INFORMATICA I

Serie del número de Euler en "C"

Ing.Juan Carlos Cuttitta

Universidad Tecnológica Nacional
Facultad Regional Buenos Aires
Departamento de Ingeniería Electrónica

4 de mayo de 2020

Enunciado del problema

El valor aproximado del número de Euler (e) se puede obtener con la siguente serie.

$$e = \sum_{n=0}^{\infty} \frac{1}{!n}$$

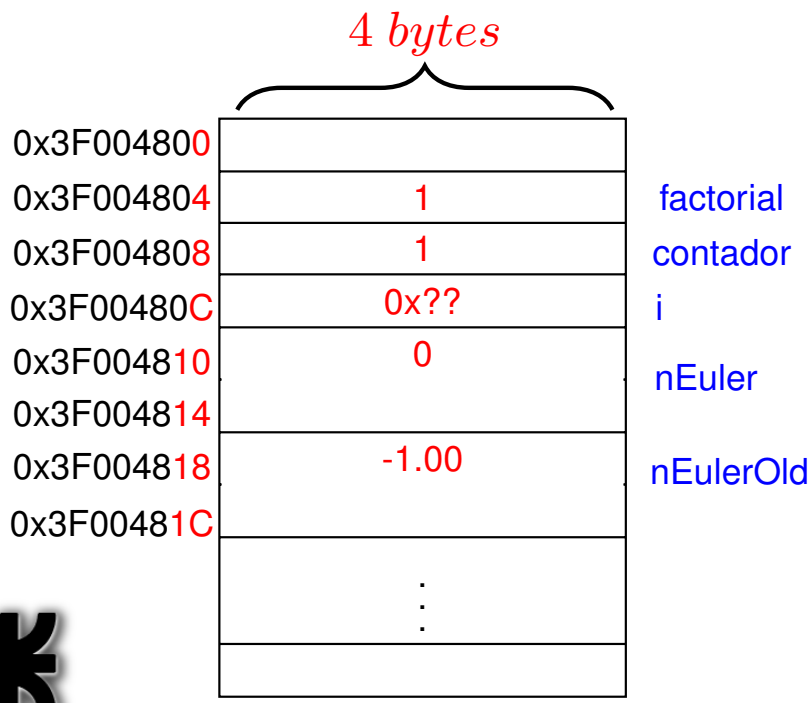
$$e = \frac{1}{!0} + \frac{1}{!1} + \frac{1}{!2} + \frac{1}{!3} + \frac{1}{!4} + \frac{1}{!5} + \cdots$$

Escribir un programa que calcule el valor aproximado de e mediante un ciclo repetitivo que termine cuando la diferencia entre dos aproximaciones sucesivas difiera en menos de 10^{-9}

Declaración y disposición en memoria

Arquitectura X86-32 bits

Disposición de las variables en la memoria



Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Declaración y disposición en memoria

Arquitectura X86-32 bits

Declara la variable i

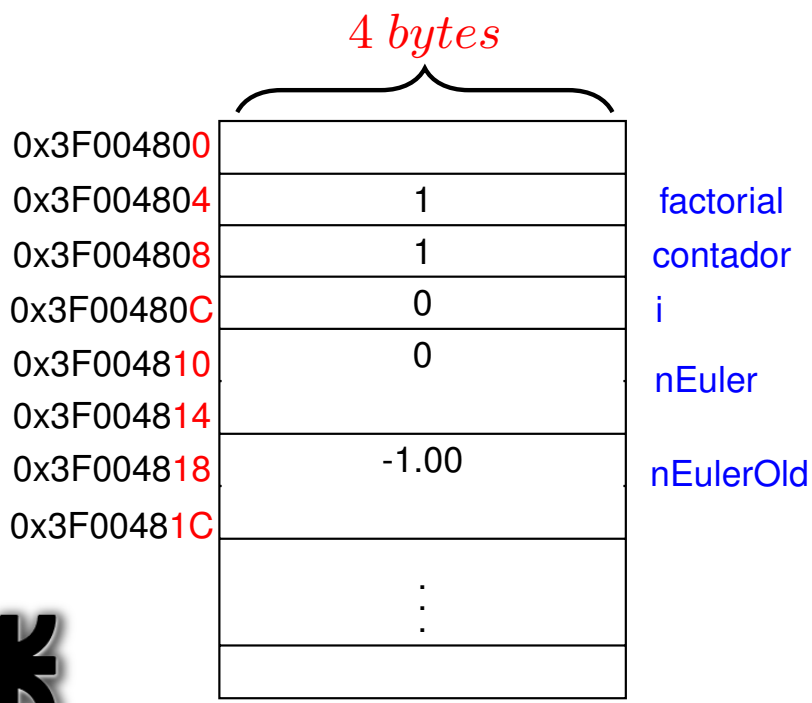
Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

Pregunta condición del for



Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

10^{-9}

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

Para conservar el e anterior

Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

Como contador no es <= a i no entra dentro del while

Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

Se debe castear la variable para que ambas sean double

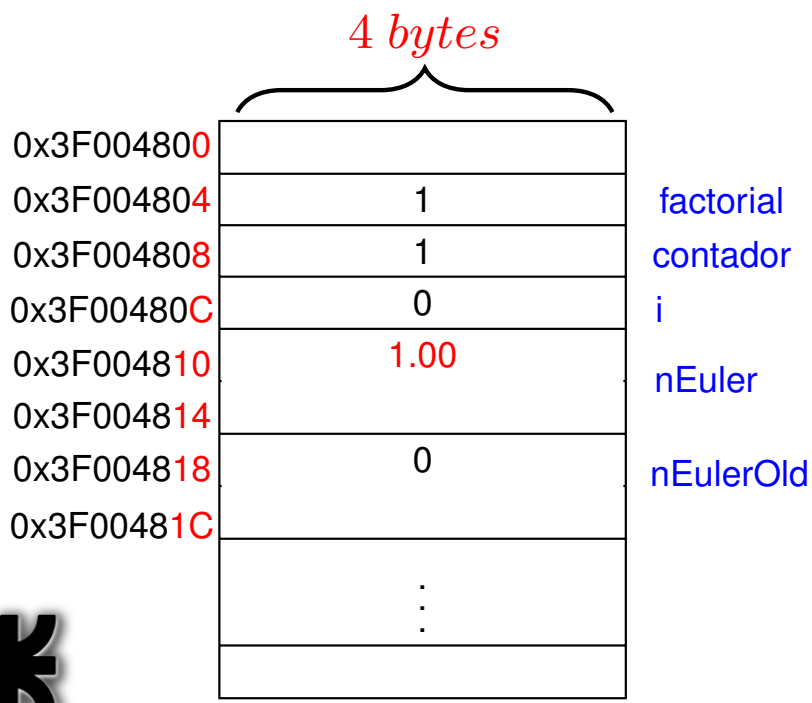
Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

almacena el resultado en la variable nEuler



Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

$0 + \frac{1}{1,00} = 1,00$

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

inicializo nuevamente las variables factorial y contador

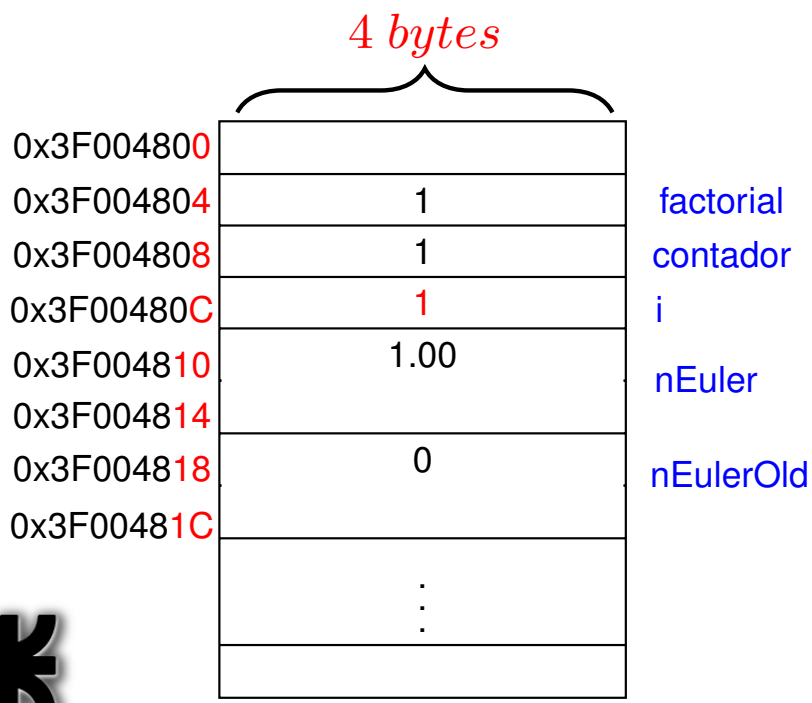
Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

