**sizeof();**

La funzione predefinita sizeof(...) fornisce la lunghezza in byte di un qualsiasi tipo o variabile.

In questo caso stampa su schermo 4 perché è la sizeof di un intero.

#include<stdio.h>

int main(){

printf(“%d\n”,sizeof(int));

return 0;

}

Scrivere un programma che acquisisca l’età da tastiera e stabilisca se sei maggiorenne oppure no.

#include<stdio.h>

int main(){

printf(“Scrivi la tua età: \n”);

int x;

scanf(“%d”,&x);

if(x>=18)

printf(“Sei maggiorenne! \n”);

else //superfluo

printf(“Non sei maggiorenne. \n”);

return 0;

}

**switch(giorno){..}**

Scrivere un programma che stampa in base al numero da 1 a 7 il giorno della settimana.

#include<stdio.h>

int main(){

int giorno;

scanf(“%d”,&giorno);

switch(giorno){

case 1: printf(“Lunedì \n”);

break;

case 2: printf(“Martedì \n”);

break;

case 3: printf(“Mercoledì \n”);

break;

case 4: printf(“Giovedì \n”);

break;

case 5: printf(“Venerdì \n”);

break;

case 6: printf(“Sabato \n”);

break;

case 7: printf(“Domenica \n”);

break;

default: printf(“Weekend \n”);

}

return 0;

}

**switch(mese){..}**

Scrivere un programma che stampa in base al numero da 1 a 7 il giorno della settimana.

#include<stdio.h>

int main(){

int mese;

scanf(“%d”,&mese);

switch(mese){

case 1: printf(“Gennaio\n”);

break;

case 2: printf(“Febbraio\n”);

break;

case 3: printf(“Marzo\n”);

break;

case 4: printf(“Aprile\n”);

break;

case 5: printf(“Maggio\n”);

break;

case 6: printf(“Giugno\n”);

break;

case 7: printf(“Luglio\n”);

break;

case 8: printf(“Agosto\n”);

break;

case 9: printf(“Settembre\n”);

break;

case 10: printf(“Ottobre\n”);

break;

case 11: printf(“Novembre\n”);

break;

case 12: printf(“Dicembre\n”);

break;

default: printf(“Anno \n”);

}

return 0;

}

Scrivere un programma che acquisisca l’età da tastiera e stabilisca se sei maggiorenne oppure no.

#include<stdio.h>

int main(){

printf(“Scrivi la tua età: \n”);

int x;

scanf(“%d”,&x);

if(x>=18)

printf(“Sei maggiorenne! \n”);

else //superfluo

printf(“Non sei maggiorenne. \n”);

return 0;

}

Scrivere un programma che dato un numero n=1 lo moltiplichi per -2 ciclicamente fino a non uscire dall’intervallo (-50, 150)

#include<stdio.h>

int main(){

int n;

for(n=1; n>-50 && n<150; n\*=-2)

printf(“%d\n”,n);

return 0;

}

Scrivere un programma che legga da terminale dei numeri interi fino a che la loro somma non supera 100, stamparne la somma finale.

#include<stdio.h>

int main(){

int n,somma=0;

while(somma<100){ //a priori non sappiamo quanti numeri verranno letti

scanf(“%d”,&n);

somma+=n;

}

printf(“%d\n”,somma);

return 0;

}

Scrivere un programma che stampi 0 se n è un numero primo altrimenti 1.

#include<stdio.h>

int main(){

int n;

scanf("%d",&n);

int i=2;

while(n%i && i<n){ /\*oppure while(n%i!=0 && i<n){\*/

i++;

/\*printf("%d\n",i);\*/

}

printf("%d\n",i<n);

return 0;

}

Scrivere un programma che riconosca i numeri primi stampando 0, altrimenti stampando 1 in un intervallo (0,10).

#include<stdio.h>

int main(){

int j, i=2, n;

for(j=0; j<10; j++)

scanf(“%d”,&n)

while(n%i && i<n)

i++;

printf(“%d\n”,i<n)

return 0;

}

//Scrivere un programma che legga interi positivi fino a che ne trova uno negativo e lo ristampa.

#include<stdio.h>

int main(){

int n;

do

scanf("%d",&n);

while(n>=0);

printf("%d\n",n);

return 0;

}

Scrivere un programma che dichiarati dei valori con un array ne calcoli la media.

#include<stdio.h>

int main(){

int eta[]={23,24,17,27,25,24,24}

int n=7, i, somma=0;

double media=0;

for(i=0; i<n; i++)

somma+=eta[i];

media=(double)somma/n;

printf(“%f\n”,media);

return 0;

}

Scrivere un programma che date le temperature di una settimana ne stampi la prima negativa.

#include<stdio.h>

int main(){

double temperature[]={2,5,5,-1,3,0,-2};

int n=7, i=0;

while(i<n && temperature[i]>=0)

i++;

printf(“%f\n”,temerature[i]);

return 0;

}

Scrivere un programma che legga da tastiera i valori di un array e li stampi nell’ordine inverso.

#include<stdio.h>

int main(){

int a[7], n,i;

for(i=0; i<7; i++){

scanf(“%d”,&n);

a[i]=n;

}

for(i=6; i>=0; i--) printf(“%d\n”,a[i]);

return 0;

}

Scrivere un programma che calcoli il quadrato di un numero tramite la chiamata di una funzione

#include<stdio.h>

int square(int x){

return x\*x;

}

int main(){

int n;

scanf(“%d”,&n);

printf(“%d\n”,square(n));

return 0;

}

Scrivere un programma che crea dinamicamente un array la cui dimensione viene immessa da tastiera.

