A.A. 08/09

Fondamenti di Programmazione

(canale E-O)

Docente: Prof.ssa Tiziana Calamoneri
calamo@di.uniroma1.it

Esercitatore: Dott. Roberto Petroccia
petroccia@di.uniroma1.it

Pagina del corso:
http://twiki.di.uniroma1.it/twiki/view/Programmazione1/EO/WebHome

---

Esercitazione del 10/12/08

Indice

Cosa è la ricorsione?

Un algoritmo che richiama se stesso.

NON BASTA

Cosa è la ricorsione?

Un algoritmo che richiama se stesso.

NON BASTA

Funzione ricorsiva che stampa un array

```
void stampa (int V[], int n, int i) {
      if (i != 0)
         return;
      for (i = 0; i < n; i++) {
         printf("%d\n", V[i]);
      }
      stampa(V, n, 1);
}
```

Cosa è la ricorsione?

Un algoritmo che richiama se stesso risolvendo lo stesso problema di partenza ma su istanze più piccole.

Funzione ricorsiva che stampa un array

```
void stampa (int V[], int n, int i) {
      if (i >= n)
         return;
      printf("%d\n", V[i]);
      stampa(V, n, i+1);
}
```

Cosa è la ricorsione?

Una funzione ricorsiva PUO' contenere cicli al suo interno. Vedi SelectionSort ricorsivo mostrato nelle slide precedenti

Soluzioni Esercizi Esonero

Ex1 Compito A: Data una stringa alfanumerica (che termina con ‘\0’) memorizzata in un vettore di caratteri si stampi il numero di occorrenze per ciascuan cifra; la stampa deve avvenire per cifre crescenti.

Soluzioni Esercizi Esonero

Ex1 Compito A: Data una stringa alfanumerica (che termina con ‘\0’) memorizzata in un vettore di caratteri si stampi il numero di occorrenze per ciascuan cifra; la stampa deve avvenire per cifre crescenti.

Le cifre sono 10: da 0 a 9

Per capire se un carattere ‘c’ è una cifra controllo ‘0’ <= c <= ‘9’

Se ‘c’ è una cifra per ottenere il numero corrispondente int x = ‘c’ - ‘0’ (quest’ultima condizione non è necessaria)

Soluzioni Esercizi Esonero

Ex1 Compito A: Data una stringa alfanumerica (che termina con ‘\0’) memorizzata in un vettore di caratteri si stampi il numero di occorrenze per ciascuan cifra; la stampa deve avvenire per cifre crescenti.

Le cifre sono 10: da 0 a 9

Per capire se un carattere ‘c’ è una cifra controllo ‘0’ <= c <= ‘9’

Se ‘c’ è una cifra per ottenere il numero corrispondente int x = ‘c’ - ‘0’ (quest’ultima condizione non è necessaria)

Scompongo il problema:

1) Contare quante volte un carattere compare all’interno di un vettore di caratteri

2) Applicare l’algoritmo precedente a tutti i caratteri che sono cifre.

Soluzioni Esercizi Esonero

PSEUDOCODICE

Funzione ContaOccorrenze:

Input vettore di caratteri ed un carattere da controllare;

inizializzo un contatore a 0;

scorro il vettore se l'elemento del vettore è uguale al carattere da controllare incremento il contatore;

restituisco il valore del contatore;

Funzione Occorrenze:

Input vettore di caratteri V;

inizializzo un vettore A[10] di interi a 0 ed un indice i a 0;

Per ogni cifra k tra '0' e '9' chiama ContaOccorrenze(V, k) e memorizza il valore ritornato dalla funzione in A[i] ed incrementa i

scorri A e se A[i] > 0 stampa: La cifra i compare A[i] volte

Soluzioni Esercizi Esonero

CASI PARTICOLARI

Se il vettore è vuoto (solo carattere ‘\0’) non c’è bisogno di effettuare alcun controllo

