Pasando argumentos vía la terminal en C

6 de Abril de 2021

Esquema de la clase

Veremos cómo indicarle al programa que tome valores por fuera del mismo.

Lema: programa que funciona no se toca.

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

//argc = argument count,
//argv = argument vector

int main(int argc,char *argv[])
{
  printf("\nArgc es igual a %i\n\n",argc);
  return 1;
}
```

```
fernando@fernando-Lenovo-G475:~/Escritorio/Otros/Fisica_computacional/Clases_2021$
./programa.e 1 vi hola 8

Argc es igual a 5
```

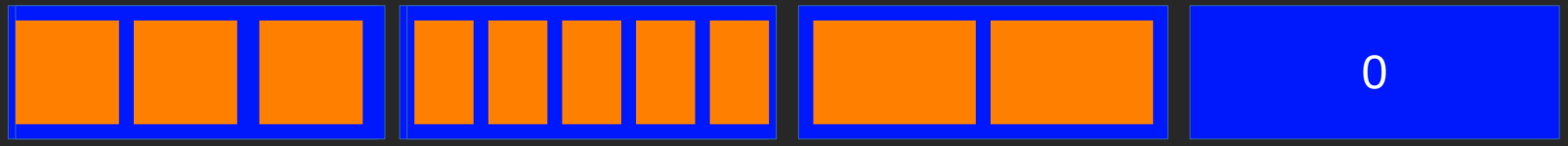
```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

//argc = argument count,
//argv = argument vector

int main(int argc,char *argv[])
{
    int i;
    for(i = 0; i<argc; i++)
        printf("\n La componente %i de argv es %s\n\n",i,argv[i]);
    return 1;
}
```

```
fernando@fernando-Lenovo-G475:~/Escritorio/Otros/Fisica_computacional/Clases_2021$
./programa.e 1 vi hola 8

 La componente 0 de argv es ./programa.e


 La componente 1 de argv es 1


 La componente 2 de argv es vi


 La componente 3 de argv es hola


 La componente 4 de argv es 8
```

¿Qué es char *argv[]?

Notemos que también lo podemos escribir como char **argv

Entonces, tenemos un puntero de otro puntero! 




```
fernando@fernando-Lenovo-G475:~/Escritorio/Otros/Fisica_computacional/Clases_2021$
./programa.e 1 vi hola 8

Argc es igual a 5
```

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

//argc = argument count,
//argv = argument vector

int main(int argc,char *argv[])
{
    int n;
    float p_1,p_2;

    if (argc==4)
    {
        sscanf(argv[1],"%d",&n);
        sscanf(argv[2],"%f",&p_1);
        sscanf(argv[3],"%f",&p_2);
    }
    printf("n es igual a %i\n",n);
    printf("p1 es igual a %f\n",p_1);
    printf("p2 es igual a %f\n",p_2);
    return 1;
```

```
fernando@fernando-Lenovo-G475:~/Escritorio/Otros/Fisica_computacional/Clases_2021$
./programa.e 8 0.4 0.6
n es igual a 8
p1 es igual a 0.400000
p2 es igual a 0.600000
```

Resumiendo:

Si tenemos el “esqueleto” de nuestro código que funciona bien pero necesitamos modificar algunas variables “cada tanto”, es conveniente pasarlas como argumento. De ésta forma evitamos modificarlo constantemente, y de paso, ahorramos tener que compilarlo cada vez que modificamos las variables que nos interesan.

Extras:

- No conviene usar malloc muchas veces a lo largo del código. Sólo úsenlo si saben que el vector va a cambiar de tamaño, sino usen memoria estática (vec[10]).

- for(i=0;i<n*n;i++) no es aconsejable porque en ese caso estamos calculando en cada paso n*n. Conviene definir una variable “f” igual a n*n y trabajar con for(i=0;i<f;i++).

Extras:

- Conviene que en la función main() haya subfunciones y no código de cálculo. De esta forma, a simple vista podemos “entender” qué hace el programa. Ej:

```
Main()
{
        lleno_red();
        Hoshen();
        percola();
        Calculo_ns();
        .
        .
}
```