#include<stdio.h>

#include<stdlib.h>

int main(){

int n,i;

scanf(“%d”,&n);

int \*a; //oppure int \*a=NULL;

a=(int \*)malloc(n\*sizeof(int)); //alloco dinamicamente lo spazio per n interi, che se li voglio avere tutti inizializzati a 0 allora avrei dovuto usare la calloc

if(a!=NULL){ //controllo che l’allocazione sia andata a buon fine

for(i=0; i<n; i++)

scanf(“%d”,&a[i]); //oppure scanf(“%d”,&(a+i));

}

else{...} //gestione nel caso di fallimento dell’allocazione

return 0;

}

Scrivere un esempio di struct, typedef e enum.

struct gatto{

int eta;

double peso;

double cibo[7];

};

//se vogliamo che un tipo abbia un range limitato e predefinito di valori

enum colore\_gatto{ //oppure typedef enum{

rosso, rosso,

nero, nero,

tigrato, tigrato,

}; }colore\_gatto;

int main(){

enum colore\_gatto colore;

colore=rosso;

return 0;

}

struct citta{

int n\_cittadini;

double latitudine;

double longitudine;

};

struct nazione{

struct \*citta capitale; //non poteva essere fatto inversamente perche’ nazione segue

double superficie;

int fuso\_orario;

};

struct persona{

struct persona \*madre;

struct persona \*padre;

int anno\_nascita;

};

struct studente\_struct{ //oppure typedef struct{

int anno\_nascita; int anno\_nascita;

int matricola; int matricola;

}; }studente;

typedef struct studente\_struct studente;

int main(){

studente Giorgio;

Giorgio.matricola=666;

return 0;

}

void aggiorna\_peso(struct gatto\* gatto, double peso){

(\*gatto).peso=peso; //oppure gatto->peso=peso;

};

int main(){

struct gatto felix;

struct gatto luna;

felix.eta=34; //per accedere sia in lettura o scrittura ai campi della struct

felix.peso=6.5;

luna.eta=felix.eta-5;

luna.cibo[1]=14.5;

luna.peso=15.3;

if(luna.peso>10)

printf(“luna ha mangiato troppe crocchette.\n”);

int i;

double somma\_cibo=0;

felix.cibo[0]=0.2;

felix.cibo[1]=0.1;

…

for(i=0; i<7; i++)

somma\_cibo+=felix.cibo[i];

printf(“In 7 giorni Felix ha mangiato: %f kg. \n”,somma\_cibo);

struct persona Ornela;

struct persona Hasan;

struct persona Lavdije;

Ornela.anno\_nascita=1996;

Ornela.madre=&Lavdije;

Ornela.padre=&Hasan;

Ornela.madre->anno\_nascita=1964; //è equivalente alla scrittura che segue

\*(Ornela.padre)anno\_nascita=1963;

struct studente\* Ornelina;

Ornelina=malloc(sizeof(struct studente)); //allocazione dinamica

Ornelina->anno\_nascita=1996;

struct studente \*studenti\_inf;

studenti\_inf=calloc(10,sizeof(struct studente));

studenti\_inf[0]->matricola=536295;

studenti\_inf[0]->anno\_nascita=1991;

return 0;

}

oppure

int main(){

struct gatto felix={34,6.5,{0.2,0.1,0.3,0.3,0.2,0.5,0.3}};

aggiorna\_peso(&felix, 6.2);

return 0;

}

//oppure

#include<stdio.h>

#include<stdlib.h>

struct point{

int x;

int y;

};

//una funzione puo' restituire una struct, esattamente come con gli altri tipi

struct point createPoint(int x, int y){

struct point pp;

pp.x=x;

pp.y=y;

return pp;

}

//una struct puo' essere passata (per valore) come parametro ad una funzione

struct point sumPoint(struct point p1, struct point p2){

p1.x=p1.x+p2.x;

p2.y=p1.y+p2.y;

return p1; //MA IL CHIAMANTE VEDE POI LE MODIFICHE?

}

void swapPoint(struct point \*pop){

int tmp=pop->x;

pop->x=pop->y;

pop->y=tmp;

}

int main(){

struct point p; //dichiarazione

p.x=10; //accesso ai campi con " . "

p.y=14;

printf("Le coordinate sono x: %d, y: %d \n", p.x, p.y);

struct point \*puntatore; //dichiarazione di un puntatore a struct

puntatore=(struct point \*)malloc(sizeof(struct point)); //allocazione dinamica della variabile puntatore

(\*puntatore).x=20;

//oppure puntatore->x=20;

//oppure puntatore[0].x=20;

//MA NO (puntatore).x=20;

printf("%d\n",(\*puntatore).x);

struct point \*pop=(struct point \*)malloc(sizeof(struct point));

pop->x=5;

pop->y=2;

swapPoint(pop);

printf("I valori sono adesso: x %d e y %d \n",pop->x, pop->y);

return 0;

}

Creazione di una lista (con 3 elementi)

include<stdio.h>

include<stdlib.h>

typedef struct Il\_node\_struct{

int info;

struct Il\_node\_struct \*next;

} Il\_node;

int main(){

Il\_node \*head=NULL; //all’inizio il puntatore head punta a null

Il\_node \*el1=(Il\_node)malloc(sizeof(Il\_node));

el1->info=5;

el1->next=NULL;

Il\_node \*el2=(Il\_node)malloc(sizeof(Il\_node));

el2->info=1;

el2->next=NULL;