CODICE

```
#include <stdio.h>
int ContaOccorrenze(char V[], char k) {
        int count = 0;
        int i = 0;
        while (V[i] != '\0') {
            if (V[i] == k) {
                count++;
            }
            i++;
        }
        return count;
}


int main() {
    leggi V;
    Occorrenze(V);
    return 0;
}
```

```
void Occorrenze (char V[]) {
    if (V[0] == '\0'){
        return;
    }
    int i = 0;
    char c;
    int A[10];
    for (i = 0; i < 10; i++) {
        A[i] = 0;
    }
    for (c = '0', i=0; c <= '9'; c++, i++) {
        A[i] = ContaOccorrenze(V, c);
    }
    for (i = 0; i < 10; i++) {
        if (A[i] > 0) {
            printf("la cifra %d compare %d volte\n",
                                i, A[i]);
        }
    }
}
```

Soluzioni Esercizi Esonero

COMPLESSITA’

Bisogna scorrere tutto il vettore di caratteri, se è lungo N vengono eseguite O(N) operazioni, oppure k*N operazioni dove k è una costante. Ossia ordine di N operazioni.

Soluzioni Esercizi Esonero

COMPLESSITA’

Bisogna scorrere tutto il vettore di caratteri, se è lungo N vengono eseguite O(N) operazioni, oppure k*N operazioni dove k è una costante. Ossia ordine di N operazioni.

CORRETTEZZA

Input = “3ab91cz33391a5”

Algoritmo chiama ContaOccorrenze(V, ‘0’), si confrontano tutti i caratteri del vettore con ‘0’, tale carattere non compare quindi la funzione restituisce il valore 0 ed A[0] resta uguale a 0. Poi viene chiamata ContaOccorrenze(V,’1’), il carattere 1 compare due volte (dopo tutti i controlli con gli elementi del vettore), A[1] = 2. ...

Si scorre A, A[0] = 0 quindi non stampa nulla, A[1] = 2 si stampa la cifra 1 compare 2 volte ....

Soluzioni Esercizi Esonero

Ex1 Compito B: Dato un vettore di interi non negativi, non necessariamente utilizzato per intero (si usi -1 come valore sentinella), si stampi il numero di occorrenze per ciascun valore minore di 10; la stampa deve avvenire per cifre crescenti.

Stessa soluzione appena vista senza caratteri ma con

int ContaOccorrenze(int V[], int k) e in Occorrenze abbiamo

```
for (i = 0; i <= 9; i++) {
    A[i] = ContaOccorrenze(V, i);
}
```

Ex1 Compito C: Data una stringa alfanumerica (che termina con '\0') memorizzata in un vettore di caratteri si stampi il numero di occorrenze per ciascuna lettera tra 'a' ed 'l'; la stampa deve avvenire per cifre crescenti.

Stessa soluzione appena vista (tra 'a' ed 'l' ci sono 10 caratteri) ma con

int ContaOccorrenze(int V[], int k) e in Occorrenze abbiamo

```
for (c = 'a', i=0; c <= 'l'; c++, i++) {
    A[i] = ContaOccorrenze(V, c);
}
```

Soluzioni Esercizi Esonero

Ex1 Compito D: Data una stringa alfanumerica (che termina con ‘\0’) memorizzata in un vettore di caratteri si stampino gli indici relativi ai caratteri uguali a quelli in posizione simmetrica rispetto alla metà delle stringa. Ossia ogni indice i il cui valore è uguale al valore in posizione n -1 - i (dove è la dimensione del vettore escluso ‘\0’).

Conta quanti elementi ci sono nella stringa

Controllo e stampo gli indici in posizione simmetrica

CODICE

```
#include <stdio.h>
int Conta(char V[]) {
        int i = 0;
        while (V[i] != '\0') {
            i++;
        }
        return i;
}


int main() {
    leggi V;
    IndiciSimmetrici(V);
    return 0;
}
```

```
void IndiciSimmetrici (char V[]) {
    if (V[0] == '\0'){
        return;
    }
    int i = 0;
    int n = Conta(V);
    for (i = 0; i < n/2; i++) {
        if (V[i] == V[n-1-i]){
            printf ("%d ", i);
        }
    }
    printf("\n").
}
```

Soluzioni Esercizi Esonero

Ex2 Compito B: Dati due vettori di interi A e B di lunghezza n, si ottenga di scambiare il primo elemento del vettore A con l'ultimo del vettore B, il secondo di A con il penultimo di B e così via. Per scambiare gli elementi si faccia uso di una funzione a cui siano passati solo gli elementi da scambiare e non l'intero vettore.

