Esercitazioni di Fondamenti di Informatica - Lez. 4 15/10/2020

Principali errori di codifica

I. rientro del testo o indentazione

don't

```
int main()
{int a=0;


if(a==0)  {
a=b;
}



if(a>0)  {
     ...
}
     else  {
          ...
     }
```

do

```
int main()
{
     int a=0;
     ...

     if(a==0)
     {
        a=b;
     }

     if(a>0)
     {
        ...
     }
     else
     {
          ...
     }
}
```

do (*alternativa*)

```
int main(){
     int a=0;
     ...

     if(a==0){
        a=b;
     }

     if(a>0){
        ...
     }
     else{
          ...
     }
}
```

II. errori logici o di costrutto

don't

```
if(a>0)
{
     ...
}
else  (a<=0)
{
     ...
}
```

La negazione di a>0 è già minore o uguale a 0.
Difatti è inutile anche:

```
if(a>0)
{
   ...
}
else
{
    if (a<0)
    {
       ...
    }
    else
    {
       if (a==0)
```

do

```
if(a>0)
{
     ...
}
else
{
     ...
}


if(a>0)
{
   ...
}
else
{
    if (a<0)
    {
       ...
    }
    else
    {
       ...
```

III. errori logici di assegnazione

don't

```
int a=1, b=2;
....
if (a = 0)
{
}

..
if (b = a)
{
}

..

int trovato = 0;
while(trovato = 0)
{
}
```

Quanto vale qui *b* alla fine?
ATTENZIONE: "=" assegna, "==" confronta.

do

```
int a=1,b=2;
....
if (a == 0)
{
}

..
if (a == b)
{
}

..

int trovato = 0;
while(trovato == 0)
{
}
```

scanf("%d", A[i]); ----- ricorda & --> scanf("%d", &A[i]);

scanf("%c", c); — char ha bisogno di uno spazio -> scanf(" %c", c);

IV. errori sui cicli

don't

```
while(x>0);
{
}

for (a=1; a<100; a++);
{
}
```

Il ";" crea un blocco di statement del ciclo vuoto

do

```
while(x>0)
{
}


for (a=1; a<100; a++)
{
}
```

Esercizi sui tipi in C

*I codici e le soluzioni degli esercizi sono nella cartella **codice4***

1. Dato il seguente codice, trovare gli errori, correggerli e indicare cosa si ottiene come output.

```
#include<stdio.h>
int main(){
int i= 65;
int j = 'A';
int k=“$";
int u=i+k;
char v = 122;
        printf("%d %d %d %d %c %d %c\n",i ,j ,k, u, u, v, v);
}
```

Soluzione

```
#include<stdio.h>
int main(){
int i = 65;
int j = 'A';
int k= '$';     <---con apici indica il valore ASCII
int u=i+k;
char v = 122;
 printf("%d %d %d %d %c %d %c\n",i ,j ,k,u,u, v, v);
}
```

Il programma stampa:

_ Il numero 65

_ Il codice ASCII (intero) del carattere 'A'

_ Il codice ASCII (intero) del carattere '$'

_ Il numero dato dalla somma tra 65 e il codice ASCII (intero) del carattere '$'

_ il carattere corrispondente al codice ASCII dato dal numero precedente

_ il numero 122

_ il carattere corrispondente al codice ASCII 122

2. Si scriva un programma che dati in ingresso N intero > 0 produce in uscita un triangolo isoscele e rettangolo di N righe composto da '*' (nota: uso di cicli innestati, una per le righe ed una per le colonne)

esempio
Inserire la lunghezza del lato: 4

```
*                       *
*  *        oppure.     * *
*  *  *               *  *  *
*  *  *  *          *  *  *  *
```

```c
#include <stdio.h>

int main() {
int n, i, j;
i = 0, j = 0;

printf("Inserire la lunghezza del lato: ");
scanf("%d", &n);

while (i < n) {
   while (j <= i) {
     printf("* ");
     j++;
   }
   printf("\n");
   j = 0;
   i++;
}
return 0;
}
```