Il\_node \*el3=(Il\_node)malloc(sizeof(Il\_node));

el3->info=20;

el3->next=NULL;

head=el1;

el1->next=el2;

el2->next=el3;

el3->next=NULL;

return 0;

}

**numeri random**

Programma che genera numeri casuali

#include<stdio.h>

#include<stdlib.h>

#include<time.h> //per ottenere un seme casuale

int main(){

int numero=0;

srand(42); //inizializzo il seme

numero=rand(); //cosi genero il numero casuale da 0 fino a 42

int i=0;

srand(time(NULL)); //inizializzo il seme

i=rand()%100+1; //oppure i=rand();

printf(“Il numero casuale generato da 1 a 100 è: %d\n”,i);

return 0;

}

Stringhe

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

int lunghezza(char \*stringa){ //restituisce la lunghezza della stringa

int i=0;

while(stringa[i]!=’\0’)

i++;

return i;

}

int main(){

char stringa[6]={‘p’,’i’,’p’,’p’,’o’,’\0’}; //la dimensione comprende anche la fine

char \*p=”pippo”; //p è una costante stringa

printf(“%s\n”,p);

char \*arraydicostanti[]={“uno”,”due”,”tre”}; //non ci vuole il carattere della fine

char \*buff=(char \*)malloc(40\*sizeof(char));

fgets(buff,40,stdin); //se non conosciamo la lunghezza della stringa a priori

scanf(“%s”,buff);

scanf(“%[^\n]”,s); // cosi non si ferma quando incontra uno spazio

char s[]=”Ciao a tutti”;

char \*p=strtok(s,” “); //token divide la stringa in sottostringhe

while(p!=NULL){

printf(“%s\n”,p);

p=strtok(NULL, “ “);

}

return 0;

}

LEZIONE 3

**Programma 3.1**

/\*SIZEOF(): Scrivere un programma che legga da tastiera un numero intero x e stabilisca il numero di byte necessario a memorizzare x variabili di tipi intero.

input 3 output 12 \*/

#include<stdio.h>

int main(){

int x;

scanf(“%d”,&x);

printf(“%d\n”,sizeof(int)\*x);

return 0;

}

**Programma 3.2**

/\*TIMECONVERTOR: Scrivere un programma che legga da tastiera un valore di temperatura (anche decimale) espresso in gradi Celsius e ne stampi la conversione in gradi Fahrenheit usando due cifre decimali. Per convertire un valore da Celsius a Fahrenheit si ha che F=C\*1.8 + 32

input 10 output 50.00 \*/

#include<stdio.h>

int main(){

float c;

scanf("%f",&c);

float f;

f=c\*1.8 +32;

printf("%.2f\n",f);

return 0;

}

//oppure un altro esempio

//Questo programma stampa la tabella Fahrenheit-Celsius.

#include<stdio.h>

int main(){

int fahr;

for(fahr=0; fahr<=300; fahr=fahr+20)

printf("%3d %6.1f\n",fahr,(5.0/9.0)\*(fahr-32));

return 0;

}

//oppure con un altro esempio analogo al precedente

#include<stdio.h>

int main(){

float fahr, celsius; //con int i gradi celsius non sarebbero del tutto precisi

int lower, upper, step;

lower=0; //limite inferiore della tabella fahr

upper=300; //limite superiore della tabella fahr

step=20; //incremento

fahr=lower;

while(fahr<=upper){

celsius=5\*(fahr-32)/9; //non si moltiplica direttamente per 5/9 perche' il C effettuerebbe un troncamento per cui qualsiasi parte frazionaria viene scartata. Essendo 5 e 9 interi la frazione viene troncata a zero e quindi restituisce niente. Usando invece i reali possiamo usare (5.0/9.0)\*(fahr-32) che non viene stroncata perche' li considera come quoziente tra due numeri decimali.

printf("%3.0f\t %6.1f\n",fahr,celsius); //\t e' il carattere di tabulazione, i numeri indicano l'incolonnamento a destra

fahr=fahr+step;

}

return 0;

}

**Programma 3.3**

/\*AVERAGE: Scrivere un programma che legga da tastiera tre numeri (anche con la virgola) e stampi in uscita la media, riportandola con tre cifre decimali.

input 1 2 3 output 2.000

input 10.5 23.46 31.24 output 21.733 \*/

#include<stdio.h>

int main(){

float x, y, z;

scanf("%f",&x);

scanf("%f",&y);

scanf("%f",&z);

float media;

media=(x+y+z)/3;

printf("%.3f\n",media);

return 0;

}

**Programma 3.4**

/\*MAX: Scrivere un programma che legga da tastiera tre numeri interi e ne stampi il massimo.

input 15 7 12 output 15 \*/

#include<stdio.h>

int main(){

int x, y, z;

scanf("%d",&x);

scanf("%d",&y);

scanf("%d",&z);

int max=0;

if(x>max) max=x;

if(y>max) max=y;

if(z>max) max=z;

printf("%d\n",max);

return 0;

/\* //oppure

int a, b, c, max=0, max1=0;

scanf("%d", &a);

scanf("%d", &b);

scanf("%d", &c);

if(a<b) max1=b;

else max1=a;

if(max1<c) max=c;

else max=max1;

printf("%d\n", max);

return 0; \*/

}

**Programma 3.5**

/\*PARI: Scrivere un programma che legga da tastiera un numero e stabilisca se pari o meno, stampando in uscita rispettivamente 1 o 0.

input 15 output 0 input 4 output 1 \*/

#include <stdio.h>

int main(){

int x;

scanf("%d",&x);

if (x%2==0) x=1;

else x=0;

printf("%d\n",x);

return 0;

}

//VARIANTE:Scrivere un programma che stampi i numeri pari minori di 10.