Definire una funzione scambia

Richiamare la funzione scambia sugli opportuni valori dei vettori

Soluzioni Esercizi Esonero

PSEUDOCODICE ITERATIVO

Funzione Scambia:

Input due interi a e b;

memorizza in a il valore di b ed in b il valore di a;

Funzione ScambiaVettori:

Input vettore di interi A e B lunghi n;

inizializza intero i a 0

scorri A e scambia A[i] con B[n-1-i] (utilizzando funzione Scambia) incrementando i ad ogni scambio.

Soluzioni Esercizi Esonero

PSEUDOCODICE Ricorsivo

Funzione Scambia:

Input due interi a e b;

memorizza in a il valore di b ed in b il valore di a;

Funzione ScambiaVettori:

Input vettore di interi A e B lunghi n, un intero i;

se i è uguale ad n esci, altrimenti

scambia A[i] con B[n-1-i] (utilizzando funzione Scambia)

richiama ScambiaVettori(A, B, n, i + 1)

Soluzioni Esercizi Esonero

CASI PARTICOLARI

Se n è pari a 0 non c’è bisogno di effettuare alcuno scambio

CODICE ITERATIVO

```
#include <stdio.h>
void Scambia(int *a, int *b) {
        int temp;
        temp = *a;
        *a = *b;
        *b = temp;
}


int main() {
    leggi A, B, n;
    ScambiaVettori(A, B, n);
    return 0;
}
```

```
void ScambiaVettori (int A[], int B[], int n) {
    if (n == 0){
        return;
    }
    int i = 0;
    for (i = 0; i < n; i++) {
        Scambia(&A[i], &B[n-1-i]);
    }
}
```

CODICE RICORSIVO

```
#include <stdio.h>
void Scambia(int *a, int *b) {
        int temp;
        temp = *a;
        *a = *b;
        *b = temp;
}


int main() {
    leggi A, B, n;
    ScambiaVettori(A, B, n, 0);
    return 0;
}
```

```
void ScambiaVettori (int A[], int B[], int n, int i) {
    if (i >= n){
        return;
    }
    Scambia(&A[i], &B[n-1-i]);
    ScambiaVettori(A, B, n, i+1);
}
```

Soluzioni Esercizi Esonero

COMPLESSITA’

Bisogna scorrere tutto il vettore di caratteri, se è lungo N vengono eseguite O(N) operazioni, oppure k*N operazioni dove k è una costante. Ossia ordine di N operazioni.

Soluzioni Esercizi Esonero

COMPLESSITA’

Bisogna scorrere tutto il vettore di caratteri, se è lungo N vengono eseguite O(N) operazioni, oppure k*N operazioni dove k è una costante. Ossia ordine di N operazioni.

CORRETTEZZA ITERATIVA

Input A = {1 2 3 4} B = {5 6 7 8} n = 4
n != 0 viene eseguito il for.
Quando i = 0 viene scambiato A[0] = 1 con B[n-1] = 8.
Quando i = 1 viene scambiato A[1] = 2 con B[n-2] = 7.
.....
Quando i = n-1 viene scambiato A[n-1] = 4 con B[n-1] = 5.

Soluzioni Esercizi Esonero

COMPLESSITA’

Bisogna scorrere tutto il vettore di caratteri, se è lungo N vengono eseguite O(N) operazioni, oppure k*N operazioni dove k è una costante. Ossia ordine di N operazioni.