incremento la variable i



Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

Pregunta condición del for

Código del programa fuente

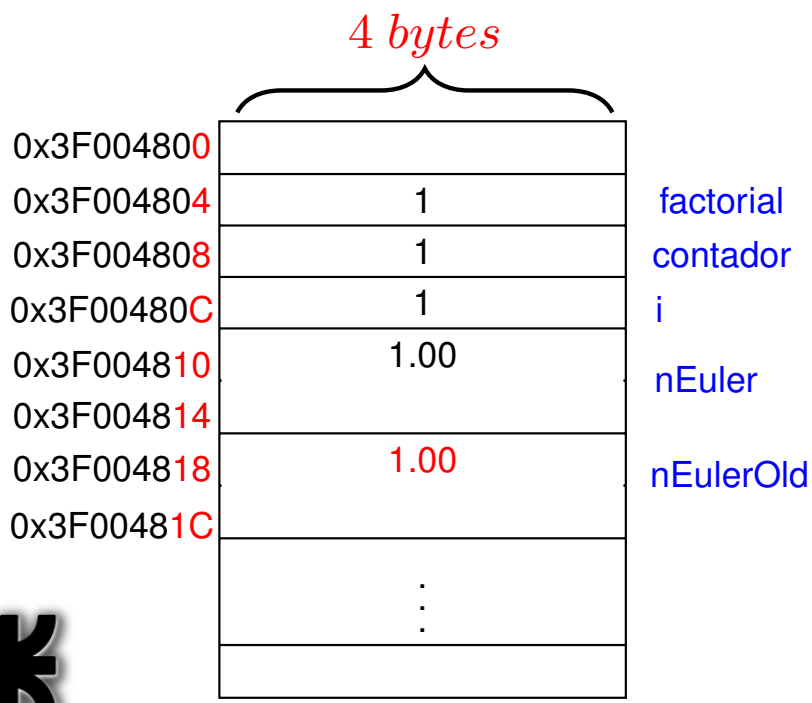
```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

10^{-9}

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

Para conservar el e anterior



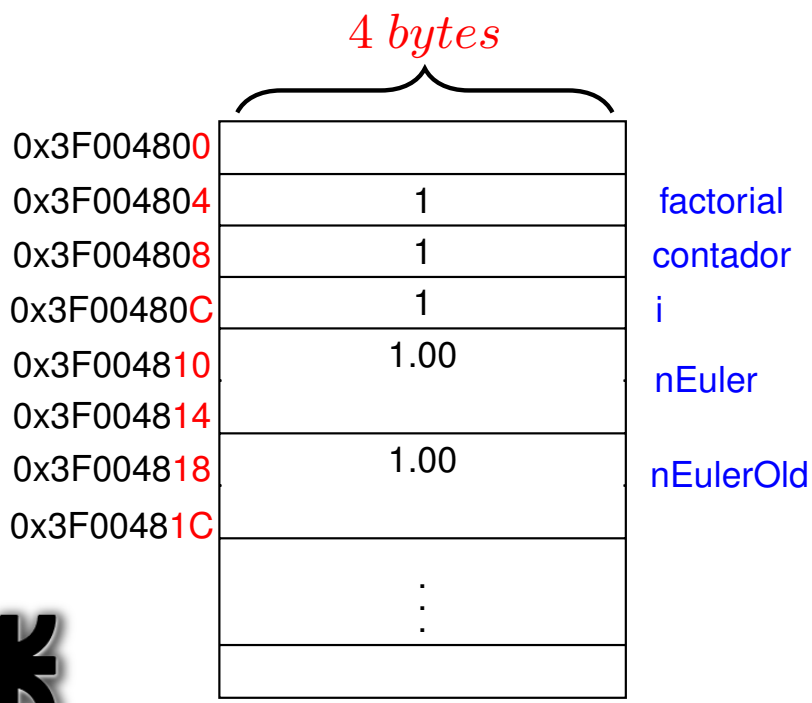
Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

Como contador es <= a i entra dentro del while



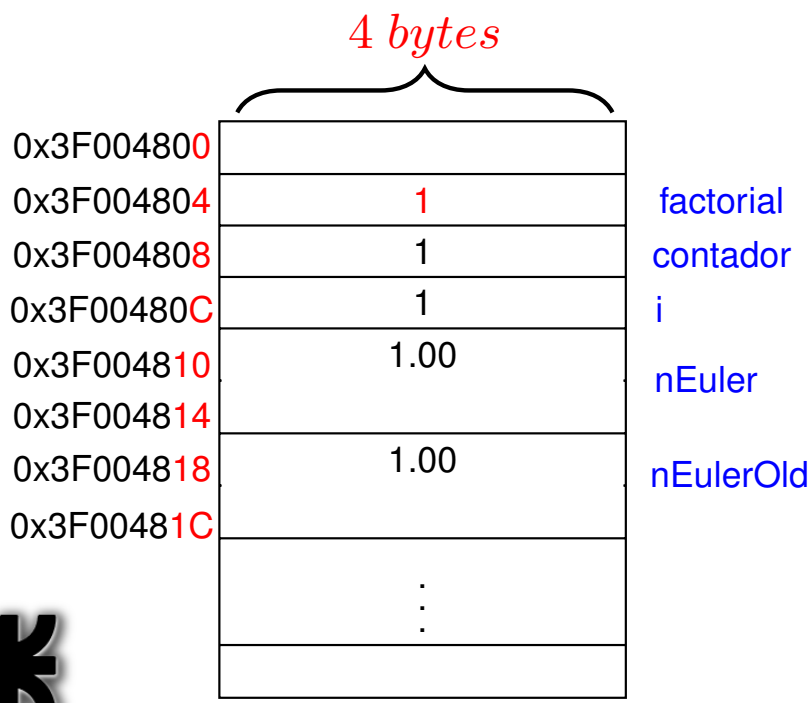
Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

almacena el resultado en la variable factorial



Código del programa fuente

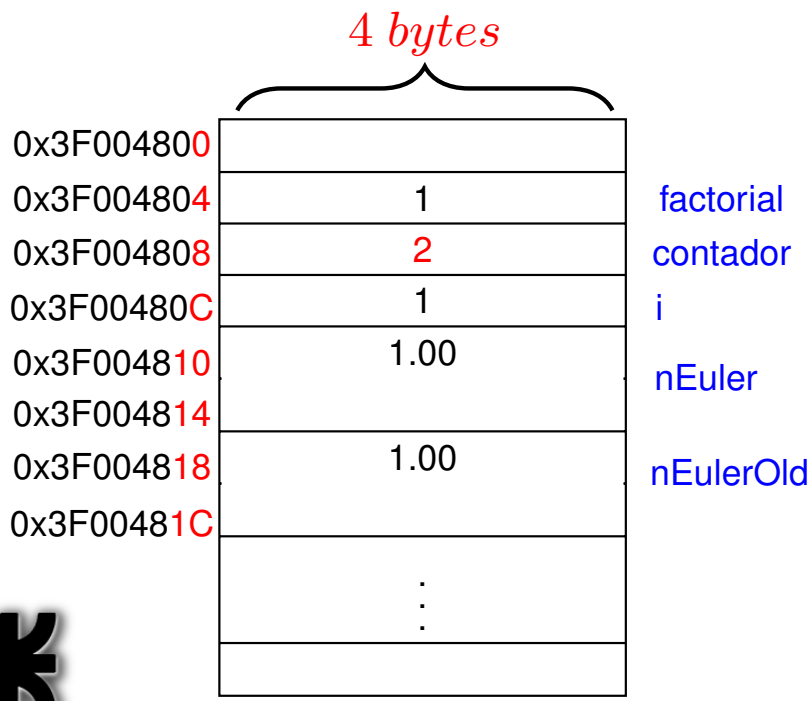
```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

$1 \cdot 1 = 1$

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

incremento la variable contador



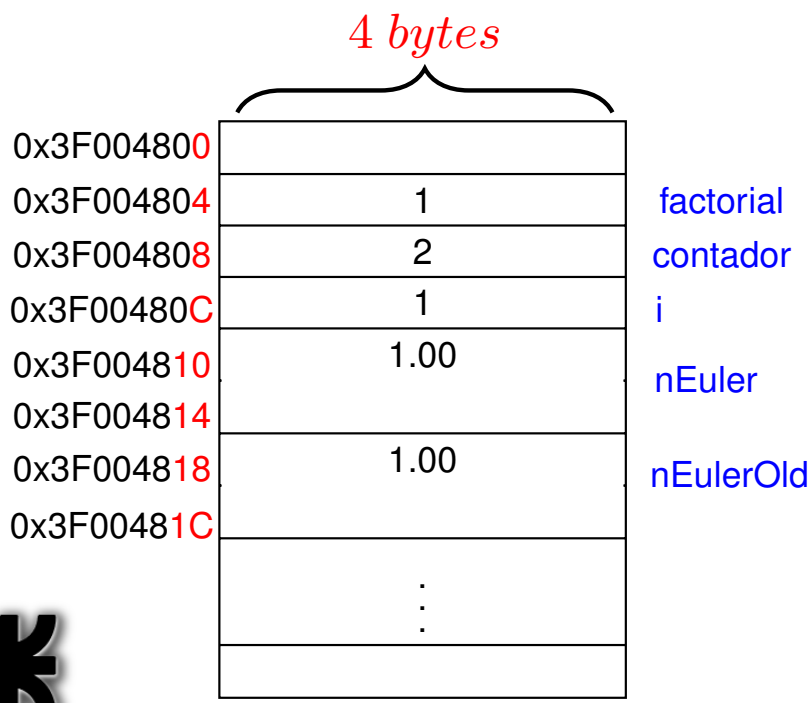
Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16       contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

Como contador no es <= a i sale del loop while



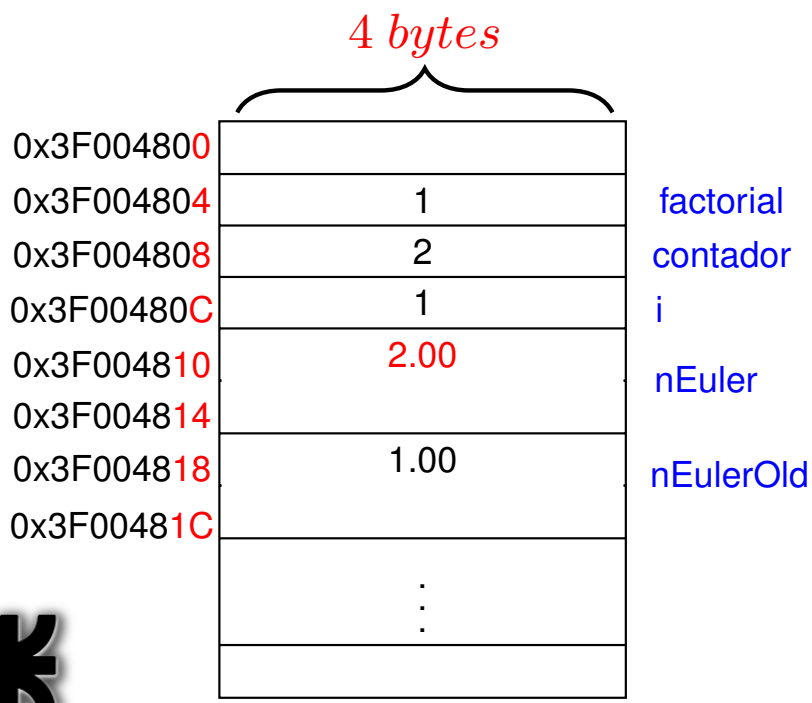
Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

almacena el resultado en la variable nEuler



Código del programa fuente

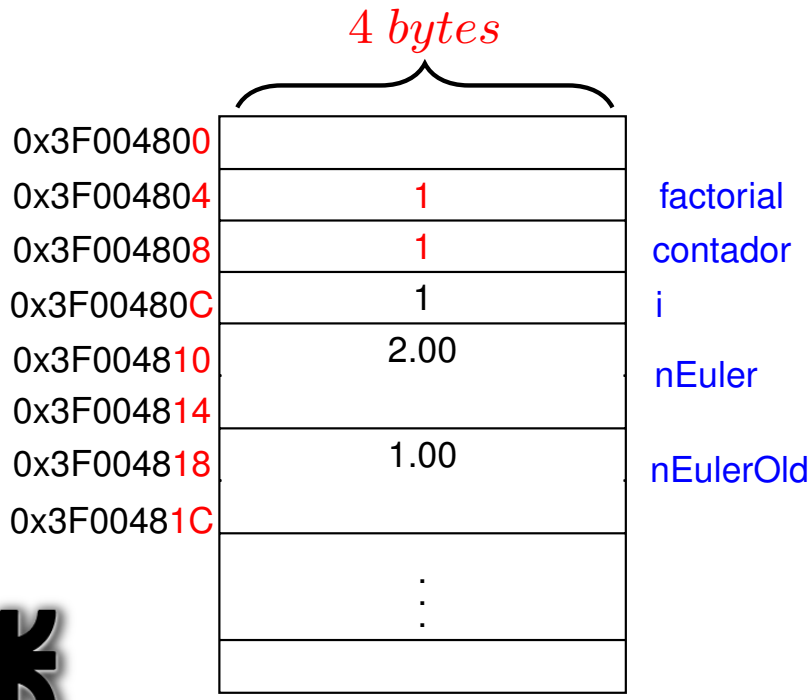
```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

$1{,}00 + \frac{1}{1{,}00} = 2{,}00$

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

inicializo nuevamente las variables factorial y contador



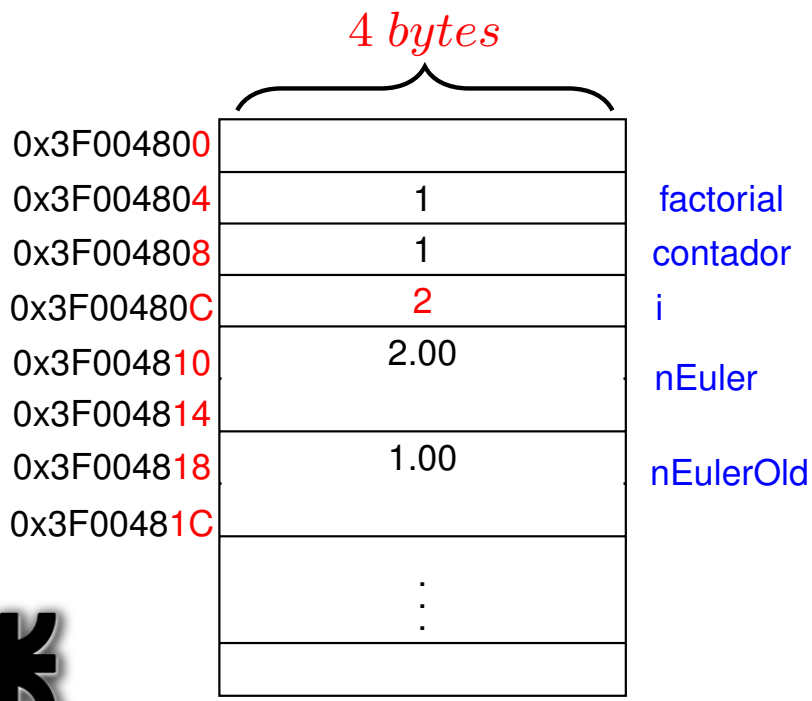
Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

incremento la variable i



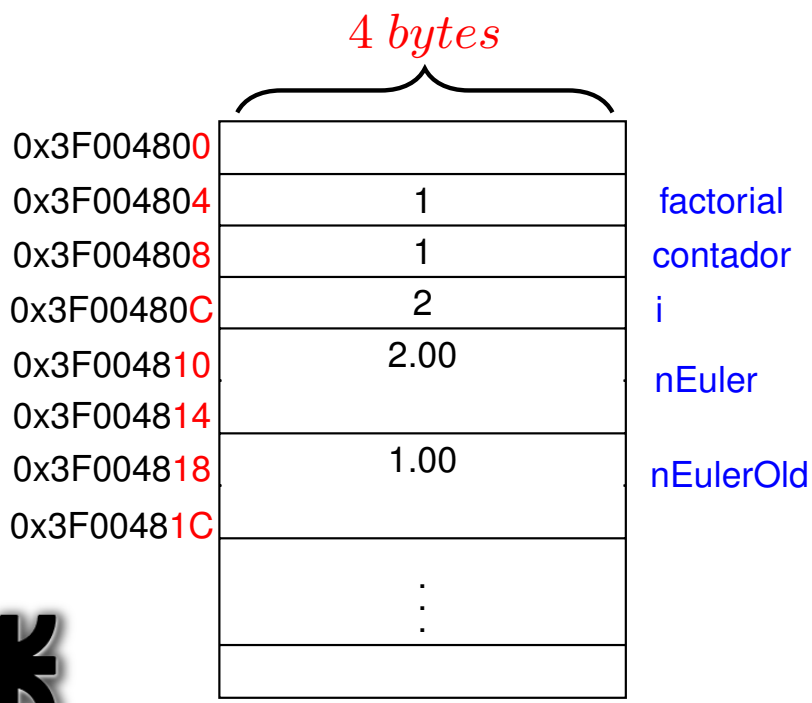
Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

Pregunta condición del for



Código del programa fuente

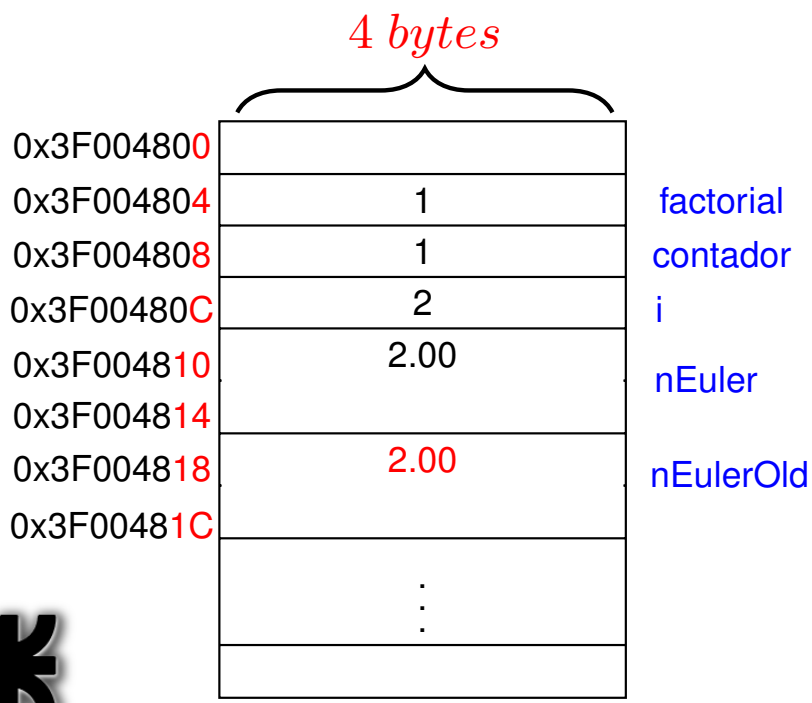
```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