#include<stdio.h>

int main(){

int i;

for(i=0; i<10; i++)

if(i%2==0)

printf(“%d\n”,i);

return 0;

}

**Programma 3.6**

/\*CAPITALIZE: Scrivere un programma che legga da tastiera un carattere alfabetico in minuscolo e stampi lo stesso carattere in maiuscolo.

input c output C \*/

#include<stdio.h>

int main(){

char x;

scanf("%c",&x);

printf("%c\n",x-32);

return 0;

}

**Programma 3.7**

/\*SECONDI: Scrivere un programma che legga da tastiera un numero che rappresenta un numero di secondi, e stampi le ore, i minuti e i secondi compresi.

Input: 5 Output: 0 h 0 min 5 s

Input: 138 Output: 0 h 2 min 18 s

Input: 3685 Output: 1 h 1 min 20 s \*/

#include<stdio.h>

int main(){

int secondi, ore=0, minuti=0;

scanf("%d", &secondi);

while(secondi>=60){

if(secondi>=3600){

ore++;

secondi=secondi-3600;

}else if(secondi<3600 && secondi>=60){

minuti++;

secondi=secondi-60;

}

}

printf("%d h %d min %d s\n", ore, minuti, secondi);

return 0;

}

**Programma 3.8**

/\*CALCOLATORE: Scrivere un programma che legga da tastiera due numeri a e b (anche con virgola) e un operatore tra + - / % e stampi il risultato ottenuto applicando l'operatore ai due numeri inseriti. Per l'operazione % gli operandi sono |a| e |b|.

Input: 5 7 + Output: 12.0

Input: 5 7 - Output: -2.0

Input: 6.2 2 / Output: 3.1 \*/

#include<stdio.h>

#include<math.h>

int main(){

float a, b;

char x; //l'operatore lo dichiaro come carattere

scanf("%f", &a);

scanf("%f", &b);

scanf("%s", &x); //qui devo usare l'identificatore per una stringa anche se e' un carattere

//in base al loro codice decimale della tabella ASCII faccio le rispettive operazioni

//oppure con uno switch

if(x==43) printf("%.1f\n", a+b);

if(x==45) printf("%.1f\n", a-b);

if(x==42) printf("%.1f\n", a\*b);

if(x==47) printf("%.1f\n", a/b);

if(x==37) printf("%d.0\n", (int)a%(int)b);

return 0;

}

**Programma 3.9**

/\*STAMPA RETTANGOLO: Scrivere un programma che stampi un rettangolo di 10 righe e 3 colonne la cui cornice sia costituita da caratteri asterisco e la parte interna da 'X'.

Output: \*\*\*

\*X\*

\*X\*

\*X\*

\*X\*

\*X\*

\*X\*

\*X\*

\*X\*

\*\*\* \*/

#include<stdio.h>

int main(){

int i=0;

if(i==0)

printf("\*\*\*\n");

for(i=1; i<9; i++)

printf("\*X\*\n");

if(i==9)

printf("\*\*\*\n");

return 0;

}

/\*oppure

int main(){

printf("\*\*\*\n");

int i=0;

for(i=0; i<8; i++)

printf("\*X\*\n");

printf("\*\*\*\n");

return 0;

} \*/

LEZIONE 4

**Programma 4.1**

/\*SEQUENZA INVERTITA: Scrivere un programma che legga da tastiera una sequenza di 10 interi, e stampi la stessa sequenza con ordine invertito, dividendo per 2 gli elementi pari della sequenza.

input 5 4 -2 45 7 9 18 -18 5 5

output 5 5 -9 9 9 7 45 -1 2 5 \*/

#include<stdio.h>

int main(){

int a[10],n,i;

for(i=0; i<10; i++){

scanf("%d",&n);

a[i]=n;

}

for(i=9; i>=0; i--){

if(a[i]%2==0)

printf("%d\n",(a[i]/2));

else

printf("%d\n",a[i]);

}

return 0;

}

**Programma 4.2**

/\*FATTORIALE: Scrivere un programma che legga da tastiera un numero intero x e stampi il valore di x! dove con x! si intende x fattoriale, ovvero x\*(x-1)\*(x-2)...\*1.

input 5 output 120 \*/

#include<stdio.h>

int fact(int x){

if(x==1) return 1; //caso base della ricorsione

return x\*(fact(x-1));

}

int main(){

int x;

scanf("%d",&x);

printf("%d\n",fact(x));

return 0;

}

//oppure

/\*Questo programma stampa su schermo il fattoriale del minimo dei due valori letti da tastiera.\*/

#include<stdio.h>

int fact(int n){

if(n==1) return 1;

else return n\*fact(n-1);

}

int min(int a, int b){

int m;

if(a<b) m=a;

else m=b;

return m;

}

int main(){

int n, m, x,l;

scanf("%d %d",&n,&m);

l=min(n,m);

x=fact(l);

printf("%d\n",x);

return 0;

}

**Programma 4.3**

/\*VERIFICA DI CONDIZIONE: Scrivere un programma che legga da tastiera 10 interi, li scriva in un array A e stampi l'indice del primo elemento che verifica la seguente condizione: A[i]=A[i+1]-A[i-1] oppure stampi -1 nel caso in cui nessun elemento verifichi la suddetta condizione.

