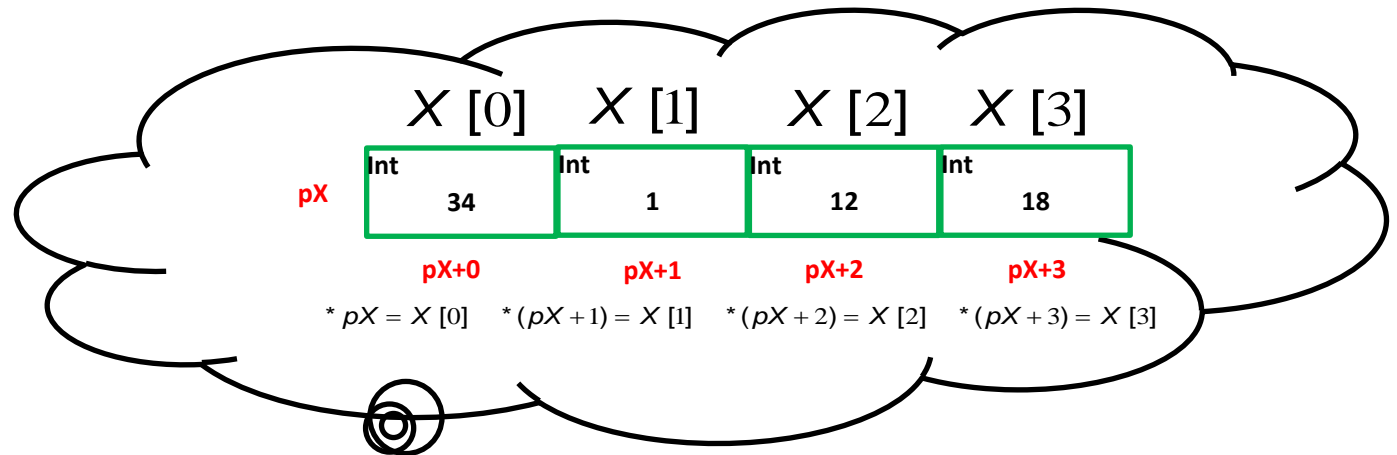


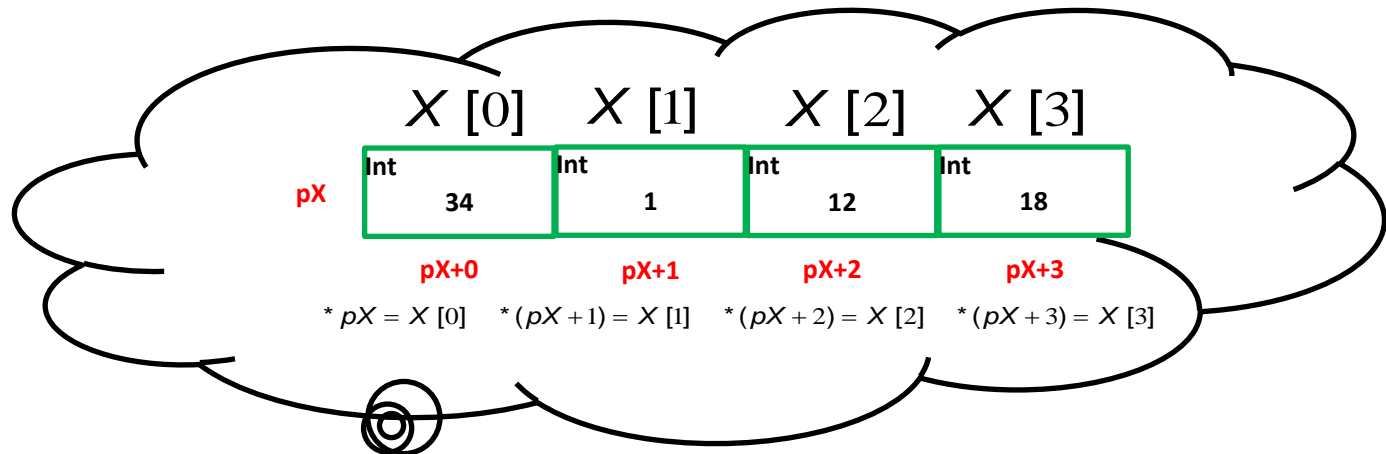
ARREGLOS DINAMICOS

Es un conjunto de variables que tienen algo en común, y que son del mismo tipo de dato, pero su tamaño puede ser definido y editado en tiempo de ejecución.

