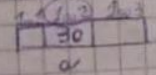
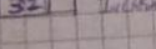
PUNTEROS

2 TIPOS DE VARIABLES (DIRECCIÓN Y POSICIÓN DE OTRA VARIABLE)

```
INT a=5
INT *Pa; //PUNTERO A UN ENTERO.
Pa = &a // Pa APUNTA A a
*Pa = 30; // a = 30
```




```
INT a=5;  INT b=3;
b = *Pa; // b = a CAMBIA EL VALOR.
*Pa = *Pa++; //
```



INCREMENTA.

PARA DECLARAR UN PUNTERO
TIPO DE DATO * NOMBRE.
& DIRECCIÓN
* INDIRECCIÓN

```
INT AUX;
INT a=4; INT b=6;
AUX=a;
a=b;
b=AUX;
```

→

```
INT AUX;
INT *Pa; INT *Pb;
Pa = &a;
Pb = &b;
*Pa = 7;
*Pb = 1;
AUX = *Pa;
Pa = *Pb;
Pb = AUX;
```

```
INT V[5] = {1,5,7,10,15};
INT I=0;
FOR(I=0, I<5; I++){
  PRINTF("%d", V[I]);
}
```

→

```
INT *P;
P = &V[0];
FOR(I=0, I<3; I++){
  PRINTF("%d", *(P+I));
}
```

PASAR POSICIÓN DE VECTOR
*P = V[0]
*(P+1) = V[1]

```
INT V[3] = {1,3,5};
INT I=0;
INT V2[3];
FOR(I=0, I<3, I++){
  PRINTF("%d", V[I]);
  V2[I] = V[I];
}
```

→

```
INT *P; INT *R;
P = &V[0];
R = &V2[0];
FOR(I=0; I<3; I++){
  *(R+I) = *(P+I);
}
```

[illegible] MALLOC - SIZE. [illegible]

EN C LO TOMA COMO VARIABLE LOCAL.

A = SIZE OFF (INT),
(A[24]);

[illegible] COMO EL INDICE EN UN VECTOR.

CHAR * A = (CHAR*) MALLOC (SIZEOF (CHAR)); ⇒ GUARDA LA POSICIÓN DE LA [illegible]
MEMORIA LIBRE EN ESA [illegible]

NO PODEMOS NOSOTROS ASIGNAR ESPACIO A LA PC.

VOID FUNCIÓN (VOID* P) → SEGÚN GALFI NO USARLO, SOLO SABER Q' ESTA.

5) REALIZAR UN PROGRAMA CON PUNTEROS DE BUSCAR TEXTO, USANDO 2 VECTORES.

```
INT MAIN()
PRINTF("%C %C %D \N", V1[I], V2[I], CANT);
{
CHAR V1[] = "HOLA, COMO ESTAS";
CHAR V2[] = "EST";
INT I, J, CANT, L, N1, N2;
N1 = STRLEN(V1);
N2 = STRLEN(V2);

FOR (I=0; I<N1-N2; I++) {
   CANT = 0;
   L = 0;
   FOR (J = I; J < (I+3); J++) {
      IF (V1[J] == V2[L])
         CANT++;
   //PRINTF("%C %C %D \N", V1[I], V2[I], CANT);
      L++;
   }
   IF (CANT == 3) {
      PRINTF ("ESTA EN LA POSICIÓN %D", I);
      RETURN X;
   }
}
PRINTF ("NO ESTA");
RETURN 0;
}
```

IMPORTANTE ⇒

```
INT MAIN() {
   INT * PTR;
   INT DATO = 3;

   PTR = &DATO;
   * PTR = 5;
   PRINTF("DATO = %D \N", DATO);
   RETURN 0;
}
```

→ ME MUESTRA EL NUEVO VALOR DE DATO ASIGNADO CON PUNTEROS

6.1) REALIZAR UN ARREGLO CON PUNTEROS, DONDE MUESTRE POR PANTALLA LAS 5 POSICIONES CON SUS RESPECTIVOS VALORES C/U Y QUE SUMA SU POSICIÓN MOSTRANDO INVERSAMENTE SU NÚMERO.

```
INT MI_ARREGLO[] = {1, 23, 17, 4, -5, 100};
INT *PTR;
INT MAIN() {
INT I;
PTR = &MI_ARREGLO[0]; // APUNTAMOS NUESTRO PUNTERO AL PRIMER ELEMENTO DEL ARREGLO
PRINTF("\n\n");
FOR(I=0; I<6; I++){
PRINTF("MI_ARREGLO[%d] = %d", I, MI_ARREGLO[I]);
PRINTF("PTR + %d = %d\n", I, *(PTR+I));
}
RETURN 0;
}
```