CORRETTEZZA ITERATIVA

Input A = {1 2 3 4} B = {5 6 7 8} n = 4
n != 0 viene eseguito il for.
Quando i = 0 viene scambiato A[0] = 1 con B[n-1] = 8.
Quando i = 1 viene scambiato A[1] = 2 con B[n-2] = 7.
.....
Quando i = n-1 viene scambiato A[n-1] = 4 con B[n-1] = 5.

CORRETTEZZA ITERATIVA

Input A = {1 2 3 4} B = {5 6 7 8} n = 4
i = 0 < n viene eseguito lo scambio tra A[0] = 1 con B[n-1] = 8 e chiamato con i+1
i = 1 < n viene eseguito lo scambio tra A[1] = 2 con B[n-2] = 7 e chiamato con i+1
.....
i = n-1 < n viene eseguito lo scambio tra A[n-1] = 4 con B[0] = 5 e chiamato con i+1
i = n esco e torno alle chiamate precedenti ma nessun ulteriore operazione viene eseguita

Soluzioni Esercizi Esonero

Ex3 Compito A: Sono definite 4 variabili intere, dividendo, divisore, quoziente e resto, come variabili locali alla funzione main. Qui alle prime due viene assegnato un valore mentre le seconde due vengono iniazilizzate a 0. Progettare una funzione C a cui vengano opportunamente passati dei parametri, che esegua la divisione tra dividendo divisore e restituisca alla funzione main quoziente e resto , che vanno memorizzate nelle omonime variabili

Soluzioni Esercizi Esonero

Ex3 Compito A: Sono definite 4 variabili intere, dividendo, divisore, quoziente e resto, come variabili locali alla funzione main. Qui alle prime due viene assegnato un valore mentre le seconde due vengono iniazilizzate a 0. Progettare una funzione C a cui vengano opportunamente passati dei parametri, che esegua la divisione tra dividendo divisore e restituisca alla funzione main quoziente e resto , che vanno memorizzate nelle omonime variabili

Bisogna controllare che il divisore sia diverso da 0 altrimenti una divisione per 0 genera un errore. In tal caso lasciamo resto e quoziente a 0

Soluzioni Esercizi Esonero

PSEUDOCODICE

Funzione Esercizio3:

Input dividendo, divisore, quoziente e resto;

se il divisore è 0 esci;

esegui la divisione tra dividendo e divisore e memorizza il quoziente ed il resto nelle omonime variabili.

Soluzioni Esercizi Esonero

CASI PARTICOLARI

Se il divisore è zero non si deve effettuare la divisione

CODICE

```
#include <stdio.h>
void Esercizio3(int dividendo,int divisore, int* quoziente, int* resto ) {
        if (divisore == 0)
            return;
        *quoziente = dividendo / divisore;
        *resto = dividendo % divisore;
}


int main() {
    leggi dividendo, divisore;
    int quoziente = 0;
    int resto = 0;
    Esercizio3(dividendo, divisore, &quoziente, &resto);
    return 0;
}
```

Soluzioni Esercizi Esonero

CORRETTEZZA

Input: Dividendo = 9

Divisore = 2

Divisore diverso da 0, metto in quoziente il valore 9/2 = 4 e in resto 9%2=1

Input: Dividendo = 9

Divisore = 0

Divisore uguale a 0, esco

DOMANDE???

STRUCT

Attraverso le strutture potete mettere insieme più cose di tipo diverso.

STRUCT

Attraverso le strutture potete mettere insieme più cose di tipo diverso.

