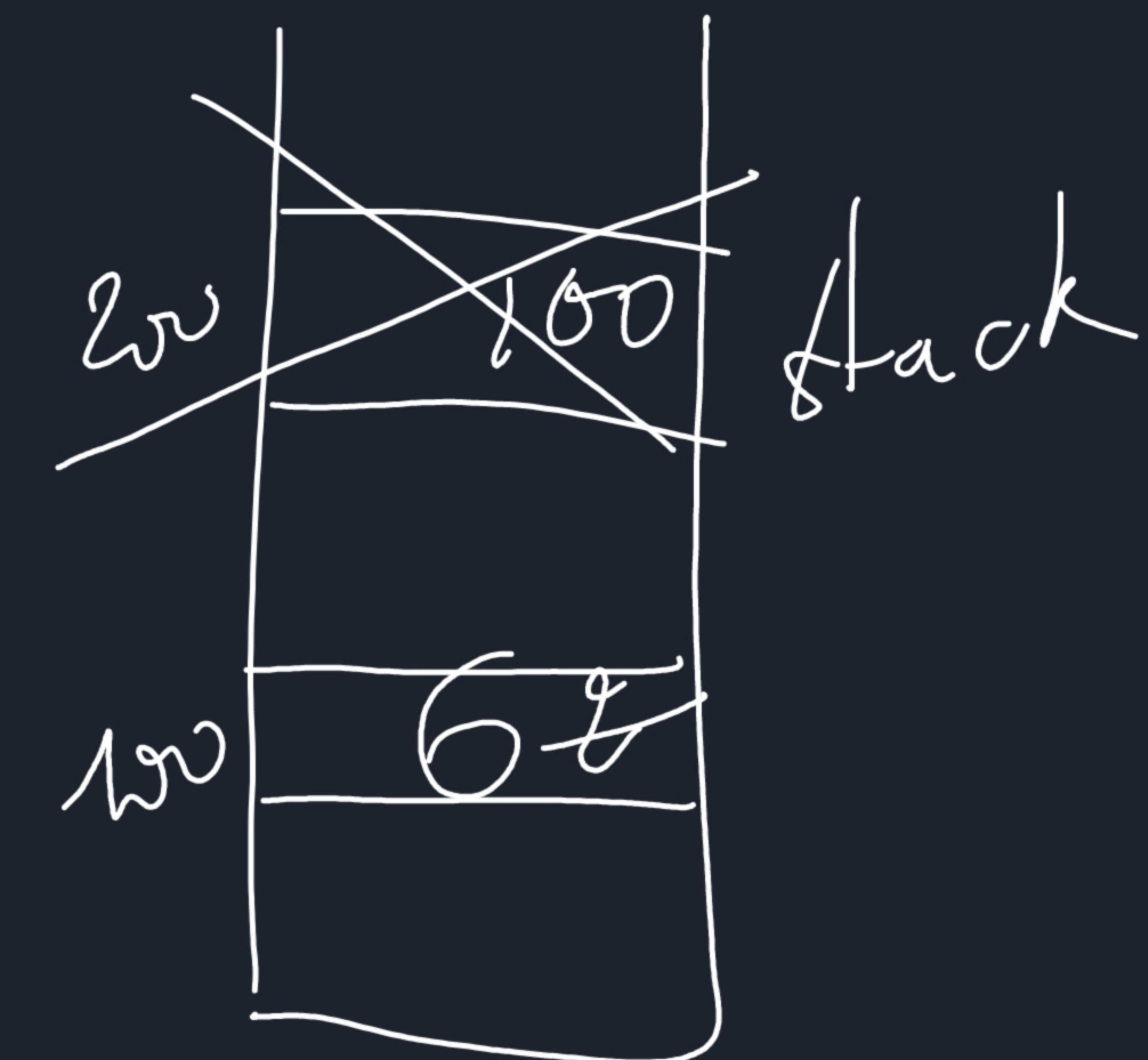
int main() {
    int i = 2;
    triple(&i);
    printf("%d\n", i);
    return 0;
}
    
```

ST	
name	a
i	100
a	200

$x = 2, y = 3;$   
 $z = \arg(x, y);$

double arg (double a, double b);

Here, a is 2 and b is 3



In C, one cannot set to a parameter (from  
inside a function) in a permanent way



## Strings