6.2) REALIZAR UNA CADENA DE CARACTERES CON PUNTEROS, DONDE COPIE UNA CADENA CON OTRA CON 2 CADENAS. Y LUEGO MUESTRE EL MENSAJE HOLA 3ER VEZ.

```
CHAR STRA[100] = "CADENA A USAR PARA EL PROGRAMA";
CHAR STRB[100];
INT MAIN(){
CHAR *PA; // PUNTERO AL CARACTER A
CHAR *PB; //  "    "    "    B
PUTS(STRA); // MUESTRA CADENA STRA
PA = STRA; // APUNTA PA A CADENA STRA
PUTS(PA); // MUESTRA DONDE APUNTA PA
PB = STRB; // APUNTA PB A CADENA STRB
PUTCHAR('\n'); // LINEA EN BLANCO
WHILE(*PA != '\0'){ // MIENTRAS EL CARACTER APUNTADO POR PA NO SEA CARACTER NUL
*PB++ = *PA++; // COPIA EL CARACT APUNTADO POR PA A PB, INCREMENTA PA Y APUNTA AL SIG CARACT E IGUAL A PB.
}
PB = '\0'; // SE AGREGA NUL (POR DEFINICIÓN).
PUTS(STRB); // MUESTRA STRB EN PANTALLA.
RETURN 0;
}
```

PUTS ACTÚA COMO PRINTF

'\0' = CARACT. NUL.

63) REALIZAR UNA ESTRUCTURA CON PUNTEROS; LUEGO LA ESTRUCTURA TENGA LUGAR CARGA LA TRABAJAR Y QUE SEA INGRESADA, LUEGO MOSTRARLOS EN PANTALLA EL NOMBRE, APELLIDO Y EDAD.

```
STRUCT FICHA {
CHAR NOMBRE[20];
CHAR APELLIDO[20];
INT EDAD;
FLOAT SALARIO;
};
//CREAMOS VARIABLE TIPO STRUCT FICHA
STRUCT FICHA MI_FICHA;
//DECLARAMOS EL PROTOTIPO MOSTRAR
VOID MOSTRAR (STRUCT FICHA *P); // PROTOTIPO DE LA FUNCIÓN
INT MAIN (){
STRUCT FICHA *ST_PTR; // PUNTERO A UNA STRUCT FICHA
ST_PTR = &MI_FICHA; // APUNTA EL PUNTERO A MI_FICHA
//INGRESAMOS LOS DATOS
STRCPY(MI_FICHA.APELLIDO, "PEREZ");
STRCPY(MI_FICHA.NOMBRE, "NICOLAS");
MI_FICHA.EDAD = 28;
MOSTRAR(ST_PTR); // LLAMAMOS LA FUNCIÓN PASÁNDOLE EL PUNTERO
RETURN 0;
}
VOID MOSTRAR (STRUCT FICHA *P){ // P APUNTA A UNA ESTRUCTURA -> DES-REFERENCIA
PRINTF("\n %S", P->NOMBRE); // IMPRIMIMOS
PRINTF("%S \n", P->APELLIDO);
PRINTF("%D \n", P->EDAD);
}
```

DESREFERENCIA UN PUNTERO A ESTRUCTURA

ST_PTR -> EDAD = 63;

64) REALIZAR UNA FUNCIÓN CON PUNTEROS; DANDO UN VALOR Y LUEGO LO ELEVE AL CUADRADO. AMBOS RESULTADOS DEBEN MOSTRARSE EN PANTALLA.

```
VOID CUADRADO (INT*);
INT MAIN (){
INT AL = 5;
PRINTF("VALOR ORIGINAL = %D \N", AL);
CUADRADO (&AL); // ENVIA A LA DIRECCIÓN DE LA VARIABLE (UN PUNTERO)
PRINTF("VALOR AL CUADRADO = %D \N", AL);
RETURN 0;
}
VOID CUADRADO (INT *NRO) {// *INT = RECIBE UN PUNTERO A UN ENTERO
*NRO = *NRO * *NRO; // VALOR DE LA VARIABLE APUNTADA POR NRO.
}
```

65) Realizar una función con punteros y usar hasta un vector de 10 números desordenados y ordenarlo de M. Burbuja ordenados en un vector distinto. Ambos vectores mostrarlos por pantalla usando el M. Burbuja ordenados en un vector distinto. Ambos vectores mostrarlos por pantalla

```
INT ARR [10] = {3,6,1,7,3,8,4,1,7,2};
VOID BUBBLE (INT *P, INT N); // PROTOTIPO
INT COMPARE (INT *m, INT *n); // COMPARA SI ES MAYOR
INT MAIN (){
INT I;
FFLUSH (stdin);
FOR (I=0; I<10; I++){
PRINTF ("%d", ARR [I]); // IMPRIME EL VECTOR DESORDENADO
}
BUBBLE (ARR, 10); // BURBUJEA CON EL VECTOR
FFLUSH (stdin);
FOR (I = 0; I < 10; I++){
PRINTF ("%d", ARR [I]); // IMPRIME EL VECTOR ACOMODADO
}
RETURN 0;
}
VOID BUBBLE (INT *P, INT N) { // METODO BURBUJA
INT I, J, T;
FOR (I = N-1; I>=0; I--){
FOR (J = 1, J <= I; J++){
  IF (COMPARE (&P[J-1], &P[J])){
  T = P[J-1];
  P[J-1] = P[J];
  P[J] = T;
}
}
}
INT COMPARE (INT *m, INT *n){
RETURN (*m > *n); // COMPARA SI M ES MAYOR A N Y LO HACE SEAS ASÍ
}
```