Esempio:

```
struct anagrafe {
    char[100] nome;
    char[100] cognome;
    int eta;
    int numeroFigli;
    char sesso;
};
```

STRUCT

```
struct anagrafe {
        char[100] nome;
        char[100] cognome;
        int eta;
        int numeroFigli;
        char sesso;
        struct anagrafe x;
};
```

ERRORE, una struttura non può contenere un'istanza di se stessa, perché la sto definendo e quindi non so quanta memoria mi occorre

STRUCT

```
struct anagrafe {
        char[100] nome;
        char[100] cognome;
        int eta;
        int numeroFigli;
        char sesso;
        struct anagrafe x;
};
```

ERRORE, una struttura non può contenere un'istanza di se stessa, perché la sto definendo e quindi non so quanta memoria mi occorre

```
struct anagrafe {
        char[100] nome;
        char[100] cognome;
        int eta;
        int numeroFigli;
        char sesso;
        struct anagrafe * x;
};
```

CORRETTO, x è un puntatore quindi un indirizzo di memoria e so quanta memoria serve per contenere un indirizzo (macchine a 32 bits occorrono 32 bits ossia 4 bytes, macchine a 64 buts occorrono 8 bytes).

STRUCT

```
struct anagrafe {
        char[100] nome;
        char[100] cognome;
        int eta;
        int numeroFigli;
        char sesso;
        struct anagrafe* nome;
};
```

ERRORE, una struttura non può contenere due campi con lo stesso nome

STRUCT

```
struct anagrafe {
        char[100] nome;
        char[100] cognome;
        int eta;
        int numeroFigli;
        char sesso;
};
```

Per sapere quanta memoria serve per un elemento di tipo struct anagrafe usiamo l'istruzione sizeof(struct anagrafe), ci ritorna il numero di bytes necessari (sarà la somma dei bytes necessari ad ogni campo della struttura).

STRUCT

```
struct anagrafe {
        char[100] nome;
        char[100] cognome;
        int eta;
        int numeroFigli;
        char sesso;
};
```

Come dichiaro un elemento di questa struttura, allo stesso modo dei tipi primitivi.

Per int avevamo: int x, oppure int x[N] per i vettori

Ora abbiamo:

struct anagrafe x, struct anagrafe x[N]

STRUCT

Possiamo anche utilizzare l'istruzione typedef per definire un nuovo tipo. Ad esempio

```
struct anagrafe {
        char[100] nome;
        char[100] cognome;
        int eta;
        int numeroFigli;
        char sesso;
};
typedef struct anagrafe pippo
```

```
typedef struct anagrafe {
        char[100] nome;
        char[100] cognome;
        int eta;
        int numeroFigli;
        char sesso;
} pippo;
```

Ci dice che abbiamo definito una struttura e che definiamo un tipo per questa struttura che si chiama pippo:

Scrivere struct anagrafe x è equivalente a scrivere pippo x

Ossia ogni qual volta leggiamo pippo possiamo sostituire ad esso struct anagrafe

STRUCT

In genere si scrive

```
typedef struct anagrafe {
        char[100] nome;
        char[100] cognome;
        int eta;
        int numeroFigli;
        char sesso;
} anagrafe;
```

Per non introdurre confusione, quindi definiamo una struttura struc anagrafe e diciamo che il tipo di tale struttura ha nome anagrafe.

STRUCT

Le uniche operazioni valide sulle strutture sono:

- Assegnare una struttura ad un'altra struttura dello stesso tipo
- rilevare l'indirizzo (&) di una variabile di tipo struttura
- accedere ai membri di una variabile di tipo struttura
- usare l'operatore sizeof per determinare la dimensione di una struttuta

N.B. le strutture non si possono confrontare con == e != (Vanno confrontati i singoli elementi che compongono la struttura)

STRUCT

Strutture e funzione

- Le strutture possono essere passate alle funzioni:
  Passando l'intera struttura
  Passando singoli membri
  In entrambi i modi il passaggio è per valore

- Per passare le strutture per riferimento:
  Passare l'indirizzo

- Vettori:
  un vettore di strutture è passato per riferimento (è un vettore!)
  creando una struttura contenente un vettore e passandola ad una funzione si ottiene il passaggio di un vettore per valore (se non si passa il riferimento della struttura e si modifica il vettore all'uscita della funzione il vettore della struttura rimane lo stesso)

STRUCT

```
typedef struct anagrafe {
        char[100] nome;
        char[100] cognome;
        int eta;
        int numeroFigli;
        char sesso;
} anagrafe;
```

```
anagrafe x;

x.nome = “Roberto”;

x.cognome = “Petroccia”;

x.eta = 27;
```