10^{-9}

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

Para conservar el e anterior



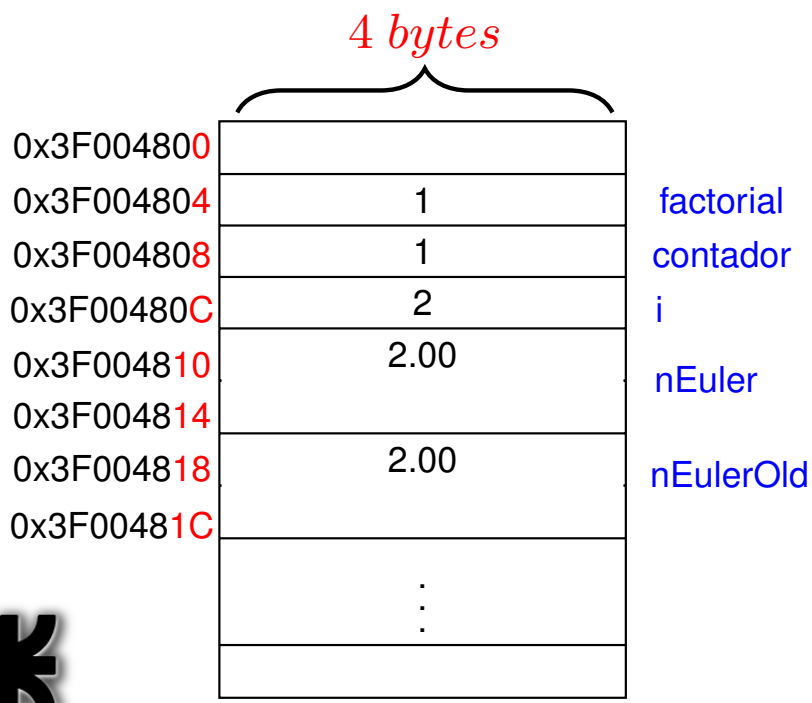
Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

Como contador es <= a i entra dentro del while



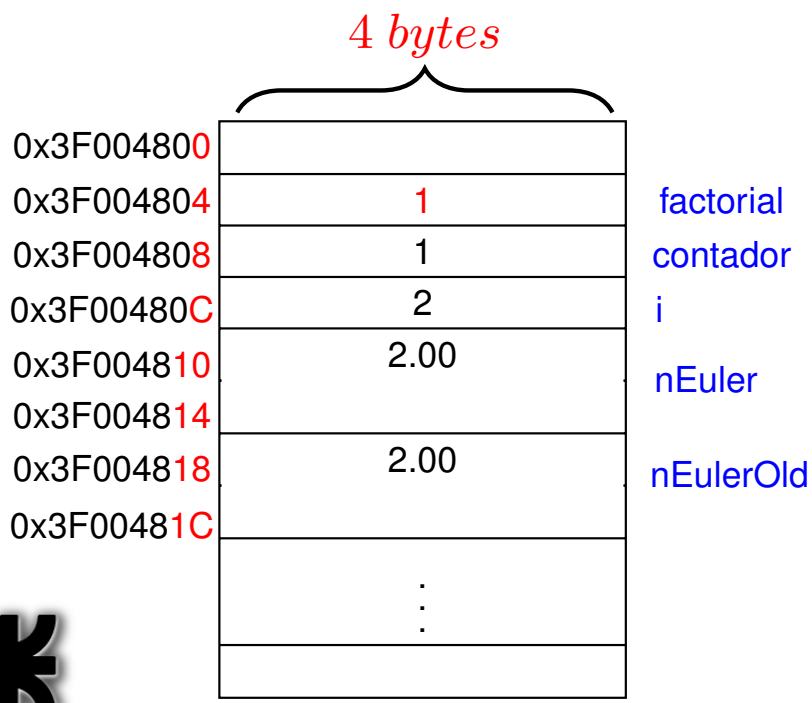
Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

almacena el resultado en la variable factorial



Código del programa fuente

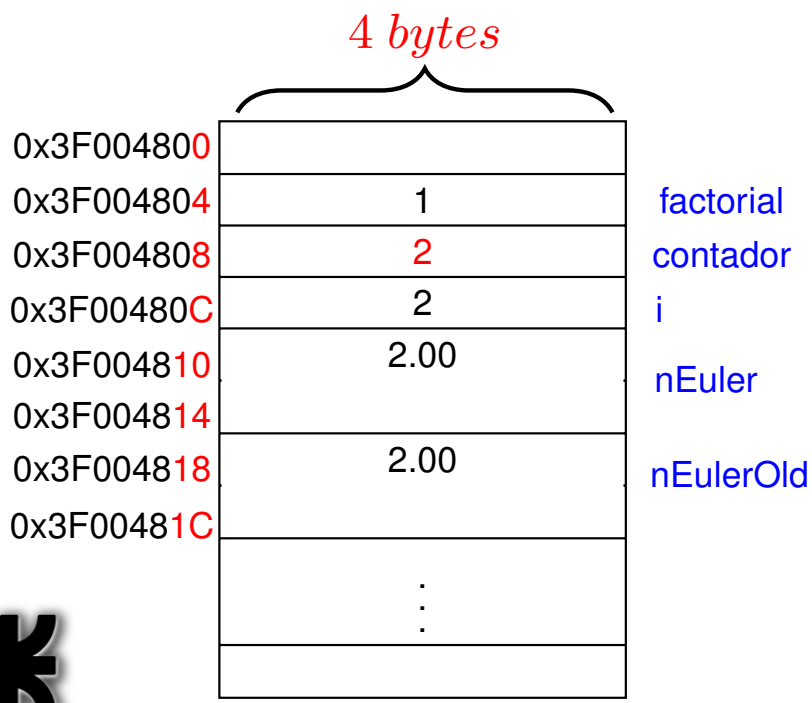
```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

$1 \cdot 1 = 1$

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

incremento la variable contador



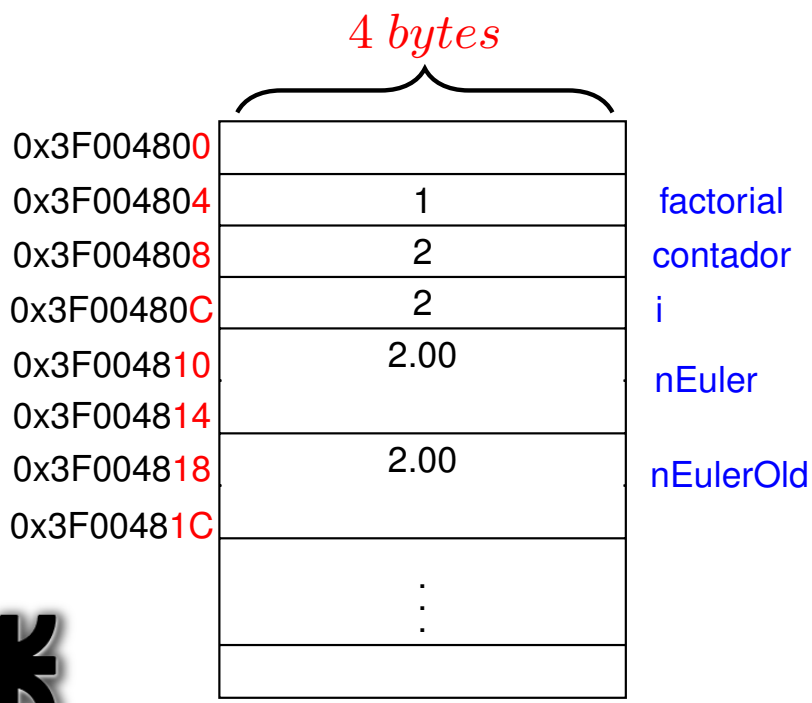
Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

Como contador es <= a i entra dentro del while



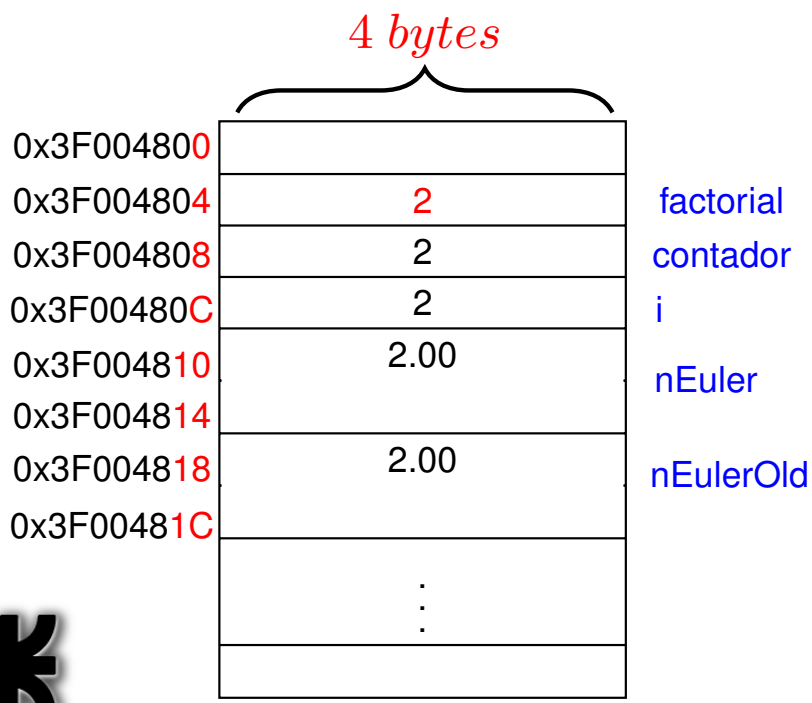
Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

almacena el resultado en la variable factorial



Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

$1 \cdot 2 = 2$

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

incremento la variable contador

Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

Como contador no es <= a i sale del loop while

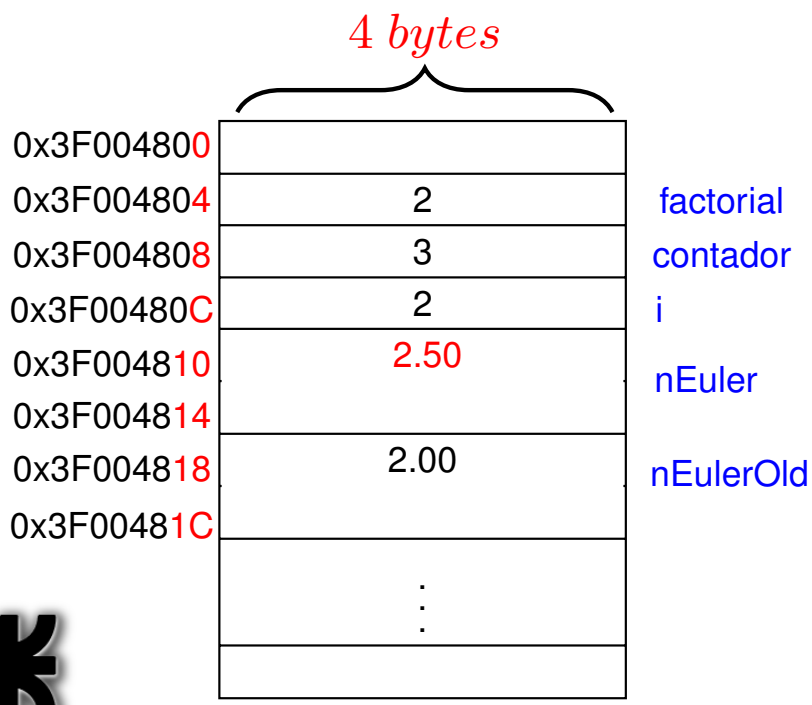
Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

almacena el resultado en la variable nEuler



Código del programa fuente

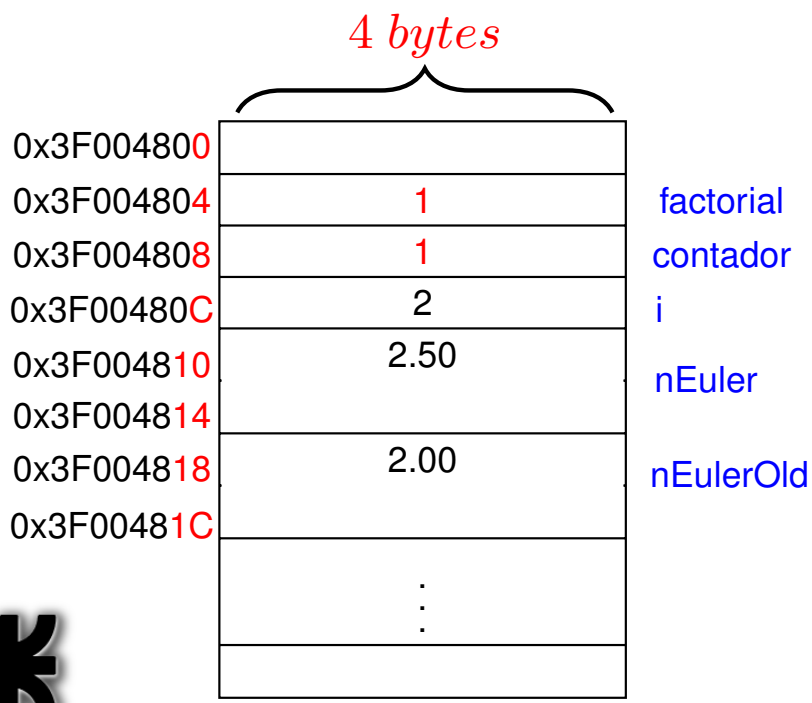
```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