Concepto de Puntero

Dirección de memoria → al declarar una variable, tiene 3 atributos:

- Nombre: N
- Tipo: entero (INT)
- Dirección en memoria: 0X4FFFD34

INT N = 40;

0X4FFFD34
40
INT

PRINTF ("%i", N); //Imprime valor N → 40

PRINTF ("%P", &N); //" Posición de la memoria N → 0X4FFFD34
(puntero)

EL PUNTERO APUNTA AL ESPACIO FÍSICO DONDE ESTÁ EL DATO O VARIABLE.

INT TASA = 100;

1000	1001	1002	1003	1004	1005
				100	
				TASA	

→ Supongamos q' en la pos. 1004 se encuentra el valor 100 de TASA

INT * P_TASA;

1000	1001	1002	1003	1004	1005
	?			100	
	P_TASA			TASA	

→ Decidió guardarlo en la pos 1001 (P_TASA) porque aún no iguala la posición de memoria donde está TASA.

P_TASA = &TASA;

1000	1001	1002	1003	1004	1005
	1004			100	
	P_TASA			TASA	

→ Ahora se la iguala a la pos de memoria de la variable TASA, entonces en P_TASA se guarda 1004 (la pos de memoria de TASA).

DECLARACIÓN DE UN PUNTERO: _ TIPO_DE_DATO (INT, CHAR, FLOAT, ETC) *NOMBRE_DE_APUNTADOR;

POR EJ:

_ CHAR *CH1, *CH2;

_ FLOAT *VALOR, *PRECARGADE;

66) Demostrar el uso básico de Punteros (Posiciones de Memoria)

```
INT NUMERO = 50;
INT *P_NUMERO;

P_NUMERO = &NUMERO; // POSICION DE MEMORIA DE NUMERO

PRINTF("VALOR DE LA VARIABLE \n");
PRINTF("DATO: %I", NUMERO);
PRINTF("\n DATO: %I", *P_NUMERO);

PRINTF("\n\n POSICION DE MEMORIA: \n");
PRINTF("POSICION: %P", &NUMERO);

RETURN 0;
```

67) Hacer una variable de tipo INT, otra FLOAT y una CHAR; almacenar datos en c/u de las variables, posteriormente imprima la pos de memoria donde se encuentren. Guardar las pos de c/variable con punteros.

```
INT N = 10, *P_N = &N;
FLOAT N2 = 10,5, *P_N2 = &N2;
CHAR N3 = 'N', *P_N3 = &N3;

PRINTF("VARIABLE ENTERA: \n");
PRINTF("DATO: %I", *P_N);
PRINTF("\n POSICION: %P", P_N);

PRINTF("\n\n VARIABLE FLOTANTE: \n");
PRINTF("DATO: %.2F", *P_N2);
PRINTF("\n POSICION: %P", P_N2);

PRINTF("\n\n VARIABLE CARACTER: \n");
PRINTF("DATO: %C", *P_N3);
PRINTF("\n POSICION: %P", P_N3);

RETURN 0;
```

68) Imprimir el alfabeto en mayús con punteros

```
CHAR LETRA;
CHAR *P_LETRA = &LETRA; // POSICION DE LETRA

FOR (LETRA = 'A'; LETRA <= 'Z'; LETRA++) {
    PRINTF("%C", *P_LETRA);
}
RETURN 0;
```

69) Dado un vector de 10 elementos = {1,2,3,4,4,7,8,9,5,4}, escribir un programa en C (usando punteros) y muestre las direcciones de memoria de cada elemento del vector.

```
INT I,N[] = {1,2,3,4,4,7,8,9,5,4};
INT *P_N;

P_N = N; // P_N = N[0];

FOR(I=0; I<10; I++){
    PRINTF("DATA N[%I] : %I", I, *P_N);
    PRINTF("\t DIRECCION : %P", P_N);
    PRINTF("\n");
    P_N++;
}
RETURN 0;
```

70) Pedir su nombre al usuario y devolver el número de vocales que hay.

```
INT CUENTAVOCALES(CHAR *);  // PROTOTIPO
CHAR NOMBRE[20];

PRINTF("DIGITE SU NOMBRE: ");
GETS(NOMBRE);

PRINTF("EL NÚMERO DE VOCALES ES: %I", CUENTAVOCALES(NOMBRE));

RETURN 0;
}
//VALIDANDO \0

INT CUENTAVOCALES(CHAR *S){
  INT CONT = 0;

  WHILE(*S){ // MIENTRAS QUE S NO SEA NULO
    SWITCH(*S){
        CASE 'A':
        CASE 'E':
        CASE 'I':
        CASE 'O':
        CASE 'U': CONT++;
    }
    S++;
  }
  RETURN CONT;
}
```