Università degli Studi di Ferrara

Corso di Laurea in Matematica - A.A. 2018 – 2019

Programmazione
Lezione 24 – Correzione Simulazione

Docente: Michele Ferrari - michele.ferrari@unife.it

In questa lezione

- Correzione simulazione
- Domande ed esercizi a piacere

Parte teorica

1 - Cosa si intende per linguaggio compilato?

1 - Cosa si intende per linguaggio compilato?

- L'espressione "linguaggio compilato" indica un linguaggio di programmazione implementato tramite un compilatore (un traduttore che converte il codice sorgente in codice macchina), invece di un interprete (che esegue direttamente il codice sorgente).
- Un compilatore genera un nuovo file eseguibile a partire dal codice sorgente
- In generale un programma compilato risulta essere più veloce ed efficiente (maggiore controllo delle risorse)

2 - In un calcolatore perché esiste una gerarchia delle memorie?

2 - In un calcolatore perché esiste una gerarchia delle memorie?

- La velocità di accesso è inversamente proporzionale alla dimensione della memoria, man mano che ci si avvicina al processore si trovano memorie sempre più piccole per massimizzare le velocità con cui si forniscono dati all’elaboratore

3 - Indicare l’elemento centrale di un sistema operativo e di cosa si occupa

3 - Indicare l’elemento centrale di un sistema operativo e di cosa si occupa

- L’elemento centrale di un sistema operativo è il kernel e si occupa principalmente di fornire ai programmi un accesso sicuro e controllato alle risorse

4 - In programmazione cosa si intende per variabile?

4 - In programmazione cosa si intende per variabile?

- Una variabile è uno spazio di memoria identificato da un nome e determinato da un tipo, il tipo ne stabilisce dimensioni ed interpretazione

5 - Indicare i 3 elementi fondamentali per la costruzione di algoritmi e come si relazionano

5 - Indicare i 3 elementi fondamentali per la costruzione di algoritmi e come si relazionano

- I 3 elementi sono: sequenza, selezione ed iterazione.

  Ciascun elemento può contenere se stesso e gli altri.

6 - Spiegare brevemente cosa si intende per sistema time sharing

6 - Spiegare brevemente cosa si intende per sistema time sharing

I sistemi a partizione di tempo (time sharing) simulano un quasi-parallelismo nell'accesso alle risorse.

Supponiamo di avere diversi programmi in memoria, ognuno con un diverso tempo di esecuzione: per evitare che un programma monopolizzi il processore, il tempo viene suddiviso in unità elementari, dette **quanti**, da assegnare ai vari processi secondo una determinata politica.

7 - Indicare e descrivere brevemente le fasi del ciclo di vita di un processo

7 - Indicare e descrivere brevemente le fasi del ciclo di vita di un processo

- Nuovo – è stata richiesta l'esecuzione del processo
- Pronto – Al processo vengono assegnate tutte le risorse necessarie TRANNE il processore
- In Esecuzione – Al processo viene assegnato anche il processore
- Terminato – Il processo ha concluso la sua elaborazione e le risorse vengono liberate

8 - In MATLAB indicare le differenze tra script e function

8 - In MATLAB indicare le differenze tra script e function

- Uno script non ha parametri d’ingresso e lavora nello spazio di memoria globale di matlab come se ogni istruzione venisse lanciata dalla command window in successione
- Una Function ha parametri d’ingresso ed uscita e le variabili create al suo interno vengono eliminate al termine dell’esecuzione della function, una function può essere chiamata da command window, da uno script o da un’altra function, una function non può essere eseguita direttamente

9 - Scrivere una funzione in C che, preso in ingresso un numero, visualizzi a schermo se il numero e pari o dispari e restituisca 0 se il numero è pari o 1 se il numero è dispari

9 - Scrivere una funzione in C che, preso in ingresso un numero, visualizzi a schermo se il numero e pari o dispari e restituisca 0 se il numero è pari o 1 se il numero è dispari

```
int pari(int x){
  if ((x%2)!=0){
      printf(“Il numero %d e’ dispari”,x);
      return 1;
  }else{
      printf(“Il numero %d e’ pari”,x);
      return 0;
  }
}
```

10 - Inizializzare un vettore di N elementi con valori casuali compresi tra 0 e 10 (che libreria devo includere?)

10 - Inizializzare un vettore di N elementi con valori casuali compresi tra 0 e 10 (che libreria devo includere?)

```
#include <stdio.h>
#include <stdlib.h>
//#include <time.h>
#define N 5

int main (void){
  int V[N], i;
  //srand((unsigned) time(NULL));
  for (i=0;i<N;i++){
    V[i]=rand()%11;
  }
    return 0;
}
```

11 - Descrivere l’algoritmo di Insertion Sort

11 - Descrivere l’algoritmo di Insertion Sort

L'insertion sort, (ordinamento per inserimento), è un algoritmo di ordinamento sul posto, (cioè ordina l'array senza doverne creare una copia), si tratta di un modo relativamente semplice per ordinare un array.