3. **Definisci un nuovo tipo di dato contenente un intero, un carattere e un numero reale, in tre modi diversi:**
   (A) definire una struttura senza definire un nuovo tipo;
   (B) definire una struttura senza definire un nuovo tipo e con una nuova variabile;
   (C) definire una struttura definendo un nuovo tipo. Provare a copiare una variabile struttura in un altra e modificarla.
   Stampare il risultato della struttura ottenuta con ogni definizione.

```c
#include < stdio.h>
int main() {

 /* modi alternativi per definire una struct */

 /* caso A definire una struttura senza definire un nuovo tipo */
 struct A {
     int  intero;
     char carattere;
     float reale;
  } ;
 /* se voglio dichiarare una variabile di tipo prova devo usare
  una struct prova */

 struct A prova;

     prova.intero = 1;
     prova.carattere = 'P';
     prova.reale = 2.4;

/* stampiamo il contenuto di prova:*/
    printf("\n A.\n");
    printf("\n intero: %d\n", prova.intero);
    printf("carattere: %c\n", prova.carattere);
    printf("reale: %f\n", prova.reale);

 /* B- definire una struttura senza definire un nuovo tipo e con una
nuova variabile
  N.B. subito dopo la definizione della struct posso
        definire una serie di variabili della struttura */

 struct B {
    int intero;
    char carattere;
    float reale;
 } prova2;

    prova2.intero = 1;
    prova2.carattere = 'P';
    prova2.reale = 2.4;

    printf("\nB)\n");
    printf("\nintero: %d\n", prova2.intero);
    printf("carattere: %c\n", prova2.carattere);
```

```
        printf("reale: %f\n", prova2.reale);


    /*C definire una struttura definendo un nuovo tipo */

        typedef struct{
            int intero;
            char carattere;
            float reale;
        } C;

        C prova3,prova4;

        prova3.intero = 1;
        prova3.carattere = 'P';
        prova3.reale = 0.24;

        printf("\nC)\n");
        printf("\n intero: %d\n", prova3.intero);
        printf("carattere: %c\n", prova3.carattere);
        printf("reale: %f\n", prova3.reale);

    /* Provare a copiare una variabile struttura in un altra ,
        modificarla e stampare il risultato.  */

      // C prova4;
        prova4 = prova3;
        prova4.intero = 4356;

        printf("\nC) struttura copiata e modificata\n");
        printf("\n intero: %d\n", prova4.intero);
        printf("carattere: %c\n", prova4.carattere);
        printf("reale: %f\n", prova4.reale);
    }
```

4. **Definire una struttura che permetta di contenere una serie di dati di squadre calcistiche: nome squadra, codice identificativo squadra, goal fatti, goal subiti.**
   Data una sequenza di squadre, stampare i nomi delle squadre che hanno fatti più goal di quanti ne abbiano subiti

Soluzione:

```
#include<stdio.h>
#define MAXS 20   /* Lunghezza massima dei nomi delle squadre */
#define MAXA 6   /*  Massimo numero di squadre memorizzabili  */


/* Definisco un tipo per i campi testuali
   N.B. C non ha un tipo stringa!!! */
typedef char String [MAXS]; // qui salveremo i nomi delle squadre
```

```c
/* Definisco una struttura per
   memorizzare una squadra */

typedef struct {
     String nome;
     int ID;
     int goal_fatti;
     int goal_subiti;
     } Squadra;

/* Dichiarazione della sequenza di squadre
   come array di strutture */

typedef Squadra Campionato[MAXA];

int main(){

     Campionato campionato = {
          {"Juventus",1,10,12},
          {"Milan",8,7,6},
          {"Inter",10,13,11},
          {"SPAL",2,9,10},
          {"Pizzighettone",5,8,4},
          {"Udinese",14,5,7}
     };
     for (int i=0; i<MAXA; i++){
          if(campionato[i].goal_fatti > campionato[i].goal_subiti){
               printf("%s differenza goal: %d \n",
               campionato[i].nome,
               campionato[i].goal_fatti-campionato[i].goal_subiti);
          }

      }
}
```