input 9 5 -2 3 5 50 4 -7 0 -7 output 2

input 9 5 -1 3 6 8 1 0 -7 5 output -1\*/

#include <stdio.h>

int main(){

int a[10],i,trovato=0;

for(i=0; i<10; i++) scanf("%d",&a[i]); /\*oppure .. scanf("%d",&n); b[j]=n;\*/

i=1;

while(i<9 && !trovato){ /\*oppure ..trovato==0)\*/

if(a[i]==a[i+1]-a[i-1]){

printf("%d\n",i);

trovato=1;

}

i++; /\*else lo posso anche omettere\*/

}

if(!trovato) printf("%d\n",-1);

return 0;

}

**Programma 4.4**

/\*SOTTO SUCCESSIONE: Scrivere un programma che legga da tastiera una sequenza di 10 interi scrivendola in un array, quindi stampi (uno per linea, e nello stesso ordine in cui sono stati immessi) i valori della sequenza che rispettano una di queste due proprieta': siano positivi e pari, oppure siano negativi e succeduti da un valore positivo.

input 13 -3 -5 5 0 2 -3 -6 3 -8

output -5 0 2 -6 \*/

#include<stdio.h>

int main(){

int a[10]; int i;

for(i=0; i<10; i++) scanf("%d",&a[i]);

for(i=0; i<10; i++)

if((a[i]>=0 && a[i]%2==0)||(i<9 && a[i]<0 && a[i+1]>=0))

printf("%d\n",a[i]);

return 0;

}

/\*oppure

int main(){

int a[10], i=0;

for(i=0; i<10; i++) scanf("%d", &a[i]);

if(a[9]>=0){

for(i=0; i<10; i++){

if((a[i]>=0 && a[i]%2==0) || (a[i]<0 && a[i+1]>=0))

printf("%d\n", a[i]);

}

}else if(a[9]<0){

for(i=0; i<9; i++){

if((a[i]>=0 && a[i]%2==0) || (a[i]<0 && a[i+1]>=0))

printf("%d\n", a[i]);

}

}

return 0;

} \*/

**Programma 4.5**

/\*SUCCESSIONE DI FIBONACCI: Scrivere un programma che legga da tastiera un numero intero n con n>=0 e stampi i valori della successione di Fibonacci, uno per riga, fino al primo valore della successione strettamente maggiore di n (questo valore non va stampato). Si ricorda che la successione di Fibonacci e' definita ricorsivamente come:

f(0)=0 f(1)=1 f(i)=f(i-1)+f(i-2)

input 13 output 0 1 1 2 3 5 8 13\*/

#include<stdio.h>

int main(){

int n,fib1,fib2,fibcorr;

scanf("%d",&n);

fib1=0; fib2=1;

do {

printf("%d\n",fib1);

fibcorr=fib1+fib2;

fib1=fib2;

fib2=fibcorr;

} while(fib1<=n);

return 0;

}

/\*oppure meno elegante

void fib(int x){

int prev=1, cor=0, tcor=0;

if(x==0 || x==1)

printf("%d\n", x);

else if(x>1){

printf("0\n1\n");

while(prev<x){

tcor=cor+prev;

if(tcor>x) break;

printf("%d\n", tcor);

cor=prev;

prev=tcor;

}

}

}

int main(){

int n;

scanf("%d", &n);

fib(n);

return 0;

} \*/

**Programma 4.6**

/\*ASTERISCHI: Scrivere un programma che legga da tastiera un numero intero n e stampi n asterischi sulla prima linea, n-2 asterischi sulla seconda linea, n-4 sulla terza e cosi via, fino ad arrivare a stampare uno o due asterischi sull'ultima linea.

input 5 output \*\*\*\*\* \*\*\* \*

input 8 output \*\*\*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\* \*\* \*/

#include<stdio.h>

int main(){

int n,i,ast;

scanf("%d",&n);

for(i=n; i>0; i=i-2){

for(ast=1; ast<=i; ast++)

printf("\*");

printf("\n");

}

return 0;

}

**Programma 4.7**

/\*RETTANGOLI: Scrivere un programma che legga da tatiera 2 interi h e l e stampi un rettangolo fatto da asterischi, con altezza h e lunghezza l.

Input: 5 10 Output: \*\*\*\*\*\*\*\*\*\*

\* \*

\* \*

\* \*

\*\*\*\*\*\*\*\*\*\* \*/

#include<stdio.h>

int main(){

int altezza, base, i, j;

scanf("%d", &altezza);

scanf("%d", &base);

for(j=0; j<altezza; j++){

if(j==0 && altezza>1){

for(i=0; i<base; i++)

printf("\*");

printf("\n");

}if(j>0 && j<altezza-1){

for(i=0; i<base; i++){

if(i==0) printf("\*");

if(i==base-2) printf("\*");

else printf(" ");

}printf("\n");

}if(j==altezza-1){

for(i=0; i<base; i++)

printf("\*");

printf("\n");

}

}

return 0;

}

**Programma 4.8**

/\*TABELLINE: Scrivere un programma che, preso in ingresso un numero intero x, stampi (con un numero per riga) la tabellina di x (partendo da x\*1 e terminando con x\*10)

Input: 2

Output: 2 4 6 8 10 12 14 16 18 20 \*/

#include<stdio.h>

int main(){

int x, i;

scanf("%d", &x);

for(i=1; i<11; i++)

printf("%d\n", x\*i);

return 0;

}

**Programma 4.9**

/\*VETTORI: Scrivere un programma che legga da tastiera due vettori interi di dimensione 5 (10 numeri interi in totale) e stampi prima i due vettori, poi la loro somma.