$$t_{EJECUCION} = t_0$$



$$t_{EJECUCION} = t_1$$



COMANDOS UTILIZADOS EN ARREGLOS DINAMICOS

Sizeof(), es una función que calcula el tamaño de byte o bit utilizado por un tipo de dato.

sizeof(Tipo de Dato)

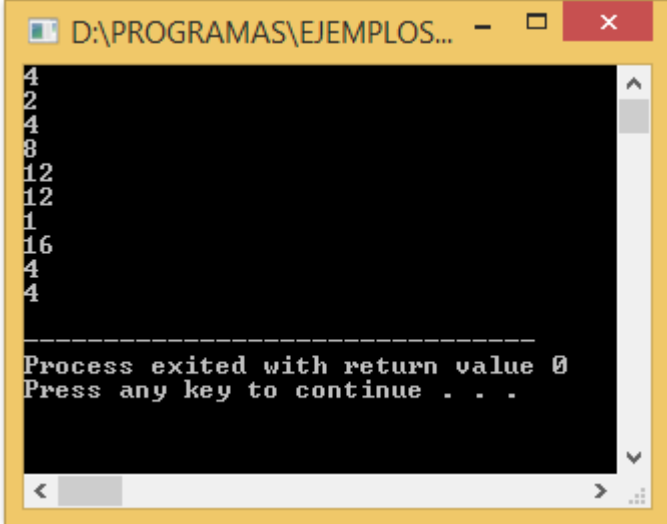
Ejemplo: *sizeof(int)*

Devuelve el tamaño del tipo de dato, para algunos casos es 4 bytes, por lo que la función devuelve 4.

COMANDOS UTILIZADOS EN ARREGLOS DINAMICOS

EJEMPLO DE USO SIZEOF()

```
1  # include <stdio.h>
2  # include <stdlib.h>
3  # include <iostream>
4
5  main(void)
6  {
7      struct Bases { int a; char b; double c; };
8
9      int x;
10     char y;
11
12     x=sizeof(int);
13     printf("%i\n",x);
14
15     x=sizeof(short int);
16     printf("%i\n",x);
17
18     x=sizeof(unsigned int);
19     printf("%i\n",x);
20
21     x=sizeof(double);
22     printf("%i\n",x);
23
24     x=sizeof(long double);
25     printf("%i\n",x);
26
27     x=sizeof(long double);
28     printf("%i\n",x);
29
30     y=sizeof(char);
31     printf("%i\n",y);
32
33     y=sizeof(Bases);
34     printf("%i\n",y);
35 }
```



```
4
2
4
8
12
12
1
16
4
4
-----
Process exited with return value 0
Press any key to continue . . .
```

COMANDOS UTILIZADOS EN ARREGLOS DINAMICOS

`malloc()`, es una función que separa memoria en una cantidad indicado por `r`, no encontrarlo devuelve `NULL`..

`malloc(r)`

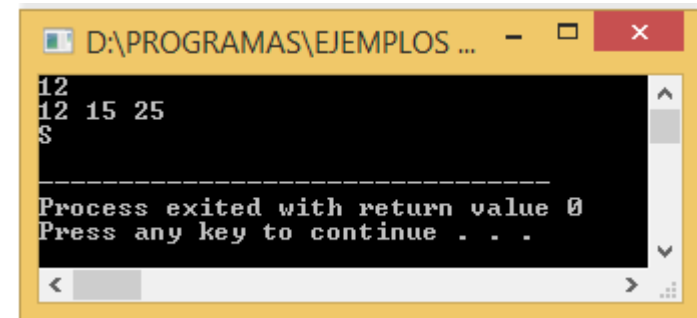
Ejemplo: *`malloc(sizeof(int))`*

Devuelve el puntero de una cantidad de byte dado por `sizeof(int)`, si ese valor devuelto por `sizeof(int)` es 4, entonces el tamaño de la memoria separada es de 4 bytes.