Strategia

L'algoritmo utilizza due indici: i e j, i quali scorrono l’array a due “velocità” diverse:

- L’indice più lento (nel nostro caso j) punta inizialmente al primo elemento dell’array, mentre l’indice più “veloce” (nel nostro caso i), punta inizialmente al secondo elemento dell'array
- L’elemento puntato dall’indice j, viene confrontato con tutti i restanti elementi dell’array puntati dall’indice i
- Ogni volta che l'elemento puntato da j è maggiore di quello a cui punta i, gli elementi vengono scambiati di posto

Insertion sort – risposta alternativa

I due indici scorrono l'array a due "velocità" diverse, cioè l'indice più lento avanza di una posizione quando l'indice più veloce ha già finito di scorrere tutti gli elementi rimanenti.

```
int vett[N] = {3, 6, 7, 5, 2};
int i, j, tmp;

for (j=0; j<N; j++) {
  for (i=j+1; i<N; i++) {
    if (vett[j] > vett[i]) {
      tmp = vett[j];
      vett[j] = vett[i];
      vett[i] = tmp;
    }
  }
}
```

12 - Data una matrice A di NxM elementi scrivere un algoritmo in C che visualizzi gli elementi della matrice, andando a capo al termine di ogni riga

12 - Data una matrice A di NxM elementi scrivere un algoritmo in C che visualizzi gli elementi della matrice, andando a capo al termine di ogni riga

```
int i,j;
for (i=0,i<N,i++){
  for (j=0;j<M;j++){
    printf("[%d] ",A[i][j])
  }
  printf("\n");
}
```

Parte pratica

1- Realizzare uno script che inizializzi una matrice NxM (con N e M parametri da assegnare all’inizio dello script) con valori casuali compresi tra 1 e 10

1- Realizzare uno script che inizializzi una matrice NxM (con N e M parametri da assegnare all'inizio dello script) con valori casuali compresi tra 1 e 10

```
N=5;

M=5;

A=randi(10,N,M);
```

2 - Si realizzi la function sommatoria: restituisce la somma degli elementi di un vettore

2 - Si realizzi la function sommatoria: restituisce la somma degli elementi di un vettore

```
function [s]=sommatoria(V)
% function sommatoria
%
% calcolo la somma degli elementi di un vettore passato come argomento
%
n=length(V);
s=0;
for k=1:n
    s=s+V(k);
end
end
```

3 - Si realizza la function massimo: restituisce il massimo degli elementi di un vettore e l’indice del valore massimo

3 - Si realizza la function massimo: restituisce il massimo degli elementi di un vettore e l'indice del valore massimo

```
function [ris]=massimo(V)
    n=length(V);
    ris(1)=V(1);
    ris(2)=1;
    for k=2:n
        if V(k)>ris(1)
            ris(1)=V(k);
            ris(2)=k;
        end
    end
end
```

4 - Tornando allo script venga fornito l’indice di riga la cui somma degli elementi risulta massima rispetto alle altre somme dei valori delle righe

```
N=5;

M=5;

A=randi(10,N,M);

????????
```

4 - Tornando allo script venga fornito l’indice di riga la cui somma degli elementi risulta massima rispetto alle altre somme dei valori delle righe

```
N=5;
M=5;
A=randi(10,N,M);
for i=1:N
   vet(i)=sommatoria(A(i,:));
end
ris=massimo(vet);
ris(2)
```

5- realizzare uno script che inizializzi un vettore di 21 punti equispaziati nell’intervallo 0, pi
6- calcoli una tabella di valori delle funzioni seno e coseno
7- si visualizzi la tabella con fprintf
8- si stampi la tabella su file

5- realizzare uno script che inizializzi un vettore di 21 punti equispaziati nell’intervallo 0, pi
6- calcoli una tabella di valori delle funzioni seno e coseno
7- si visualizzi la tabella con fprintf
8- si stampi la tabella su file

```
x=linspace(0,pi,21);
c=cos(x);
s=sin(x);
z=1:21;
disp('-----------------------------------');
fprintf('k\t x(k)\t cos(x(k))\t sin(x(k))\n');
disp('-----------------------------------');
fprintf('%d\t %3.2f\t %6.5f\t %6.5f\n',[z;x;c;s]);
save tabella.dat z x c s -ascii
```

9 - realizzare uno script che carichi da file la tabella e realizzi in una stessa finestra grafica due sottografici: seno e coseno

10 - salvare su file il grafico in formato jpeg

9 - realizzare uno script che carichi da file la tabella e realizzi in una stessa finestra grafica due sottografici: seno e coseno

10 - salvare su file il grafico in formato jpeg

```
% visualizzavalori.m
% Esempio di input da file con load
%
load tabella.dat
A=tabella;
Subplot(1,2,1);
plot(A(2,:),A(4,:))
Subplot(1,2,2);
plot(A(2,:),A(3,:))
print -djpeg grafico.jpg
```

Grazie per l’attenzione