Input: 5 12 34 67 89 13 25 38 62 10

Output: [5,12,34,67,89] [13,25,38,62,10] [18,37,72,129,99]

Input: -1 1 1 -1 1 1 -1 -1 1 1

Output: [-1,1,1,-1,1] [1,-1,-1,1,1] [0,0,0,0,2] \*/

#include<stdio.h>

int main(){

int a[5], b[5], i;

for(i=0; i<5; i++) scanf("%d", &a[i]);

for(i=0; i<5; i++) scanf("%d", &b[i]);

printf("[");

for(i=0; i<5; i++){

if(i==4) printf("%d", a[i]);

else printf("%d,", a[i]);

}

printf("]\n");

printf("[");

for(i=0; i<5; i++){

if(i==4) printf("%d", b[i]);

else printf("%d,", b[i]);

}

printf("]\n");

printf("[");

for(i=0; i<5; i++){

if(i==4) printf("%d", a[i]+b[i]);

else printf("%d,", a[i]+b[i]);

}

printf("]\n");

return 0;

}

LEZIONE5

**Programma 5.1**

/\*FUNZIONE MEDIA: Si realizzi un programma nel linguaggio C che, data una sequenza di 10 interi da standard input, facendo uso di una funzione di nome media, calcoli la media aritmetica (reale) di tutti i valori diversi da zero e di segno uguale all'ultimo valore della sequenza. La media deve essere stampata con esattamente due cifre decimali. Assunzione: l'ultimo elemento della sequenza e' sempre diverso da zero.

input: -2 5 3 -1 6 -4 -9 2 12 1 output: 4.83 \*/

#include<stdio.h>

float media(int somma, int conta){

return ((float)somma)/conta;

}

int main(){

int a[10], i, contap=0, contan=0;

float sommap=0, somman=0;

for(i=0; i<10; i++) scanf("%d",&a[i]);

for(i=0; i<10; i++){

if(a[i]>0 && a[i]!=0){

sommap+=a[i];

contap++;

}else if(a[i]<0 && a[i]!=0){

somman+=a[i];

contan++;

}

}

if(a[9]>=0) printf("%.2f\n",media(sommap,contap));

else if(a[9]<0) printf("%.2f\n",media(somman,contan));

return 0;

}

/\* oppure

float media(int a[],int dim){

int somma=0; //oppure int somma; somma=0;

int quantita=0;

float med=1;

int i;

if(a[dim-1]>0)

for(i=0; i<dim; i=i+1)

if(a[i]>0){

somma=somma+a[i];

quantita=quantita+1;

}

if(a[dim-1]<0)

for(i=0; i<dim; i=i+1)

if(a[i]<0){

somma=somma+a[i];

quantita=quantita+1;

}

med=(float)somma/quantita;

return med;

}

int main(){

int a[10]; float med; int i;

for(i=0; i<10; i++)

scanf("%d",&a[i]);

med=media(a,10);

printf("%.2f\n",med);

return 0;

} \*/

/\*oppure

float media(int a[], int dim){

int i, sommap=0, somman=0, contap=0, contan=0;

for(i=0; i<dim; i++){

if(a[dim-1]>0 && a[i]>0){

sommap=sommap+a[i];

contap++;

}else if(a[dim-1]<0 && a[i]<0){

somman=somman+a[i];

contan++;

}

}

if(sommap!=0)

return (float)sommap/contap;

else if(somman!=0)

return (float)somman/contan;

}

int main(){

int i,a[10];

for(i=0; i<10; i++) scanf("%d", &a[i]);

printf("%.2f\n",media(a, 10));

return 0;

} \*/

**Programma 5.2**

/\*FUNZIONE MULTIPLI: Si realizzi un programma nel linguaggio C che, dati due interi N e M da standard input, facendo uso di una funzione di nome multipli, calcoli l'insieme degli interi appartenenti all'intervallo [1,N] che sono multipli di M. La chiamata a funzione dovra' stampare tutti gli elementi dell'insieme ordinati in ordine crescente.

input: 10 3 output: 3 6 9 \*/

#include<stdio.h>

int multipli(int n, int m){

int a[n],i;

for(i=1; i<n+1; i++){

a[i]=i;

if(a[i]%m==0)

printf("%d\n",i);

}

return;

}

int main(){

int x, y;

scanf("%d",&x);

scanf("%d",&y);

int mul=multipli(x,y);

return 0;

}

/\*oppure

void multipli(int n, int m){

int i;

for(i=1; i<n+1; i++)

if(i%m==0)

printf("%d\n",i);

}

int main(){

int m,n;

scanf("%d",&n);

scanf("%d",&m);

multipli(n,m);

return 0;

} \*/

**Programma 5.3**

/\*SOMMA DI POTENZE: Si realizzi un programma nel linguaggio C che legga da input un intero n ed un floating point x>0 e, utilizzando una funzione sumpow calcoli la somma delle potenze di x, da 0 ad n (ovvero, il valore s=x^0 +x^1 +x^2 +.. +x^n. Il valore restituito dalla funzione deve essere stampato sullo standard output. Il risultato deve essere stampato con esattamente due cifre decimali.

input: 5 2 output: 63.00 \*/

#include<stdio.h>

float sumpow(float x,int n){

if(n==0) return 1;

if(n==1) return x;

if(n%2==0) return sumpow(x,n/2)\*sumpow(x,n/2);

else if(n%2!=0) return x\*sumpow(x,n/2)\*sumpow(x,n/2);

}

int main(){

int a,i;

scanf("%d",&a);

float b,risultato=0;

scanf("%f",&b);

for(i=a; i>=0; i--)

risultato+=sumpow(b,i);

printf("%.2f\n",risultato);

return 0;