```
#include <string.h>
```

```
int strlen(char *s);
```

```
int strcpy(char *dst, char *src);
```

```
int strcmp(char *s1, char *s2);
```

→  $\begin{cases} 0 & \text{if } s_1 < s_2 \\ 1 & \text{if } s_1 > s_2 \end{cases}$

char \*s = "bonjour";  
s → [b o n | j o u r n a l]

\*s = 'B';  
s[0] = 'B';  
if ( $s_1 >= s_2$ ) ---  
if (!strcmp(s, "a")) ---

end

of

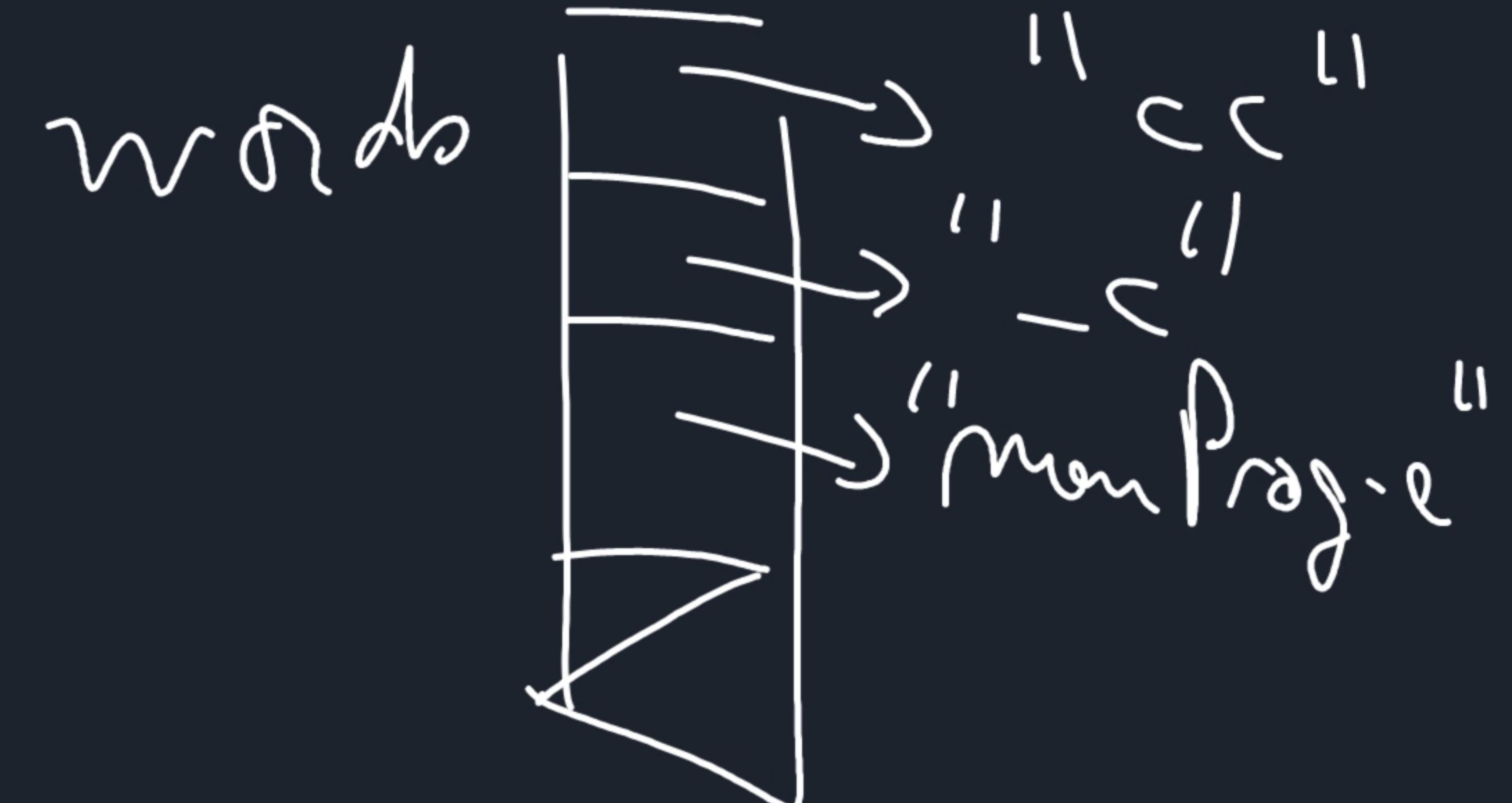
string

# Command line parameters

\$ cc -c mainProg.c

int main (int nWords, char\* words[]) { ... }

nWords | 3



HOME /home/me

atoi ("3") → 3

\$ ./sum 3 < d - 1 2

8

(\* sum.c \*)

```
#include <stdio.h>
#include <stdlib.h>
int main(int nWords, char* words[])
{
    int sum = 0;
    for (int i = 0; i < nWords; i++)
        sum += atoi(words[i]);
    printf("%d\n", sum);
    return 0;
}
```

1  
sum ---

2  
sum ---

3  
sum ---

4  
sum ---

5  
group root /etc/passwd  
6  
group route  
7  
group root /etc/password

```
res = msconfig(words[:], "lo", &cur)
} (res == 1) num += cur;
```