$2{,}00 + \frac{1}{2{,}00} = 2{,}50$

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

inicializo nuevamente las variables factorial y contador



Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

incremento la variable i

Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

Pregunta condición del for

Código del programa fuente

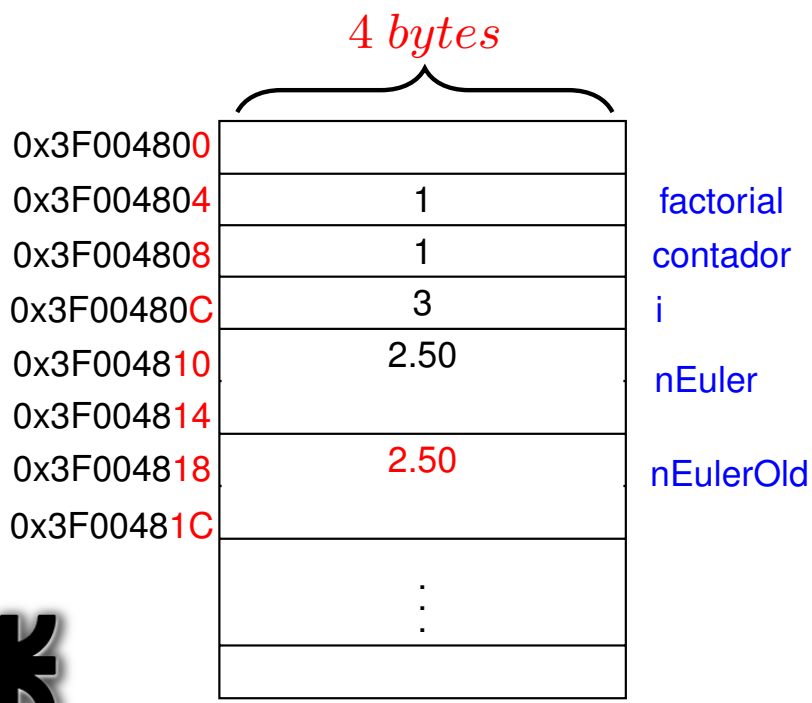
```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

10^{-9}

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

Para conservar el e anterior



Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

Como contador es <= a i entra dentro del while

Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

almacena el resultado en la variable factorial

Código del programa fuente

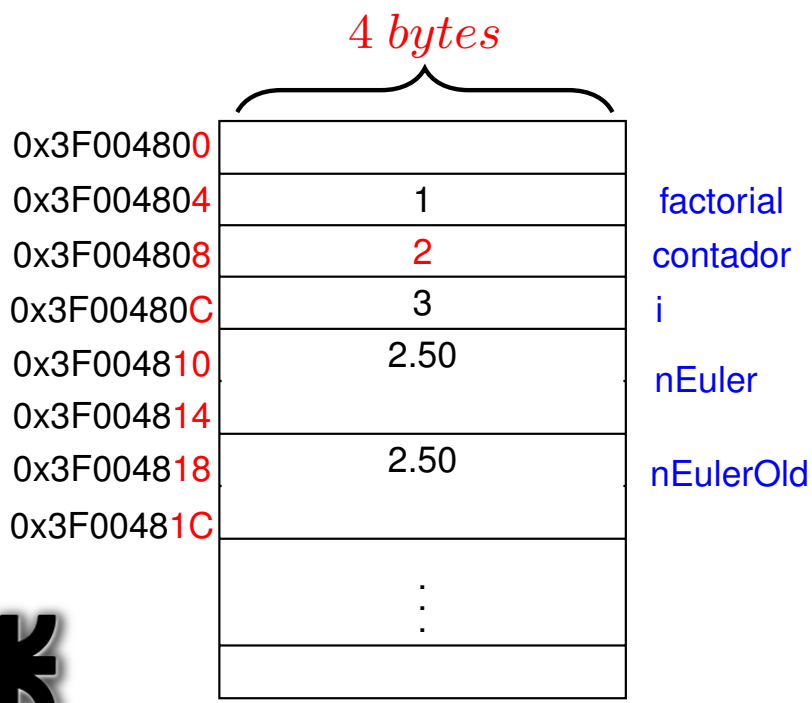
```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

$1 \cdot 1 = 1$

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

incremento la variable contador



Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

Como contador es <= a i entra dentro del while

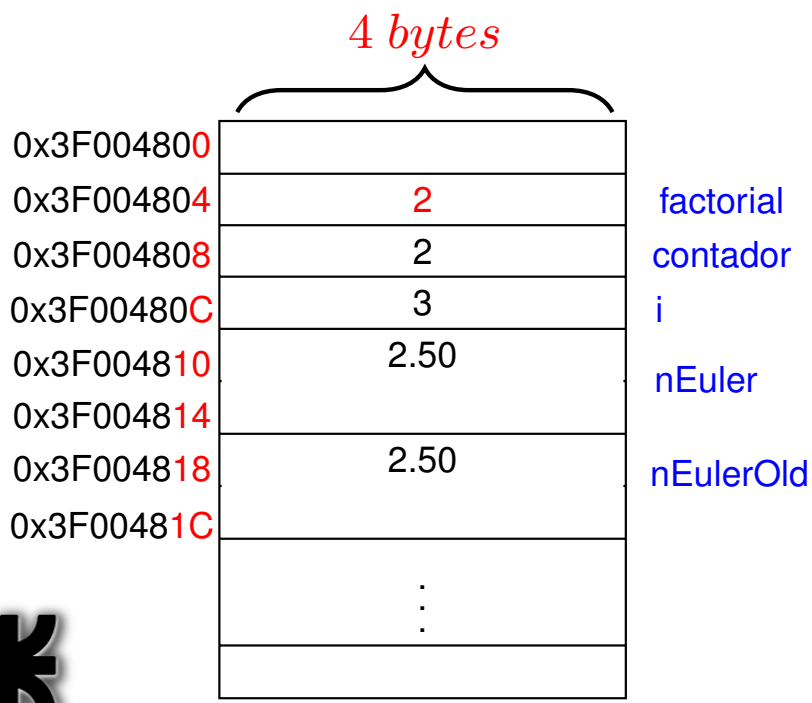
Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