}

/\*oppure

void sum\_pow(int n, float x){

int i;

float somma=1, potenza=1;

for(i=1; i<=n; i++){

potenza=potenza\*x;

somma=somma+potenza;

}

printf("%.2f\n",somma);

}

int main(){

int n; float x;

scanf("%d",&n);

scanf("%f",&x);

sum\_pow(n,x);

return 0;

} \*/

**Programma 5.4**

/\*MCD & MCM: Si realizzi un programma in C che acquisisca da tastiera 2 numeri interi N ed M e utilizzi 2 funzioni per calcolare il minimo comune multiplo e il massimo comune divisore tra N ed M. Il programma stampa sullo standard output prima il mcd(n,m) e poi il mcm(n,m).

input: 2 5 output: 1 10 \*/

#include<stdio.h>

int mcd(int x, int y){

if(x==y) return x;

if(x<y) return mcd(x,y-x);

else return mcd(x-y,y);

}

int mcm(int x, int y){

int prodotto=x\*y;

return prodotto/mcd(x,y);

}

int main(){

int n,m;

scanf("%d",&n);

scanf("%d",&m);

printf("%d\n",mcd(n,m));

printf("%d\n",mcm(n,m));

return 0;

}

/\* oppure

int mcd(int n, int m){

while(n!=m){

if(n>m) n=n-m;

if(n<m) m=m-n;

}

return n;

}

void mcm(int n, int m){

printf("%d\n",(n\*m)/mcd(n,m));

}

int main(){

int n,m;

scanf("%d",&n);

scanf("%d",&m);

mcd(n,m);

printf("%d\n",mcd(n,m));

mcm(n,m);

return 0;

} \*/

**Programma 5.5**

/\*METODO BABILONESE: La radice quadrata di un numero z intero non negativo puo' essere calcolata in modo approssimato tramite il metodo detto Babilonese che utilizza la seguente relazione ricorsiva: sqrt(z)=1/2 \*(sqrt(z-1) + z/(sqrt(z-1))) se z>0 oppure 1 se z=0. Scrivere un programma che acquisisca da tastiera un intero non negativo e restituisca l'approssimazione della sua radice quadrata calcolata tramite il suddetto metodo implementato tramite una funzione approxsqrt. Il risultato deve essere stampato con esattamente 4 cifre decimali.

input: 86 output: 9.2738 \*/

#include<stdio.h>

float approxsqrt(int z){

if(z==0) return 1;

else if(z>0) return (approxsqrt(z-1)+(z/approxsqrt(z-1)))/2;

}

int main(){

int x;

scanf("%d",&x);

printf("%.4f\n",approxsqrt(x));

return 0;

}

**Programma 5.6**

/\*CALCOLO PI GRECO: Il numero PiGreco puo' essere calcolato in modo approssimato tramite la serie Gregory Leibniz che utilizza la seguente relazione: PiGreco=4/1 -4/3 +4/5 -4/7 +4/9 -... Scrivere un programma che acquisisca da tastiera un intero non negativo n e restituisca l'approssimazione di PiGreco usando la serie di Gregory Leibniz con n termini, implementata tramite una funzione approx\_pi. Il risultato deve essere stampato con esattamente 6 cifre decimali.

Input:1 Output: 4.000000 Input:10 Output: 3.041840 Input:100 Output: 3.131593 \*/

#include<stdio.h>

float GregoryL(int n){

int i, j;

float ris=0;

for(i=0; i<n; i++){

j=i+1;

if(i%2==0) ris=ris+(float)4/(i+j);

else if(i%2!=0) ris=ris-(float)4/(i+j);

}

return ris;

}

int main(){

int n;

scanf("%d", &n);

if(n>0) printf("%.6f\n", GregoryL(n));

return 0;

}

**Programma 5.7**

/\*NUMERI MAGGIORI: Scrivere un programma che acquisisca da tastiera un intero x e un array di 5 altri interi e restituisca il numero di elementi maggiori di x presenti nell'array. Il programma deve usare una funzione count\_larger che prende come parametri x, l'array e la lunghezza dell'array e restituisce il numero di elementi maggiori di x.

Input: 0 12 34 -2 56 2 Output: 4 \*/

#include<stdio.h>

int countlarger(int x, int a[], int dima){

int i, conta=0;

for(i=0; i<dima; i++){

if(a[i]>=x)

conta++;

}

return conta;

}

int main(){

int x, a[5], i;

scanf("%d", &x);

for(i=0; i<5; i++) scanf("%d", &a[i]);

printf("%d\n", countlarger(x, a, 5));

return 0;

}

LEZIONE 6

**Programma 6.1**

/\*FUNZIONE MODULO DIFFERENZA: Si realizzi un programma C che contenga una funzione diff\_abs che prenda come parametri 2 float (a e b) e la cui esecuzione faccia si che la variabile corrispondente al primo argomento contenga il valore di a-b, mentre la seconda variabile corrispondente al secondo argomento contenga b-a. Si stampi a video il contenuto delle 2 variabili all'uscita dall'esecuzione della funzione, con precisione alla seconda cifra decimale.

input: 3.14 -2.71 output: 5.85 -5.85

#include<stdio.h>

float diffabs1(float x, float y){

return x-y;

}

float diffabs2(float x, float y){

return y-x;

}

int main(){

float a, b;

scanf("%f",&a);

scanf("%f",&b);

printf("%.2f\n",diffabs1(a,b));

printf("%.2f\n",diffabs2(a,b));

return 0;

