|  |  |  |
| --- | --- | --- |
| Tipos de variables  Int var;  Float var;  Char var;  Char var[10];  Char var=’h’ ;  Char var[20]=”hola”;  sizeof(int) | 4byte  4bytes  1byte  Muestra cuanto ocupa bytes | Determina el tamaño y tipo de espacio en memoria  %i(%d,%o%x%b)  %f  %c  %s  Ver comillas simples para un solo carácter  Comillas dobles para una cadena. |
| Printf(“asdf %s”, var); | |  |
| System(“cls”); | | Limpiar pantalla. |
| System(“pause”); | | Pausar la consola |
| Scanf(“%c”, &var);  Scanf(“%s”, var);  scanf (“%d” , &var[i]);  scanf(“%d”, var+1); aritmetica de punteros  Var= getchar();  **Solo pide caracteres (los espacios no son)**  Get(var);  gets(var)  <conio.h>  X=getch()  X=getche() | | & para int float y char simple.  En caso de ser una cadena string no lleva & trabaja diferente a los demas  dirección de memoira +4bits (sizeof(var)) == &var[1]  Especialista en guardar un carácter(toma el carácter que esta en el bufer y lo devuelve)  Si acepta espacios.  Automáticamente toma el primer carácter que pulsemos, no muestra el carácter en pantalla  Similar a getch pero nos muetra el carácter que presionamos |
| #include<ctype.h>  Var=tolower(var);  var=toupper(var) | | Biblioteca  Minúscula  Mayúscula |
| Puts(cadena\_string)  **gets(cadena\_string)**  fgets(var\_cadena, TAM, stdin);  #include <string.h>  strlen(cadena\_string)  strcpy(nombre, “jonathan”);  strncpy(nombre, nombre2,5);  strcmp(cadena1, cadena2);  stricmp( cadena1, cadena2);  strlwr(cadena);  strupr(cadena)  strcat(cadena1 , cadena 2);  sprintf(cadena, “hola me llamo %s, tengo %d”,nombre,edad);  snprintf(cadena,100, “hola me llamo %s, tengo %d”,nombre,edad); | | Para testear muestra cadena nomas (no muy usado)  **A diferencia de scanf, gets admite espacios en blanco**  Esta función evita que el usuario desvorde el array  ------  Muestra la cantidad de caracteres que tiene la cadena  Para asignar un cadena directamente el total de  Copia la var2 en la var1. Solo 5 caracteres  0 se las cadenas son iguales, 1 o -1si son diferentes.  Su función hace: resta posición por posición ascii  Entonces 1 si el primero tiene mayor valor, -1 si el segundo tiene mayor valor.  Igual que strcmp() pero antes transforma en minúscula ambas cadenas.  Una función especial para convertir un vector cadena en minuscula  mayuscula.  **concatena:** La suma de sus caracteres tiene que ser menor al espacio de cadena1  concatena una cadena grande. En la variable **cadena**  tener en cuanta que cadena tiene que tener mas de la cantidad de las demás variables juntas.  Concatena y le restringimos el tamaño máximo  Retornos:  -Retorna la cantidad de caracteres que puedo guardar incluyendo el carácter terminador.  -Si el texto supera el limite retorno a la cantidad que requiere mas el carácter terminador  -si hay una falla retorna un numero negativo que indica el error que ocurrio. |
| Notas stings | |  |
| Svar=scanf(“%d” ,&g);  Var= scanf (“%d %d %dn”, &dia, &mes, &año)  Var= scanf (“%d/%d/%d”, &dia, &mes, &año) | | Nos devuelve 0-1 si puedo validar o no  Si esta todo bien nos devulve 3 o hasta donde puedo gargar |
| Validar un puntero  Int func(…, int\* pHora);  {  If (phora!=NULL){}  } | | Siempre antes de comenzar a trabajar con un puntero hay que validar con ¡=NULL |
| Printf(“%02d” ,dia);  02.d  Var= pintf(…..) | | Nos completa los valores si silo hay 1 carcater num  va ocupar 2 espacios y si no hay números entonces se completa con el cero.  Nos devuelve la cantidad de caracateres que imprimio |
| Explicación | | Que pasa con &?  Las variables se guaran en “diccionarios”  Nombre - Dirección  Var - 19  Var == valor de la variable  &var== dirección de la variable |
| Basura en el buffer | | En los tipos %c %s  Al ingresar varios datos y al momento de llegar un %c%s  Queda almacenado en “enter” y se guarda en el %c%s  Solución: antes de leer un %c%s (**fflush(stdin)**) para limpiar el buffer.  Fflush: limpiar el estándar ingreso(teclado) |
| Solución dato flotantes redondeados   * Promedio=num\_int/2.0; * Promedio=(float)num\_int/2; | | En operaciones matemáticas al hacer puro operaciones con enteros el resultado es entero por más que debería ser float |
| Parsing  Casting explicito (parte humana y tenemos que decirle exactamente lo q hacer por que sino la pc falla)**(float)**  Casting implícito (parte de la maquina lo hace automáticamente y no hay perdida de datos) (8-> 8.00) | | Pasar de string a number  Pasar de number a number (int-float) |
|  | |  |
| Do{  }while(cond); | | Mínimo se realiza una vez |
| %% | | Para imprimir % en printf |