almacena el resultado en la variable factorial



Código del programa fuente

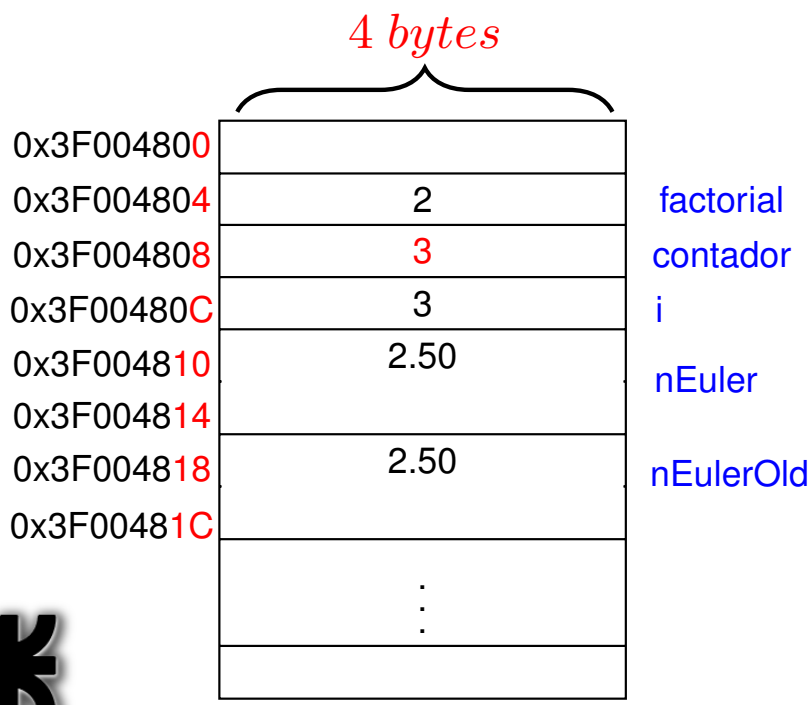
```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

$1 \cdot 2 = 2$

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

incremento la variable contador



Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

Como contador es <= a i entra dentro del while

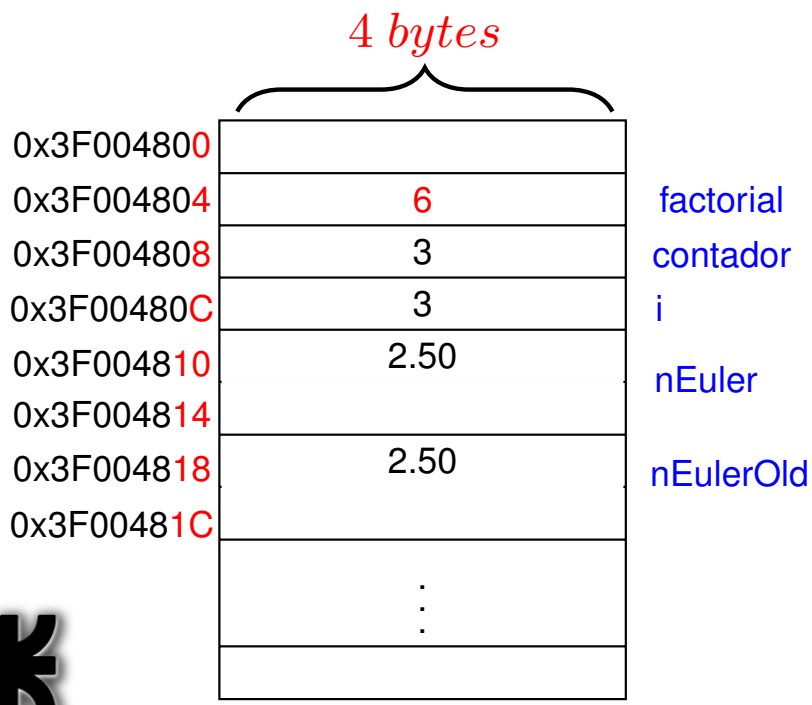
Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

almacena el resultado en la variable factorial



Código del programa fuente

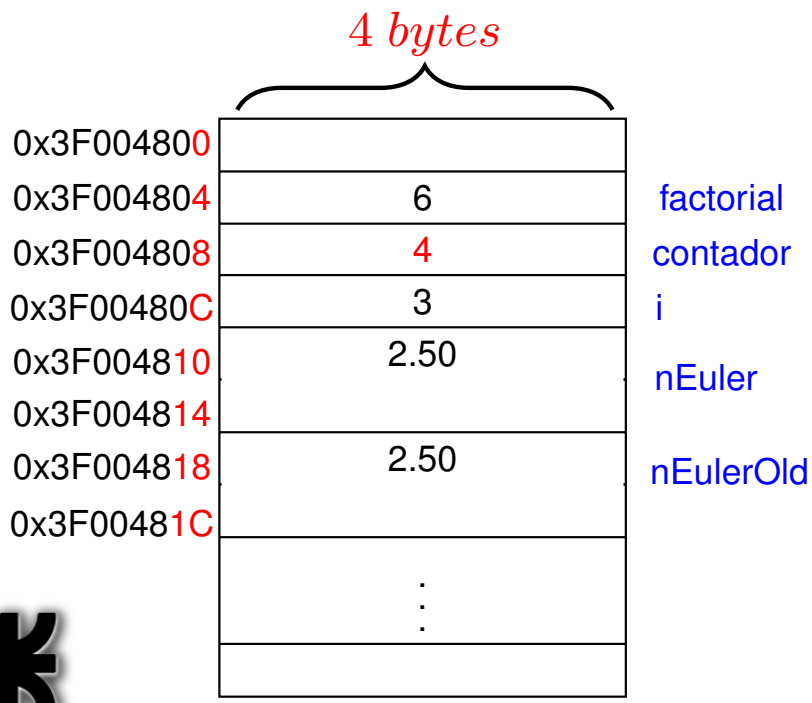
```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

$2 \cdot 3 = 6$

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

incremento la variable contador



Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

Como contador no es <= a i sale del loop while

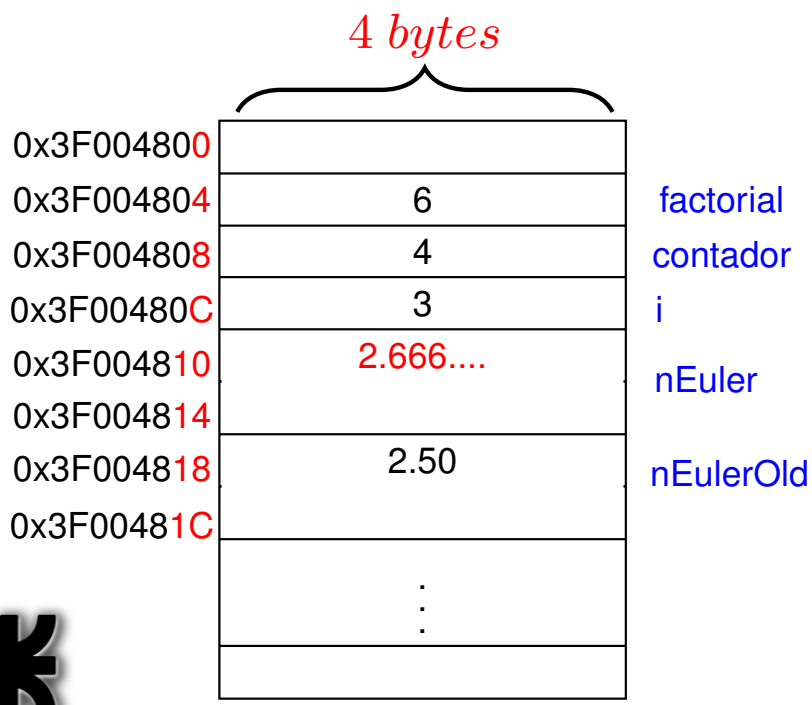
Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

almacena el resultado en la variable nEuler



Código del programa fuente

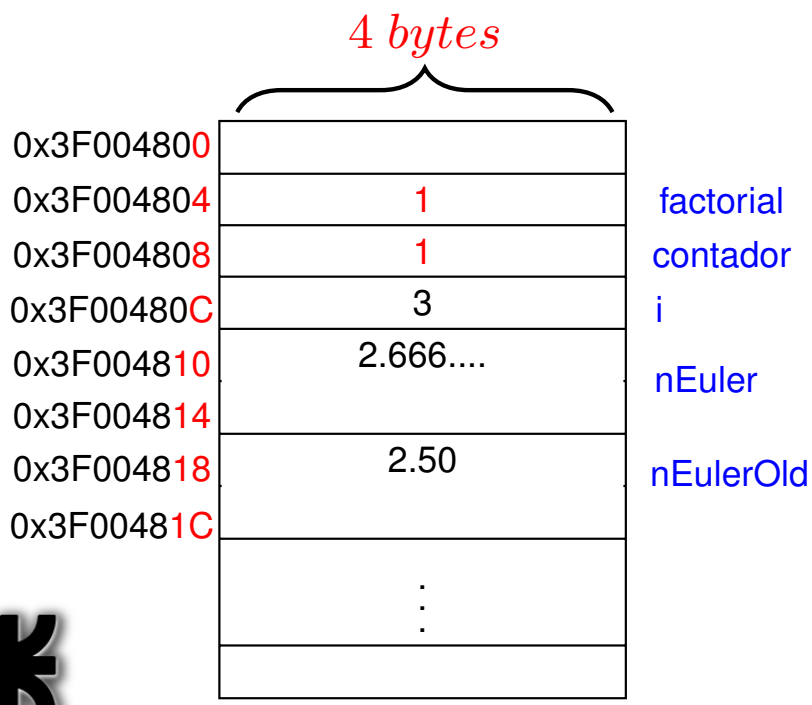
```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

$2{,}50 + \frac{1}{6{,}00} = 2{,}66\widehat{6}$

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

inicializo nuevamente las variables factorial y contador



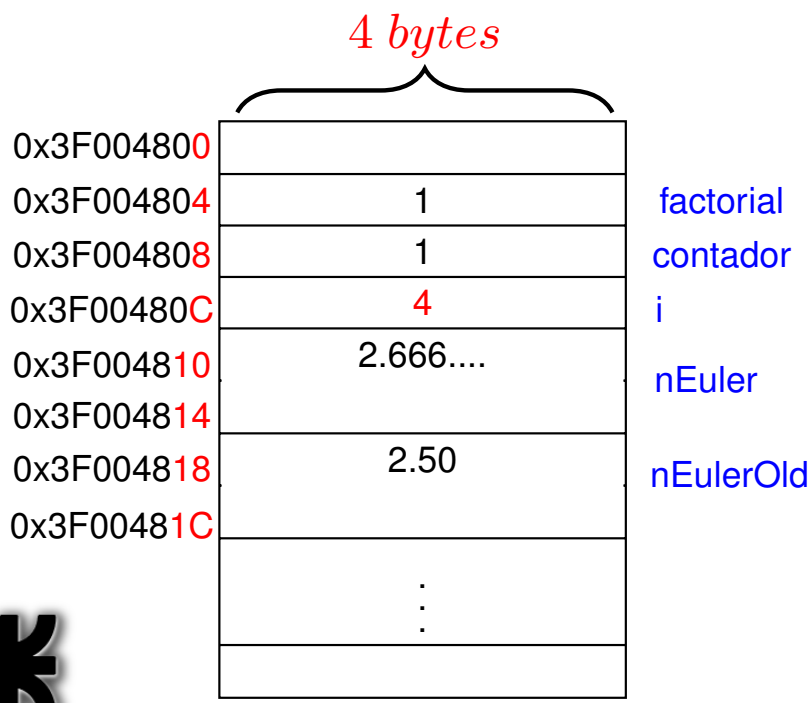
Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

incremento la variable i



Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

Pregunta condición del for

Código del programa fuente

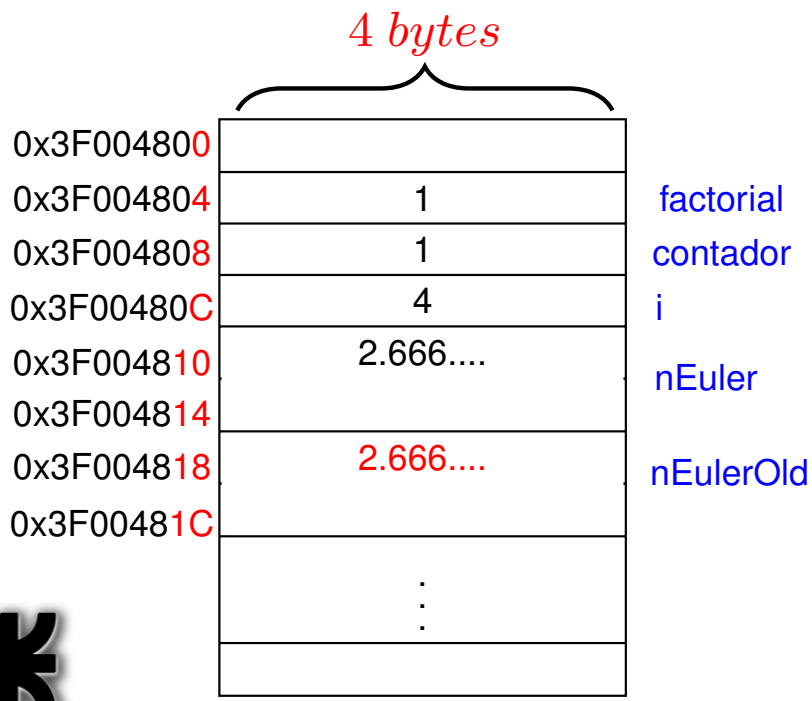
```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

