APUNTADORES

APUNTADORES O PUNTEROS

Un apuntador es una variable que almacena direcciones de memória

APUNTADORES O PUNTEROS

```
int x=3;

cout <<  "\n Valor de x = " << x;

cout <<  "\n Direccion de memoria de x = " << &x;
```

APUNTADORES O PUNTEROS

```
int x;
int *apuntador;
```

APUNTADORES O PUNTEROS

```
int x;
int *apuntador;

x = 10;
apuntador = &x;
```

APUNTADORES O PUNTEROS

```
int x;
int *apuntador;

x = 10;
apuntador = &x;

cout <<  "\n Valor de x = " << x ;
cout <<  "\n Direccion de memoria de x = " << apuntador;
```

PUNTEROS

```
// Una variable puede ser accedida de dos formas distintas.
// Recordemos que una variable no es mas que una posición de memoria
// que se reserva para almacenar un valor, y que se asigna en el
// momento de la declaración

int x=3;

cout << x;  // imprime 3

cout << &x; // imprime la dirección de memoria
                    // 0x7fff5c137bfc

& aplicado a una variable obtiene la dirección de memória

* aplicadoa un apuntador obtiene el contenido de la dirección de
  la variable a la que apunta.
```

PUNTEROS Y VALORES

```
int x;
int *apuntador;

x = 10;
apuntador = &x;

cout <<  "\n Valor de numero = " << x ;
cout <<  "\n Direccion de memoria de x = " << apuntador;

// Imprimir valore del apuntador
cout <<  "\n Direccion de memoria de x = " << *apuntador;
```

PUNTEROS Y VALORES

```
int x;
int *apuntador;

x = 10;
apuntador = &x;

*apuntador = 20;

cout <<  "\n Valor de numero = " << x;
cout <<  "\n Direccion de memoria de x = " << apuntador;

// Imprimir valore del apuntador
cout <<  "\n Direccion de memoria de x = " << *apuntador;
```

PUNTEROS

```
void suma(int a, int b);

int total; // Variable global , declarada fuera del main

int main (void){
  int a,b; // Variables locales
  a=b=1;
  suma(a,b);
  cout << " a = " << a << ", b= " << b << ", suma = " << total;
}

void suma(int a, int b ){
  total = a+b; // Total es variable global
  a = 2;
  b = 2;
}
```

PUNTEROS

```
void suma(int *a, int *b);

int total; // Variable global , declarada fuera del main

int main (void){
  int a,b; // Variables locales
  a=b=1;
  suma(&a,&b);
  cout << " a = " << a << ", b= " << b << ", suma = " << total;
}

void suma(int *a, int *b ){
  total = *a+*b; // Total es variable global
  *a = 2;
  *b = 2;
}
```

EJERCICIOS

```
char cadena[100];
int i=0;
cout << "Escriba una frase:";


// Imprime la cadena completa
cin.getline(cadena,40);

cout << "\nCadena completa: " << cadena;
```

EJERCICIOS

Escribir un programa que realice los siguientes puntos:

1.- Un procedimiento que solicite n números enteros y los almacene en un arreglo
2.- Un procedimiento que imprima el arreglo
3.- Un procedimiento que imprima la dirección en memoria de cada uno de los valores del arreglo
4.- Un procedimiento que reciba cada uno de los valores del arreglo en un apuntador y que incremente en 1 cada uno de los valores proporcionados por el usuario

EJERCICIOS

```
// Solicita total de numeros

cout << "\n Introduzca cantidad de numeros a solicitar: ";
cin >> total;

int arreglo[total];
```

EJERCICIOS

```
// Solicita numeros

for(int i=0; i < total; i++){
  cout << "\n Introduzca un numero: ";
  cin >> arreglo[i];
}
```

EJERCICIOS

```
// Imprimir Arreglo

cout << "\n Los valores del arreglo son: \n";

for(int i=0; i < total; i++){
  cout << arreglo[i]);
}
```

EJERCICIOS

```
// Imprimir Direcciones De Memoria

cout << "\n Las direcciones del arreglo son: \n";

for(int i=0; i < total; i++){
  cout << &arreglo[i];
}
```

EJERCICIOS

```
// Incrementar Numeros Del Arreglo

cout << "\n Los valores del arreglo + 1 son: \n";

for(int i=0; i < total; i++){
  arreglo[i]++;
}
```

EJERCICIOS

```
void solicitarNumeros();
void imprimirArreglo();
void imprimirDireccionesDeMemoria();
void incrementarNumerosDelArreglo();

int main(){
  int total;

  cout << "\n Introduzca cantidad de numeros a solicitar: ";
  cin >> total;
  int arreglo[total];

  solicitarNumeros();
  imprimirArreglo();
  imprimirDireccionesDeMemoria();
  incrementarNumerosDelArreglo();
  imprimirArreglo();

  return 0;
}
```

EJERCICIOS

```
void solicitarNumeros(int *arreglo, int total){
  for(int i=0; i < total; i++){
        cout << "\n Introduzca un numero: ";
    cin >> arreglo[i];
  }
}
```

EJERCICIOS

```
void solicitarNumeros(int *arreglo, int total);

int main(){
  int total;

  cout << "\n Introduzca cantidad de numeros a solicitar: ";
  cin >> total;
  int arreglo[total];

  solicitarNumeros(&arreglo, total);

  return 0;
}

void solicitarNumeros(int *arreglo, int total){
  for(int i=0; i < total; i++){
    cout << "\n Introduzca un numero: ";
    cin >> arreglo[i];
  }
}
```

EJERCICIOS

Escribir un programa que contenga los siguientes puntos:

1.- Un procedimiento que muestre un saludo por pantalla.
2.- Una función que calcule el cuadrado de un número cualquiera y lo muestre en pantalla
3.- Un procedimiento que muestre por pantalla los números del 1 al 100
4.- Una función que realice la media de dos números, utilizar una función.
5.- Un procedimiento que pida el nombre, el apellido y un lugar y que muestre una pequeña historia usando los datos solicitados
6.- Un procedimiento que pida un número del 1 al 10 y que muestre el número escrito en letras

EJERCICIOS

```
char cadena[100];
int i=0;

cout << "Frase: ";

fflush(stdin); //limpia el buffer del teclado
gets(cadena); //lee una cadena que puede contener espacios

//Imprimir cadena, pero caracter por caracter
while(cadena[i]!=0){
  cout << cadena[i++];
}

cout << " Cadena completa " << cadena;
```

EJERCICIOS

7.- Que pida por pantalla una temperatura en grados Celsius,
muestre un menú para convertirlos a Fahrenheit o Kelvin y
muestre el equivalente por pantalla, utilizar funciones
8.- Que muestre por pantalla si un número es par o impar,
utilizar una función
9.- Que rellene un array del 1 al 20, utilizar un procedimiento
10.- Que muestre el contenido del array anterior mediante
un procedimiento
11.- Que muestre una tabla de multiplicar de un número cualquiera
por pantalla, el número se pedirá en el programa principal
12.- Que muestre 3 números ordenados de ascendentemente y
descendentemente, utilizar un procedimiento para cada operación
13.- Que verifique que un carácter introducido es un número,
utilizar funciones
14.- Que transforme un número del 0 al 999 a números romanos,
utilizar funciones.