COMANDOS UTILIZADOS EN ARREGLOS DINAMICOS

EJEMPLO DE USO SIZEOF()

```
1  # include <stdio.h>
2  # include <stdlib.h>
3  # include <iostream>
4
5  main(void)
6  {
7
8  int x, *px;
9  char y, *py;
10
11  px=(int *)malloc(sizeof(int));
12  *px=12;
13  printf("%i\n", *px);
14
15  px=(int *)malloc(3*sizeof(int));
16  *px=12;
17  *(px+1)=15;
18  *(px+2)=25;
19
20  for (int i=0;i<3;i++) printf("%i ", *(px+i));
21  printf("\n");
22
23  py=(char *)malloc(sizeof(char));
24  *py='S';
25  printf("%c\n", *py);
26  }
27
```



The screenshot shows a Windows command prompt window with the title bar "D:\PROGRAMAS\EJEMPLOS ...". The window contains the following text:

```
12
12 15 25
S
-----
Process exited with return value 0
Press any key to continue . . .
```

COMANDOS UTILIZADOS EN ARREGLOS DINAMICOS

`calloc()`, es una función que separa memoria en una cantidad de veces indicado por `r`, busca en la memoria espacios de memoria adyacentes, de no encontrarlo devuelve `NULL`.

`calloc(r, sizeof(Tipo de Dato))`

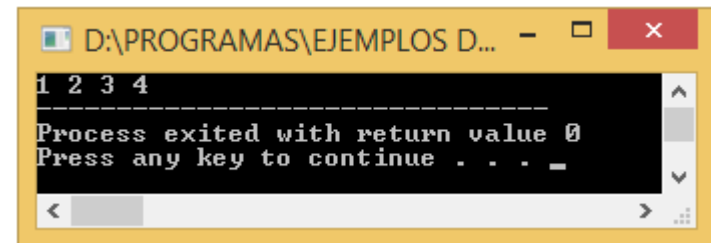
Ejemplo: *`calloc(3, sizeof(int))`*

Devuelve el puntero de tres espacios de memoria de un tamaño dado por `sizeof(int)`, entonces el tamaño de la memoria separada es de 12 bytes, para tres valores enteros.

COMANDOS UTILIZADOS EN ARREGLOS DINAMICOS

EJEMPLO DE USO calloc()

```
1  #include <iostream>
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  main(void)
6  {
7      int n=4;
8      int *pA,*pB;
9      pB=(int*) calloc(n,sizeof(int));
10     *pB=1;
11     *(pB+1)=2;
12     *(pB+2)=3;
13     *(pB+3)=4;
14     for (int i=0; i<4;i++) printf("%i ",*(pB+i));
15 }
16
```



```
D:\PROGRAMAS\EJEMPLOS D...
1 2 3 4
-----
Process exited with return value 0
Press any key to continue . . . _
```

COMANDOS UTILIZADOS EN ARREGLOS DINAMICOS

DEFINE, es una directiva preprocesador (#), que permite definir variable y funciones antes de ser procesado el programa.

define Etiqueta funcion;

Ejemplo 1: *# define G 3.14;*

Cada vez que se utilice el valor G, el procesador lo interpreta como 3.14.


Ejemplo 1: *# define f(x,y) 2 * x + y * y;*

Cada vez que se utilice el la función f(x,y) devolverá la evaluación de la regla de correspondencia $2*x+y*y$

COMANDOS UTILIZADOS EN ARREGLOS DINAMICOS

EJEMPLO DE USO define

```
1  #include <iostream>
2  #include <stdio.h>
3  #include <stdlib.h>
4  #define G 3;
5  #define f(x,y) 2*x+y*y;
6
7  main(void)
8  {
9      int n, m;
10     n=G;
11     m=f(1,2);
12
13     printf("%i %i ",n, m);
14 }
```



```
D:\PROGRAMAS\EJEMPLOS DE CLA...
3 6
-----
Process exited with return value 0
Press any key to continue . . . _
```