TEORÍA DEV-C

EL C es un lenguaje compilable, que a diferencia de Python la ejecución de las órdenes no es inmediata.

El programa pasa por un intérprete que va leyendo instrucción por instrucción y cuando la a la leído ejecuta la acción que le estas diciendo en el dispositivo.

Si quieres hacer algo rápido en vez de usar un lenguaje intérprete usar uno compilable.

Compilador: otro programa que lee el fichero y lo traduce a código maquina a 0 y 1, lenguaje que el ordenador pueda entender. Lo traduce a lo que se llama ejecutable, porque este es la forma rápida del ordenador para hacer algo.

Los ficheros ejecutables se distinguen por la ejecución .exe.

Los programas de DEV primero se **compila (coger un fichero, programa y transformarlo en un ejecutable, de cc a .exe).**

DENTRO DELPROGRAMA DEV-C

Instrucciones al procesador🡪 ficheros de cabecera

Iostream: corriente de entrada y salida

Main: cuerpo principal. Primera función

Las llaves {}: tabuladores en Python

Std:: estándar

* Voy a hacer una instrucción de tipo entrada y salida y se utiliza el procedimiento estándar.

Cout: de entrada

Cin: tecla para terminar

*FINALIZAR SIEMPRE CON ;*

Cuando el programa ha concluido poner al final un: return 0;

*🡪EXECUTE-COMPILAR PARA VER SI EL PROGRMA TIENE ALGUN ERROR*

PROGRAMA 2:

ERRORES SINTACTICOS: errores de escribir mal las cosas

ERRORES LÓGICOS/ DE CONCEPTO/PROGRAMACION: el programa está bien pero no hace lo que se quiere hacer.

PROGRAMA 4:

Paréntesis para el if y for siempre

PROGRAMA 5-6 🡪VECTORES

Arrays: listas no variables

PROGRAMA 10 🡪 FICHEROS

Fichero: contiene la memoria de lo que voy a escribir

Txt: fichero de carácter texto

Endl: salto de la línea

fichero.close(); : para que se pueda guardar todo después

getline(fichero, mensaje): para escribir una línea, leo el fichero, abre una línea y la mete en el mensaje.

Eof: end of file, va al punto del fichero que le toca leer y si o ultimo que lee es el final de fichero, devuelve 1, porque es el final

PROGRAMA 12 🡪 AGENDA

Char: es una letra, variable tipo letra

String: conjunto de números que trata como numeros

EXAMEN🡪 BUCLE DE UNA VARIABLE INTRRUPTORA: ES UN BUCLE DONDE LEO UNA VARIABLE Y DEPENDIENDO DE SU VALOR ME QUEDO O ME VOY

PROGRAMA 13🡪 LECTURA HASTA EL FINAL DE UN FICHERO

Explica esta instrucción:

ifstream fichero(""); 🡪el comando ifstream busca la ruta que escriba, (lo lee del disco duro y lo vuelca, lo escribe en la memoria ram) una vez abierto utiliza:

getline(fichero); 🡪 va la memoria ram y lee todos los símbolos que hay hasta que encuentre el símbolo de retorno de carro. Guarda la línea de texto en una variable, que llamaremos línea.

PROGRAMA 18🡪 PUNTEROS- NUMERO BINARIO TEORIA

1 Byte son 8 bites: 1 0 0 0 1 1 0 0. Si empieza por 1 es negativo, si empieza por 0, negativo.

La información de mide en byte

Ejemplo: 6 letras x 1 byte/letra

= 6 letras

Tabla ASCII

PROGRAMA 18🡪 PUNTEROS 3

Declarar en la memoria dos variables que son vectores de 5 letras, cuando declaras un vector lleva incluido el porcentaje por lo que no hay que poner %.

POR QUÉ no ponemos “ & “

PROGRAMA 19🡪 PUNTEROS

Indireccionar,

Si añado asterisco: le estoy diciendo que me diga lo que está dentro

+TEORIA:

* PUNTEROS: cuando hay que reservar espacio de la RAM se hace una variable. Se llaman punteros porque apuntan a otro lugar. PX es para apuntar a X.