STRUCT

```
typedef struct anagrafe {
        char[100] nome;
        char[100] cognome;
        int eta;
        int numeroFigli;
        char sesso;
} anagrafe;
```

```
anagrafe* x;
```

```
x->nome = “Roberto”;          oppure    (*x).nome = “Roberto”;
x->cognome = “Petroccia”;     oppure    (*x).cognome = “Petroccia”;
x->eta = 27;                  oppure    (*x).eta = 27;
```

STRUCT Esempi

Scrivere una funzione che dato un vettore di interi restituisca qual è il valore minimo, qual è il valore massimo e qual è la media dei valori del vettore.

```
typedef struct mat {
        int min;
        int max;
        double media;
} mat;


mat funzione (int V[], int n) {
        mat m;
        m.min = trovaMinimo(V,n);
        m.max = trovaMassimo(V,n);
        m.media = trovaMedia(V,n);
        return m;
}
```

STRUCT Esempi

Scrivere una funzione che dato un vettore di interi restituisca qual è il valore minimo, qual è il valore massimo e qual è la media dei valori del vettore.

```
typedef struct mat {
        int min;
        int max;
        double media;
} mat;



void funzione (int V[], int n, mat* m) {
        m->min = trovaMinimo(V,n);
        m->max = trovaMassimo(V,n);
        m->media = trovaMedia(V,n);
}
```

STRUCT Esempi

Scrivere una funzione che dato un vettore di coppie restituisca qual è la somma d
tutti i valori del vettore.

```
typedef struct coppia {
        int val1;
        int val2;
} coppia;
```

```
int sommaCoppie (coppia V[], int n) {
    int somma = 0;
    int i;
    for (i = 0; i < n; i++) {
        count += (V[i]).val1;
        count += (V[i]).val2;
    }
    return somma;
}
```

STRUCT Esempi

Scrivere una funzione che dato un vettore di coppie restituisca qual è la somma d
tutti i valori del vettore.

```
typedef struct coppia {
        int val1;
        int val2;
} coppia;
```

```
int sommaCoppie (coppia V[], int n) {
    int somma = 0;
    int i;
    for (i = 0; i < n; i++) {
        count += (V[i]).val1;
        count += (V[i]).val2;
    }
    return somma;
}
```

```
int sommaCoppieR (coppia V[], int n) {
        if (n <= 0)
            return 0;
        return V->val1 + V->val2 +
                sommaCoppieR(V+1, n-1);
}
```

DOMANDE???

Esercizi su struct

1. Scrivere una funzione che presa in input una matrice di double, restituisca in una struttura l'indice di riga e di colonna del valore minimo e l'indice di riga e di colonna del valore massimo.

La struct è: (Potete usare typedef se volete)

```
struct indici {
        int min_riga;
        int min_colonna;
        int max_riga;
        int max_colonna;
};
```

Prototipi della funzione (due versioni):

```
struct indici funzione (int rig, int col, int M[][col]);
void funzione2(int rig, int col, int M[][col], struct indici* x);
```

2. Scrivere una funzione che presa in input un vettore di interi restituisca la somma di tutti i valori e la media dei valori del vettore in una struttura. Implementare sia funzione con tipo di ritorno la struttura che con tipo di ritorno void. Sia versione iterativa che ricorsiva.

3. Scrivere una funzione legga da tastiera un valore n, crei un vettore di struct (contenenti due array di caratteri lunghi 20 per i campi nome e cognome ed un intero per l'età). Legga da tastiera n volte un intero (eta) e due stringhe (nome e cognome) e li memorizzi nella struttura del vettore. Le stringhe vanno lette carattere per carattere con la funzione getchar() finchè non trovate uno '\n', va anche inserito il carattere di fine stringa. La funzione stampa poi i valori inseriti nel vettore.

10/12/08 - Roberto Petroccia: petroccia@di.uniroma1.it