| <ctype.h>  isDigit(x)  isAlpha(x)  isspace(x) | | Devuelve 0-1 si es un nro 0-9 (string) |
| Setbuf(stdout,NULL); | | Para el eclipse |
| Poner texto con ascci  Printf(“mostrar la letra %c en ascci.”, 164 );  Printf(“mostrar el numero %d en ascci.”, ‘A’); | |  |
| Funciones | |  |
| Int nom\_func();  Char nom\_func();  Float nom\_func();  Int nom\_func(int numero1, char letra);  Void nom\_func(char letra);  NOTA:  Parametros formales: son lo que se encuentran el el prototipo de la función.  parámetros acuales: son los parámetros que le pasamos en el main. | | Declarar una función  Demás pasamos los parámetros.  Esta función devuelve valores por lo tanto no tiene retun |
| Para documentar funciones susar **doxygen** que se encuentra en la market | |  |
| #define NADA 0 | | Reemplaza en todo el código nada =0 |
| Variables punteros **Punteros**  **Int var=2;**  **Int\* variable= va1;**  Printf(“%x”, variable);  Printf(“%p”, variable);  Printf(“%d”, \*variable); | | Variables que guardan direcciones de otras varibles  En variable se guarda la dirección de var1.  Muestra la dirección de var1 en exa.  Lo muestra en puntero(solo le aumenta 00).  **Nos muestra el valor de la dirección de variable que guardó previmente.** |
| Paso de paramentros con punteros  Void funcion1(int\* variable){  \*variable=3;  } | | Como usar punteros en funciones.  La ventaja principal es que no necesitamos y un return. Ya que modificamos el valor de las variables directamente desde su dirección en memoria.  Ahora var=3. Una especie de variable global. Se modifico a traves de **variable** |
| Funciones recursivas | | Se llaman a si mismas. |
| Stack  Es una parte de la pc donde se cargan las funciones de manera ordenada.  Las funciones hacen una fila y desaparecen a medida que terminan su acción y pasa a su antecesor. Asi hasta llegar a main y terminar. | |  |
| Cadena de caracteres  **Nota: tratar de declarar múltiplos de 4**  Char nombre[5]={‘j’,’o’,’n’,’y’,’\0’}; | | Carácter terminador ‘\0’ |
| int aux;  int numeros[T]={2,6,3,9,2};  for (int i=0;i<T-1;i++){  for(int u =i+1;u<T;u++){  if(numeros[i]>numeros[u]){  aux=numeros[i];  numeros[i]=numeros[u];  numeros[u]=aux;  }  }  } | | Ordenamiento   * Para ordenar en formar descendente   if(numeros[i]<numeros[u]){   * Para ordenar un flotante   Declarar las variables en float   * Para ordenar un caracter   Declarar las variables en char |
| Matrices  Int matriz[2][4]={{1,2,3,4}{1,2,3,4}};  Char matriz[2][4]={{“jon”},{“apu”}}  Para pasar como parámetro a una función  mostraMatriz(matriz);  int mostrarMatriz(int fila, int col, int **matriz[][col]**) | | **Necesita que le pasemos el segundo tamaño** |
| ***Estructuras*** | | |
| Typedef struct{  }eNombre; | | Estructura de las struct con el nombre eNombre  Fuera del main |
| (main)  aNombre alumno1; | | Declaramos estructura llamada alumno1, ahora alumno tiene todas las propiedades de eNombre |
| typedef struct{  int legajo;  char nombre[20];  char sexo;  int nota1;  int nota2;  float promedio;  }eAlumno;  (Main)  eAlumno alumno1={123123, “jonathan”, ‘m’, 5, 7, 6.00};  (Main)  eAlumno alumno1;  alumno1.legajo=123123;  strcpy(alumno1.nombre,”jonahtan”);  alumno1.sexo=’m’;  alumno1.nota1=5;  alumno1.nota2=7;  alumno1.promedio=6; | | **Llenar datos**  Forma1  Foma2 |
| Void muyFuncion(aNombre\* a){}  Void muyFuncion(aNombre a[], int tam){} | | **Pasar una estructura como parámetro a una función**  El tipo de variable es una estructura y se le pone el nombre de la estructura (aNombre)  Pasar una lista estructura como parámetro. |
| (Main)  eAlumno alumnoCopia;  eAlumno alumno1={123123, “jonathan”, ‘m’, 5, 7, 6.00};  alumnoCopia=alumno1; | | **Copiar una estructura a otra estructura**  Declaración y asiganacion  Ahora tenemos 2 varibles estructura similares |
| eAlumno alumnos[TAM];    eAlumno lista[TAM]={  {123123, “jose”, ‘f’, 5, 7, 6.00},  {123124, “jon”, ‘m’, 6, 9, 7.00},  {123125, “jona”, ‘m’, 8, 10, 8.00}  }; | | **listas** |
| **Estructura dentro de otra estructura**  Typedef struct{  -----  }eFecha;  Typedef struct{  -----  eFecha nacimiento  }ePersona; | | **Notas:** no hereda todas las propiedades.  Simplemente es una estructura que trabaja dentro de la otra estructura. Por lo tanto para llamar a una prop:  ePersona.nacimiento.dia |
| **Pasar una estructura por referencia**  Por ejemplo queremos cargar los valores de una estructura en una función.  (main)  eAlumno alumno;  cargarAlumnos(&alumno);  void cargarAlumnos(eAlumnos\* pAlumno){    } | | **Importante** |

Memoria dinámica