10^{-9}

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

Para conservar el e anterior



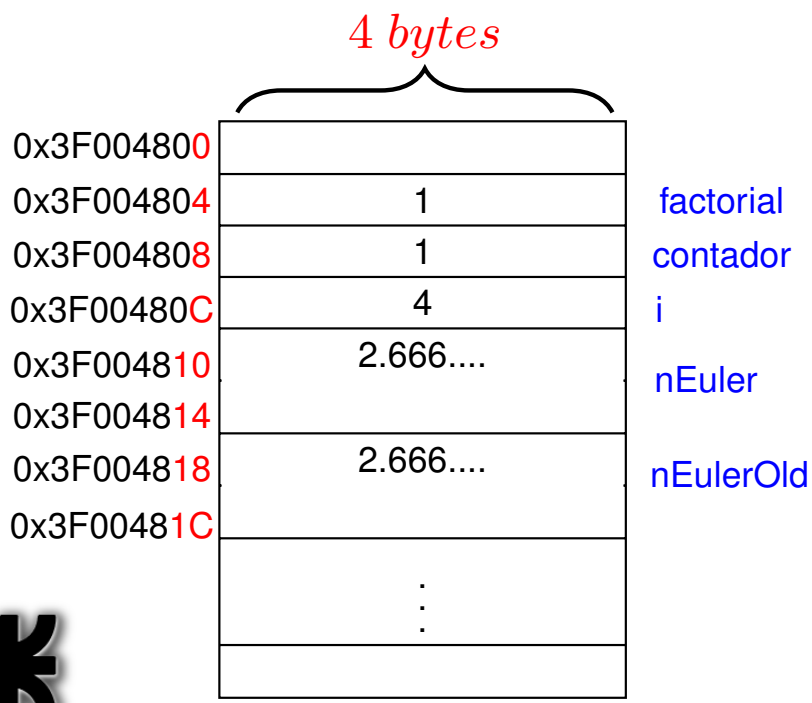
Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

Como contador es <= a i entra dentro del while



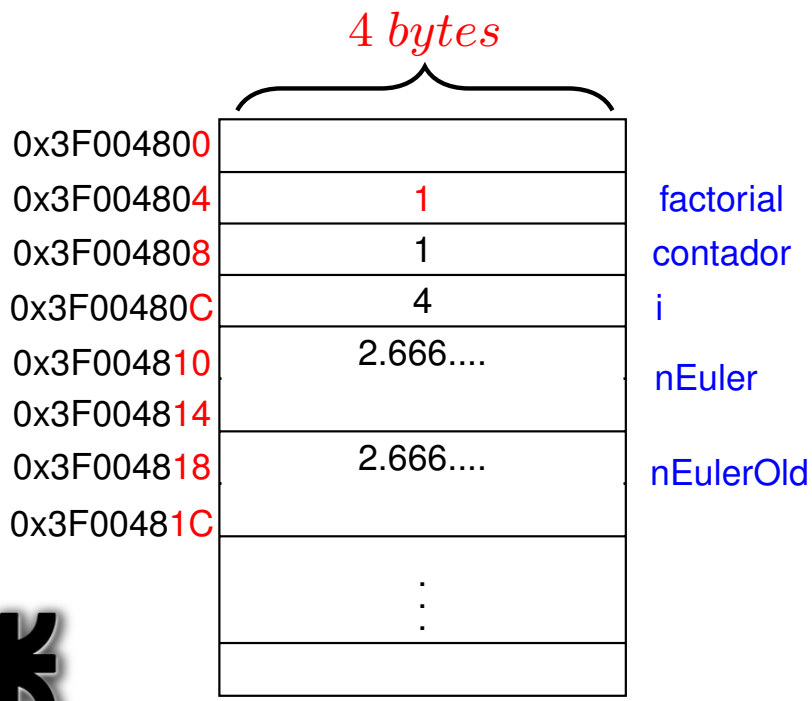
Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

almacena el resultado en la variable factorial



Código del programa fuente

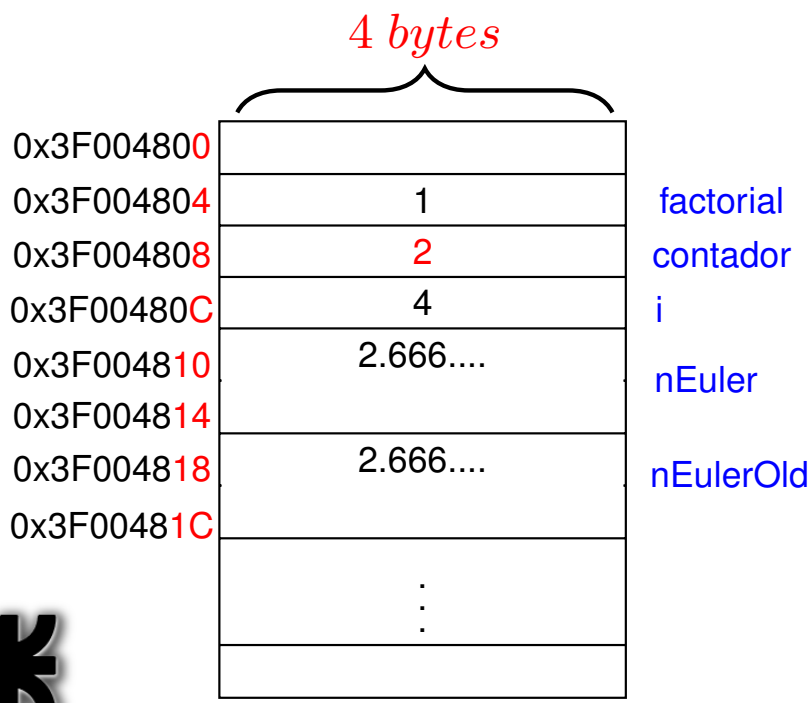
```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

$1 \cdot 1 = 1$

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

incremento la variable contador



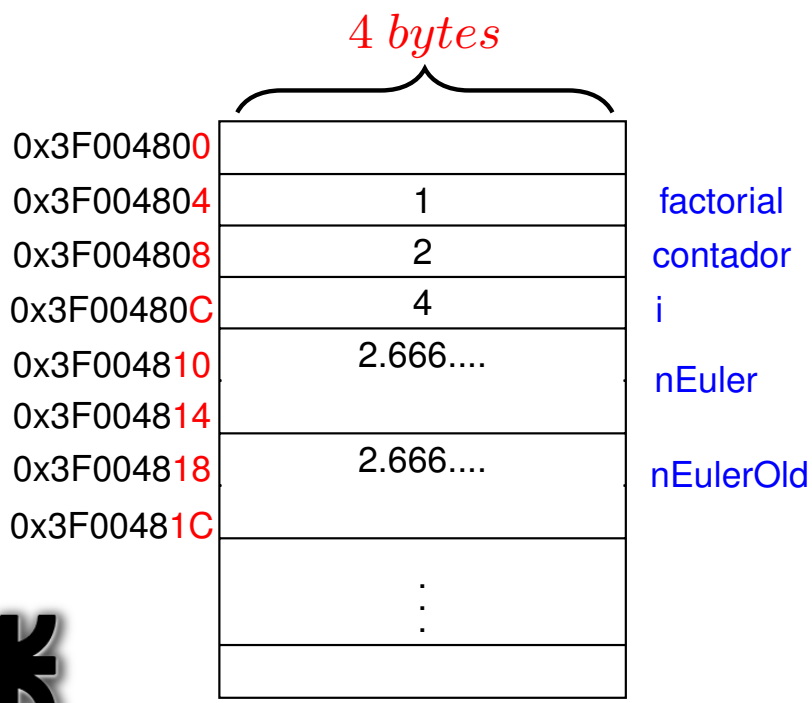
Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

Como contador es <= a i entra dentro del while



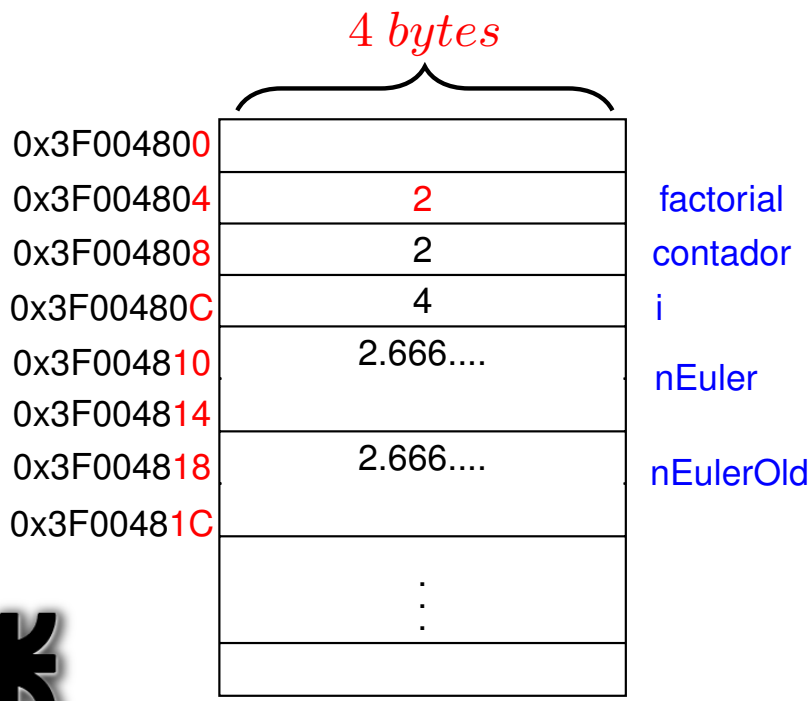
Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

almacena el resultado en la variable factorial



Código del programa fuente

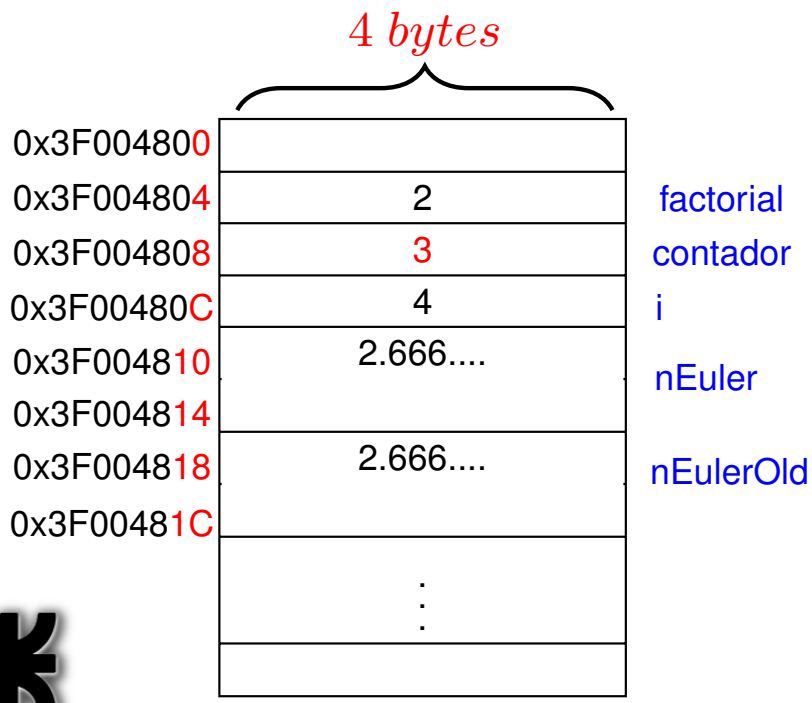
```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

$1 \cdot 2 = 2$

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

incremento la variable contador



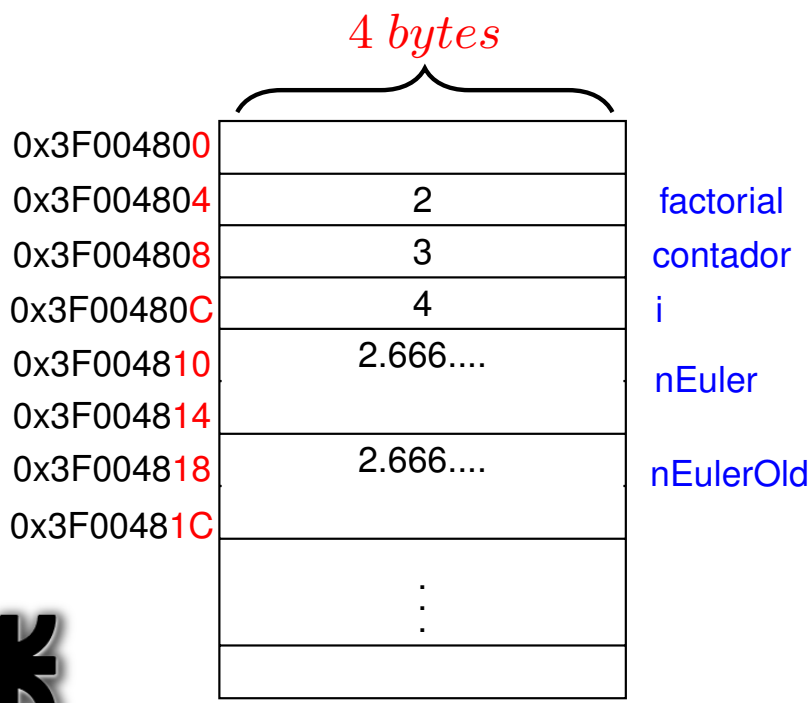
Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