Guardar lista de palabras: (EJ: nombre de los 3 reyes magos):

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| M | E | L | C | H | O | R |  |
| G | A | S | P | A | R |  |  |
| B | A | L | T | A | S | A | R |

Una lista de palabras es una lista de punteros.

El asterisco es como ir abriendo puertas. \*, entras,\*lista, vas a la lista, + 6 vas hasta esa posición

Da un diseño de una memoria RAM con números y palabras y habría que decir como llegarías a la letra tal. Para ello: (ejemplo buscar la palabra R de M E L C H O R)

(\*(\*lista) +6)

(ej: buscar la letra P de G A S P A R)

(\*(\*lista+1)+3

+TEORÍA

* Memory allocation
* Buffer del teclado
* Strcpy:
* Auxiliar: sitio donde guardas todo lo que metw
* INTERCAMBIO DE 3 BANDAS

ALGORITMOS:

Algoritmos para ordenar, existen dos tipos: el algoritmo de Burbuja y el de Ordenación por selección, en ellos se busca que sea exacto y que lo realice en el menor tiempo posible porque así el algoritmo es más valioso.

EL MÉTODO DE ORDENACION POR BURBUJA: (método clásico)

(se pueden ordenar números, nombres…)

Se cogen dos elementos y se plantea si están bien ordenados, la respuesta puede ser sí o no, si es no, se invierten los números.

Se cogen otros dos elementos y se vuelven a invertir si no están ordenados. Con esto he conseguido que algunos de los números se desplacen pero no está ordenado entero, para ello se vuelve a comenzar el proceso. Hasta que se ordenen los elementos.

¿Por qué se llama burbuja? : Cuando se localiza un número muy grande mal colocado ese número sube a la superficie como las burbujas del champan.

EXAMEN:

1. Declaración y uso de variables y estructuras (utilizar tal varibale para guardar tal dato, ejemplo:

int a;

float peso;

char nombre[10];

int notas[30];

Va a preguntar qué tipo de pregunta: que significa cada una de estas líneas, diferencia entre dos.

* Int a:Esto es un variable llamada a que va a guardar una variable de tipo entero
* La variable peso que va a guardar un numero decimal
* Char nombre: Vector o array que recorre 10 posiciones de memoria y en cada una de ellas vamos a guardar una letra
* Int notas: vector o array para guardar en 30 posiciones 30 números enteros

2. EstructuraS:

Ejemplo:

Struct persona{

String nombre; // librería que necesita para hacer estructuras

Int edad;

};

**Una estructura** es una manera de definir datos de la manera deseada, datos que están formados por distintos datos simples.

Char letra;

Char \*pa;

Letra=’A’;

Pa=&letra //escribe por pantalla la dirección de memoria de la variable letra.

Char \*pa; // Variable que se llama A y contiene un puntero, una dirección e memoria a otra posición que es una letra

Printf(“%c”, letra); le estoy diciendo que saque por pantalla el contenido de letras,

Printf(“%x”,pa); vamos a mostrar algo que va a ser hexadecimal,

Printf(“%c”, \*pa); muestra en pantalla PA

SI NOS DICE:

Int x:

Print(“%i”,x);

Print(“%x,&x);

LOS %

%i 🡪 es un número entero

%c 🡪 es la letra

%x 🡪 es un número hexadecimal

%s 🡪 es una cadena de caracteres

%f 🡪 es un numero decimal (float)

SI PIDE:

Int \*px;

Int x;

X=3

Px= &x; //dirección de memoria

Printf(“%i%,x); //escribe por pantalla un numero entero que esta guardado en la variable x, es decir 3

Printf(“%i”,\*pa); //escribe por pantalla una dirección de memoria que es la dirección de x, Printf(“%x”,&x); //dice la dirección como numero hexadecimal, 44CA0 (la dirección).

PREGUNTA EXAMEN:

Completa las direcciones de memoria y el dibujo para que esto sea realmente una lista de…. Ejemplo: los ganadores entre H, T, C

Char \*ganadores[3]; //vector de punteros

Malloc( //reserva de espacio, para buscar el tamaño de memoria que necesita