5. **Definire le strutture dati per memorizzare i dati anagrafici di un gruppo di persone: nome, cognome e data di nascita. La data di nascita deve essere nel formato 01-GEN-1970.**
   Identificare chi è la persona più giovane dell'anagrafica e stamparne nome e cognome.

```c
#include<stdio.h>
#define MAXS 30 /* lunghezza massima campi testuali */
#define MAXP 3  /* numero massimo persone in anagrafica */

/*  Enumero tutti i mesi */
typedef enum{GEN=1, FEB, MAR, APR, MAG, GIU,
          LUG, AGO, SET, OTT, NOV, DIC} Mese;

/*  Definisco un tipo per i campi testuali */
typedef char String [MAXS];

/*  Struttura per memorizzare una data_nascita */
typedef struct{
int giorno;
Mese mese;
int anno;
} Data;

/*  Struttura con i dati di una persone */
typedef struct{
String nome;
String cognome;
Data data_nascita;
} Persona;

/*  Dichiarazione dell'anagrafica come collezione di strutture persona */
typedef Persona Anagrafica[MAXP];

int main(){
     Anagrafica anagrafica = {
          {"Mario","Rossi",{10,APR,1980}},
          {"Filippo","Bianchi",{10, APR,1970}},
          {"Maria","Verdi",{5,APR,1990}}
     };

     int i;

     /*  Calcolo della persona più giovane */

     /*  Inizializzo il risultato con la */
     /*  prima persona di anagrafica */
     Persona piu_giovane = anagrafica[0];

     /*  Cicliamo attraverso tutte le persone */
     /*  contenute in anagrafica */
     for(i=0;i<MAXP;i++){
          /*  confronto sull'anno */
          if(piu_giovane.data_nascita.anno !=
anagrafica[i].data_nascita.anno){
```

```
if(anagrafica[i].data_nascita.anno>piu_giovane.data_nascita.anno){
                    piu_giovane = anagrafica[i];
               }
          }
          /*  confronto sul mese */
          else{
               if(piu_giovane.data_nascita.mese !=
anagrafica[i].data_nascita.mese){

if(anagrafica[i].data_nascita.mese>piu_giovane.data_nascita.mese){
                         piu_giovane = anagrafica[i];
                    }
          }
          /*  confronto sul giorno */

               else{
                if(piu_giovane.data_nascita.giorno !=
anagrafica[i].data_nascita.giorno){

if(anagrafica[i].data_nascita.giorno>piu_giovane.data_nascita.giorno){
                              piu_giovane = anagrafica[i];
                    }
                }
             }
        }
     }/*  fine ciclo for */

     printf("La persona piu' giovane e' %s
%s\n",piu_giovane.nome,piu_giovane.cognome);
}
```

6. **Dichiara una variabile intera *lunghezza* ed una stringa *s* di 30 caratteri.**
   Inverti la stringa carattere per carattere con un ciclo.
   Stampa la stringa con printf(”%s”, …)

```
#include<stdio.h>
#include<string.h> /* contiene funzioni specifiche per le
stringhe */
int main() {

     char s[30];
     int lunghezza;

     char temp; /* variabile temporanea */
```

```
    int i;

    printf("Inserire una stringa: \n");
    scanf("%s", s);

    lunghezza = strlen(s);

    for (i = 0; i < lunghezza / 2; i++) {
        temp = s[lunghezza - i - 1];
        s[lunghezza - i - 1] = s[i];
        s[i] = temp;
    }
    printf("\n%s<------ parola invertita\n", s);
}
```

7. **Si scriva un programma che legga da input due array A di 10 elementi e B di 5 elementi.**
   Il programma stampi: “CONTIENE” se A contiene la sequenza contigua dei numeri di B.
   ES: A= [3,4,66,77,88,9,33,11, 66,100] B=[77,88,9,33,11]
   output: “CONTIENE" ma non stampa nulla per B = [77,88,9,34,11]

soluzione

```
#include <stdio.h>
#define MAX 10
//
int main() {
    int A[MAX] = { 3,4,66,77,88,9,33,11, 66,100 };
    int B[MAX / 2] = { 77,88,9,343,11 };
// per semplicita mettiamo il calcolo vero e proprio senza lettura /
scanf...
    int i, j = 0;
 // con lettura  da input
/*
    for (i = 0; i < MAX; i++) {
        scanf("%d", &A[i]);
    }
    for (i = 0; i < MAX / 2; i++) {
        scanf("%d", &B[i]);
    }
*/
    for (i = 0; i < MAX && j < MAX/2; i++) {

        if (A[i] == B[j]) {
            j++;
        }
        else {
```

```
            j = 0;
        }
    }
    if (j == MAX / 2)
        printf("CONTIENE\n");
    else
        printf("non CONTIENE\n");
 return 0;
}
```