Como contador es <= a i entra dentro del while



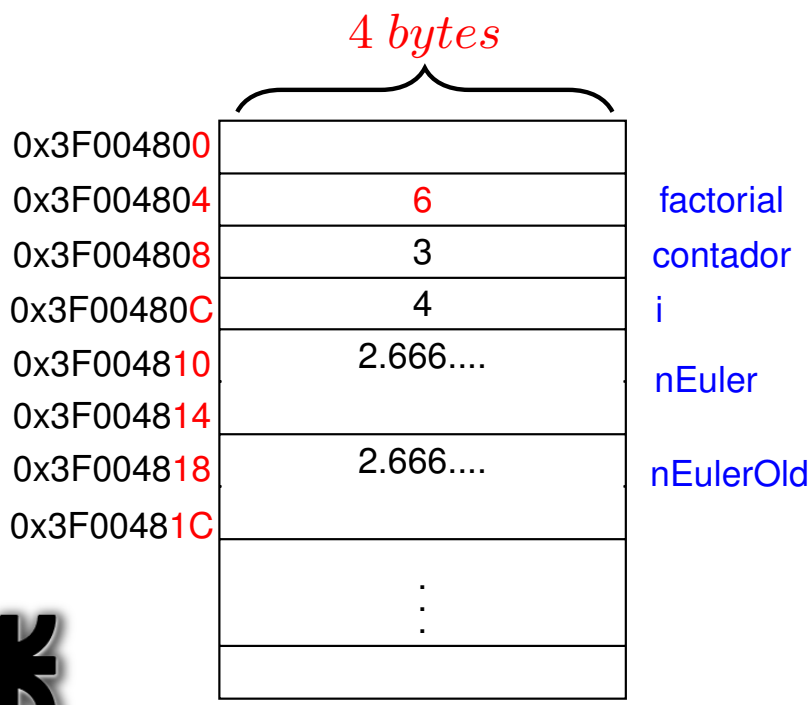
Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

almacena el resultado en la variable factorial



Código del programa fuente

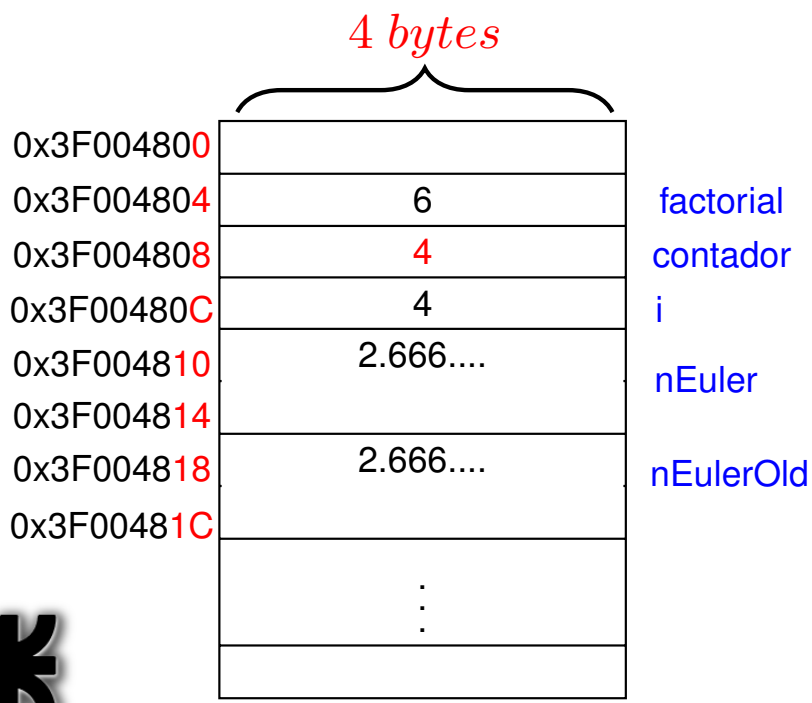
```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

$2 \cdot 3 = 6$

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

incremento la variable contador



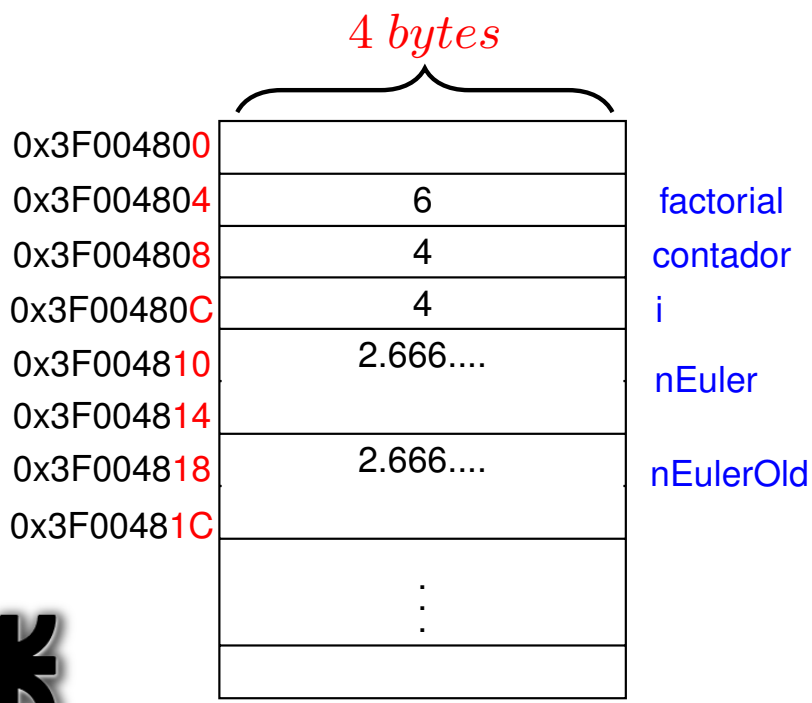
Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

Como contador es <= a i entra dentro del while



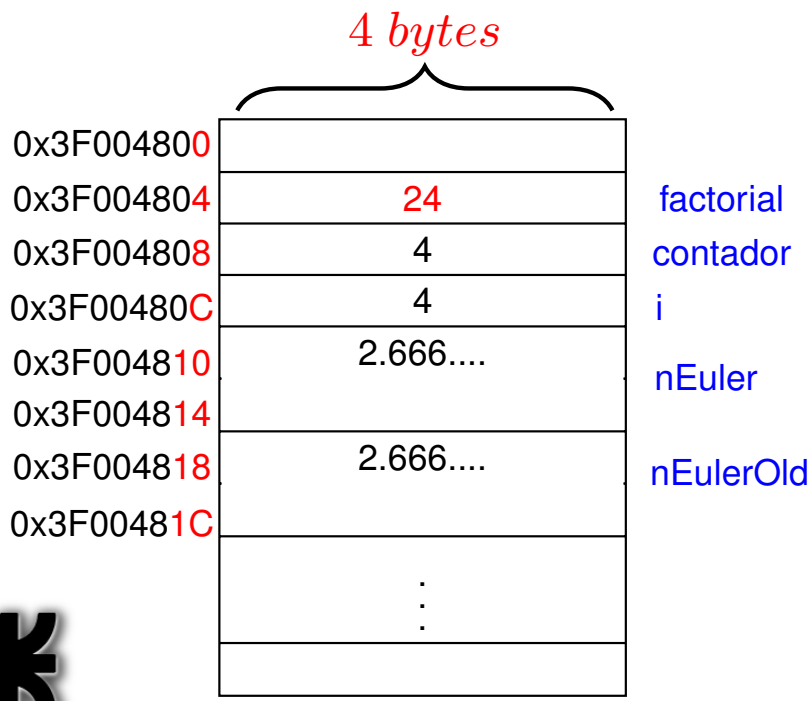
Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

almacena el resultado en la variable factorial



Código del programa fuente

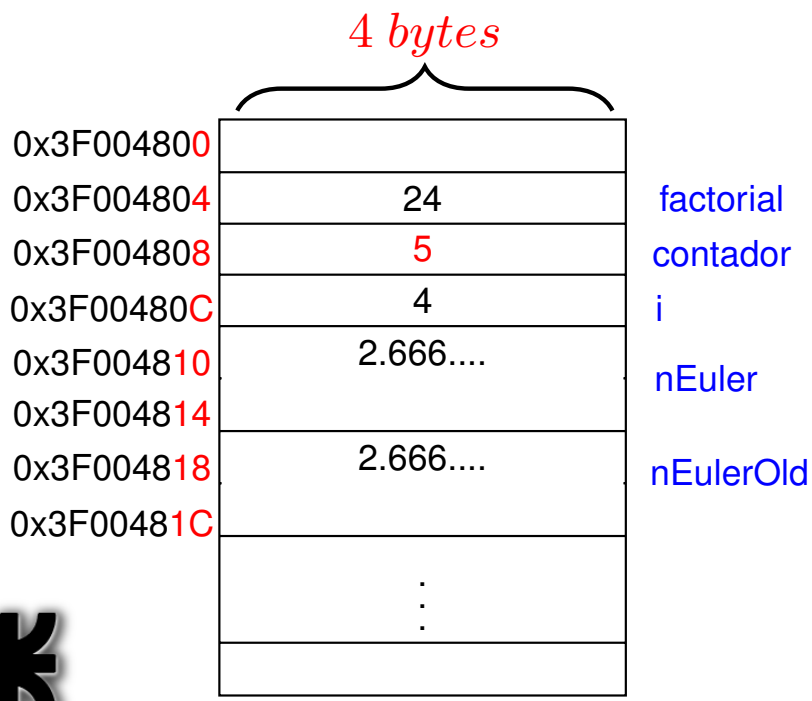
```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

$6 \cdot 4 = 24$

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

incremento la variable contador



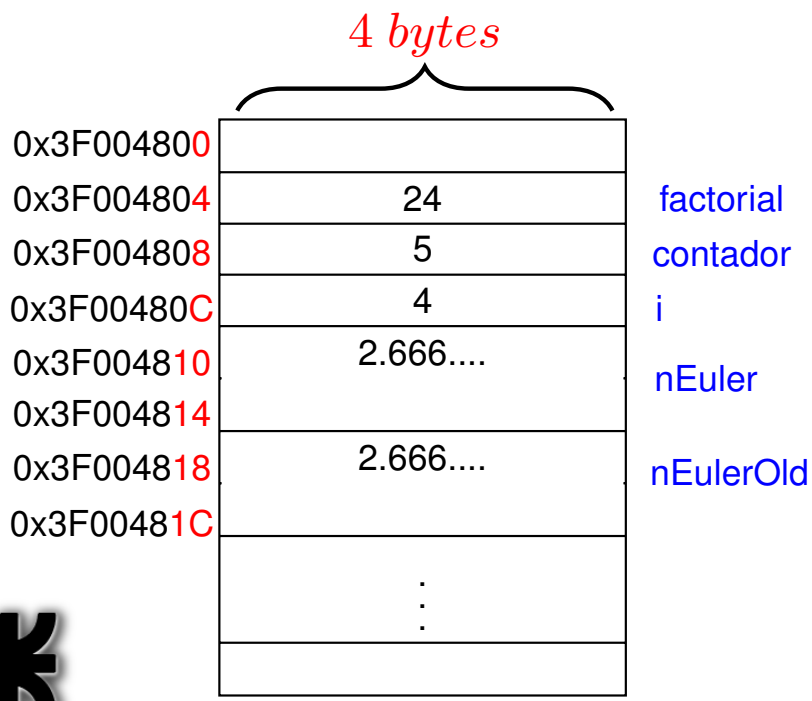
Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

