LINGUAGGI PROCEDURALI

Linkking: procedura che avviene durante la compilazione e consiste nel collegare le librerie al sorgente

Il main può essere void oppure int, nel caso di int restituisce 0 se il programma termina correttamente

Costanti:

- #Define => sostituisce il valore definito nelle parti con il nome
- Const => tipo di dato costante, se messo nei parametri di una funzione il parametro e in sola lettura

Printf(format, arg1, arg2)

Formattazioni:

- %6d => Occupa 6 caratteri a dx
- %-6d => occupa 6 caratteri a sx
- %ld => long int, lf per double
- %*d => la larghezza viene passata come parametro
- %.2f => tronca a 2 decimali
- %.6d => aggiunge eventuali 0 se il numero non è lungo 6
- %.4s => stringa tagliata a 4 caratteri

Scanf(format, args[])

Per la variabile indicare con & prima del nome, attraverso le formattazioni è possibile dimensionare l'input

Quando si usa scanf con caratteri bisogna pulire il buffer dal \n, usare scanf con uno spazio davanti al tipo di dato da leggere

Random

Generazione numeri random:

Usare libreria time.h, usare srand(time(NULL)) per inizializzare il seme, es: rand()%10 numeri random da 0 a 9.

Rand() genera numeri da 0 a 1, per generare numeri random in un range usare

rand()%(max-min+1)+min

variabili static: sono variabili che anche se dichiarate localmente a una funzione mantengono lo stato anche dopo che la funzione è terminata.

Le variabili globali possono essere oscurate localmente ridichiarandole nella funzione.

Vettori

Ci sono 3 tipi di vettori:

- Dimensione statica => es: int vet[10]
- VLA (Variable Length Array) => es: int n=10; int vet[n]
- A memoria dinamica

Gli array non hanno un valore di default => inizializzare l'array, si possono inizializzare a 0 in fase di dichiarazione es: int vet[10]={0} oppure è possibile caricarli in fase di dichiarazione es: int vet[]={1,2,3}

Passaggio dei parametri

- Valore: crea copia del valore passato nello stack, modifiche solo locali alla funzione
- Riferimento: copia l'indirizzo di memoria del parametro, le modifiche sono effettive all'esterno della funzione

Gli array sono passati per riferimento poiché risulterebbe oneroso copiare un array nello stack.

Per gli array bisogna passare anche la dimensione.

Matrici

Es: int mat[10][5] // matrice 10 righe 5 colonne

Nelle funzioni bisogna specificare la dimensione delle colonne poiché bisogna sapere quando la matrice termina la riga.

Stringhe

Le stringhe in C vanno gestite come array di caratteri, esiste una libreria string.h con funzioni utili

- strcat => concatena stringhe
- strcpy=>copia una stringa
- strcmp=>compara stringhe se 0 sono uguali, può essere usate per comparare caratteri restituisce 1 o -1
- strlen => lunghezza stringa

non si può riassegnare una stringa con str="blbl" bisogna usare le funzioni di copia!!

È possibile convertire tipi di dato dalle stringhe tramite la libreria ctype.h

- isDigit() => dice se è un numero
- isLower() => verifica se minuscolo
- isWhite() => verifica se è vuoto
- atoi() => string to int
- atof() => string to double

sprintf e sscanf leggono e scrivono su buffer specificato come primo parametro esempio file, non lo fanno su stdin e stdout come printf e scanf

Puntatori

Int a=10;

*Int * punt;// sintassi per dichiarare un puntatore*

punt =&a;//assegna a punt l'indirizzo di a con &

*a=*p; //leggi il valore di p con **

puntatore generico void *, un puntatore può essere incrementato o decrementato e aumenta/decresce in base al tipo di dato che punta es: puntatore intero incrementato di 1 scala di 4 byte

si possono gestire dei vettori tramite puntatori perché dati contigui, per le matrici bisogna fare attenzione alle colonne. Sizeof restituisce la dimensione in byte di un tipo o array, per gli array fare sizeof(vet)/sizeof(tipo).

Puntatori a funzione

Es: int (*fp) (int) // puntatore di una funzione che ritorna int con un parametro intero

Per passare il parametro inserire il nome della funzione, per richiamarla si fa fp(parametri)

Con fp nome del puntatore, può essere cambiato

Qsort

Per mette di ordinare elementi specificando la funzione di comparazione

qsort(array, dimensione, sizeof(tipoDatoArray), compareFunction);