Como contador no es <= a i sale del loop while



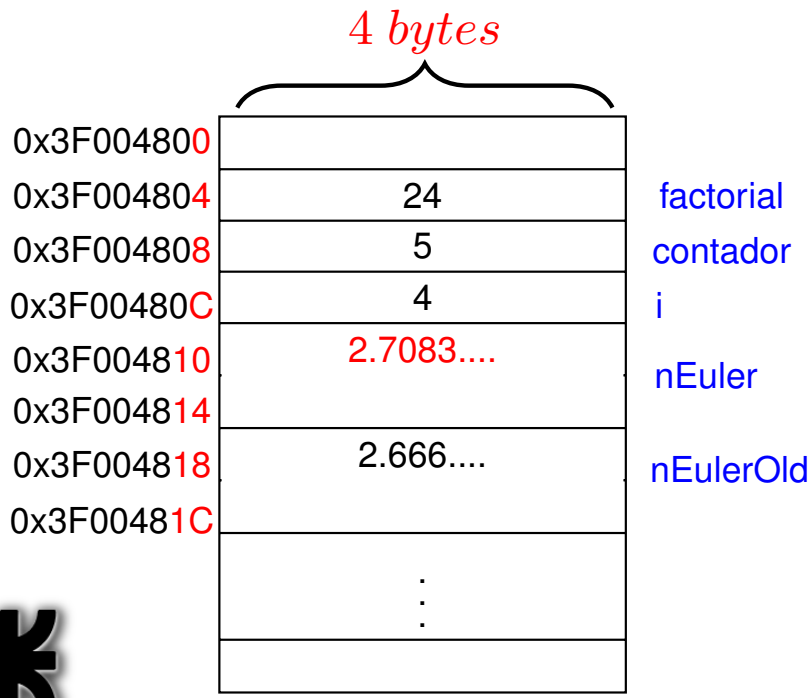
Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

almacena el resultado en la variable nEuler



Código del programa fuente

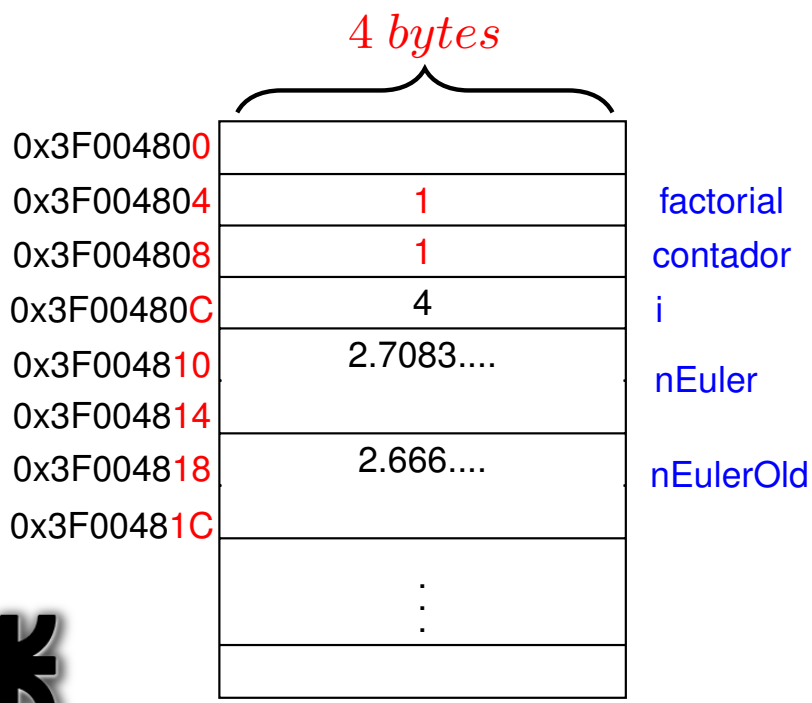
```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

$2{,}66 + \frac{1}{24{,}00} = 2{,}7083\widehat{}$

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

inicializo nuevamente las variables factorial y contador



Código del programa fuente

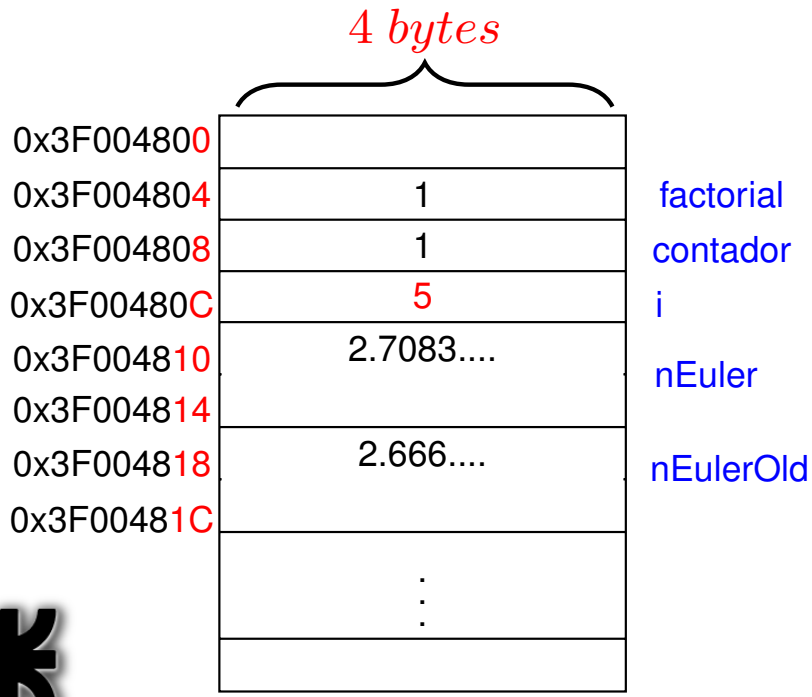
```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

$2{,}66 + \frac{1}{24{,}00} = 2{,}7083\widehat{}$

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

incremento la variable i



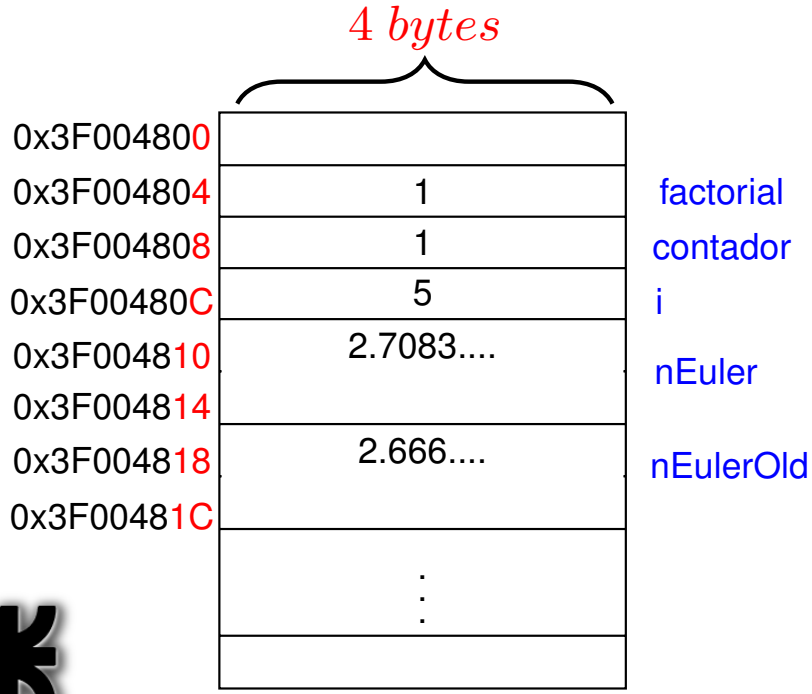
Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

Pregunta condición del for



Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

10^{-9}

Terminando el ciclo for

Si seguimos observando paso a paso como se va ejecutando el programa y como van cambiando los valores de las variables de memoria, llegaremos al punto donde en el ciclo del for no se va a cumplir la condición

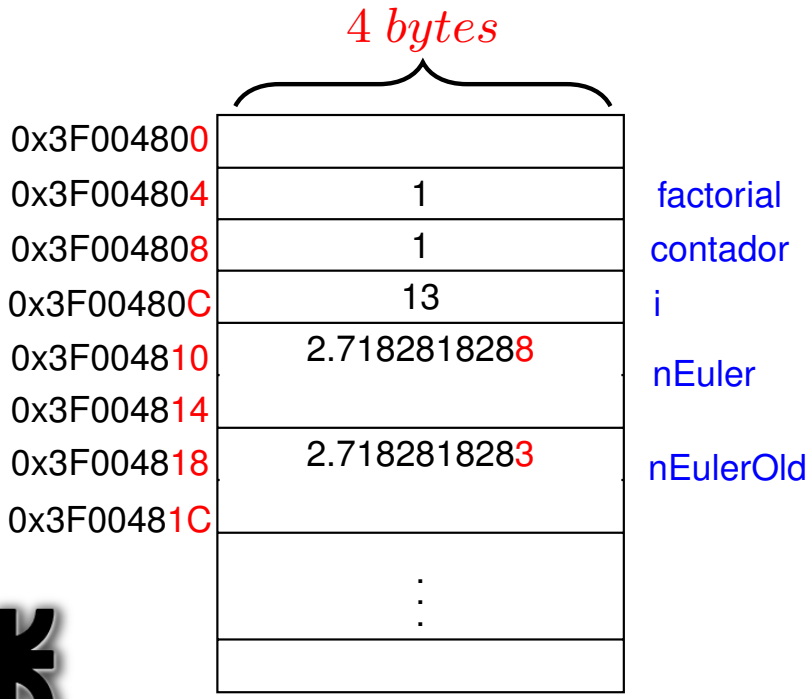
$$(nEuler - nEulerOld) >= LIMITE$$

y por lo tanto los valores de memoria tendrán los siguientes valores

Recorriendo el programa paso a paso

Arquitectura X86-32 bits

No cumple condición del for
$(nEuler - nEulerOld)$ *es* 0,0000000005



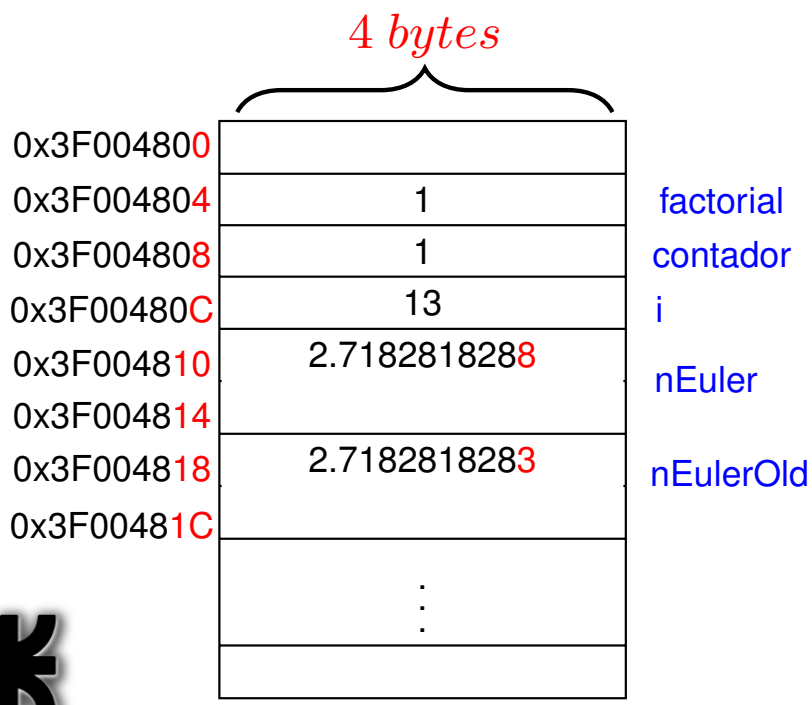
Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```

10^{-9}

Arquitectura X86-32 bits

e es 2,7182818288
e $(lib\ math.h)$ es 2,7182818285



Código del programa fuente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define LIMITE    pow(10,-9)
5  int   main(void)
6  {
7  int       factorial = 1,contador =1,i;
8  double    nEuler=0,nEulerOld=-1;
9
10 for(i=0;(nEuler-nEulerOld)>= LIMITE;i++)
11   }
12       nEulerOld=nEuler;
13       while(contador <= i)
14       {
15        factorial = factorial * contador;
16        contador++;
17       }
18       nEuler = nEuler + (1/(double)factorial);
19
20       factorial = 1;
21       contador =1;
22   }
23   printf("e es %0.10f \n",nEuler);
24   printf("e(lib math.h)es %0.10f\n",M_E);
25   return 0;
26 }
```