}

/\*oppure

void diff\_abs(float \*a,float \*b){

float temp1=0, temp2=0;

temp1=\*a-\*b;

temp2=\*b-\*a;

\*a=temp1;

\*b=temp2;

}

int main(){

float num1, num2;

scanf("%f", &num1);

scanf("%f", &num2);

diff\_abs(&num1, &num2);

printf("%.2f\n%.2f\n", num1, num2);

return 0;

}

//oppure

void diff\_abs(float \*a,float \*b){

float temp1=0;

temp1=\*a-\*b;

\*a=temp1;

\*b=-\*a;

}

//oppure

void diffabs(float \*a, float \*b){

float v=\*a;

\*a=\*a-\*b;

\*b=\*b-v;

}

int main(){

float num1, num2;

scanf("%f", &num1);

scanf("%f", &num2);

diff\_abs(&num1, &num2);

printf("%.2f\n%.2f\n", num1, num2);

return 0;

} \*/

**Programma 6.2**

/\*FUNZIONE SWAP ORDINATO: Si realizzi un programma che legga 3 interi da standard input ed applichi la funzione a 3 argomenti ordered\_swap(,,) alle 3 variabili in cui sono stati memorizzati i 3 valori. I valori letti devono essere passati alla funzione nell'ordine in cui vengono acquisiti da tastiera. Si implementi la funzione ordered\_swap, che cambia il contenuto delle 3 variabili di input in modo che la variabile corrispondente al primo argomento contenga l'intero di valore maggiore. Si stampi a video il contenuto delle 3 variabili all'uscita dall'esecuzione della funzione.

input: 14 -1 7 output: -1 7 14 \*/

#include<stdio.h>

int ordered1(int x, int y, int z){

int min=x;

if(y<x) min=y;

else if(z<min) min=z;

return min;

}

int ordered2(int x, int y, int z){

int max=z;

if(y>z) max=y;

else if(x>max) max=x;

return max;

}

int ordered(int x, int y, int z){

int minimum=ordered1(x,y,z);

int maximum=ordered2(x,y,z);

if(x!=minimum && x!=maximum) return x;

else if(y!=minimum && y!=maximum) return y;

else if(z!=minimum && z!=maximum) return z;

}

int main(){

int a, b, c;

scanf("%d",&a);

scanf("%d",&b);

scanf("%d",&c);

printf("%d\n",ordered1(a,b,c));

printf("%d\n",ordered(a,b,c));

printf("%d\n",ordered2(a,b,c));

return 0;

}

**Programma 6.3**

/\*ALLOCAZIONE DINAMICA ARRAY: Si realizzi un programma che legga un intero N da tastiera e lo passi ad una funzione che alloca un array di dimensione N e lo riempie con i valori dei primi N multipli di 5. Nel main si stampi a video il contenuto dell'array restituito dalla funzione. Assunzione: N e' un intero strettamente maggiore di 0.

input: 5 output: 5 10 15 20 25 \*/

#include<stdio.h>

#include<stdlib.h>

void allocazione(int n,int \*a){

int i;

a=malloc(n\*sizeof(int));

for(i=0; i<n; i++){

\*a=5\*(i+1);

printf("%d\n",\*a);

}

}

int main(){

int x, \*v, i;

scanf("%d",&x);

allocazione(x,v);

return 0;

}

/\* oppure

void funzione(int n){

int \*a=(int \*)malloc(n\*sizeof(int)), i;

for(i=0; i<n; i++) a[i]=(i+1)\*5;

for(i=0; i<n; i++) printf("%d\n", a[i]);

}

int main(){

int n;

scanf("%d", &n);

funzione(n);

return 0;

}

**Programma 6.4**

/\*RIDIMENSIONAMENTO ARRAY: Si realizzi un programma che legga 6 interi da tastiera e assegni i primi 5 valori letti ad altrettante posizioni di un array; si assuma di memorizzare il sesto valore in una variabile N. Si scriva una funzione che dato l'array inizializzato con i 5 valori letti da tastiera e l'intero N, restituisca l'array in input ridimensionato per contenere 5+N elementi, tale che ciascuno dei nuovi N elementi corrisponda alla somma dei numeri che lo precedono nell'array ad es: arr[i]=sommatoria per j=0 a i-1 di arr[j]. Nel main, si stampi a video il contenuto dell'array restituito dalla funzione.

input: -1 2 3 0 4 3

output: -1 2 3 0 4 8 16 32 \*/

#include<stdio.h>

#include<stdlib.h>

void posizioni(int \*v, int f, int somma){

v=malloc((f)\*sizeof(int));

int i=0;

v[i]=somma;

for(i=1; i<f; i++){

\*v=(\*v)\*2;

printf("%d\n",\*v);

}

}

int main(){

int a, b, c, d, e, f,i=0;

scanf("%d",&a);

scanf("%d",&b);

scanf("%d",&c);

scanf("%d",&d);

scanf("%d",&e);

scanf("%d",&f);

int \*arr=malloc((5)\*sizeof(int));

arr[i]=a;

arr[i+1]=b;

arr[i+2]=c;

arr[i+3]=d;

arr[i+4]=e;

for(i=0; i<5; i++)

printf("%d\n",arr[i]);

int somma=a+b+c+d+e;

printf("%d\n",somma);

posizioni(arr,f,somma);

return 0;

}

/\*oppure

int main(){

int n, \*a=(int \*)malloc(5\*sizeof(int)), i, j=0;

for(i=0; i<5; i++) scanf("%d", &a[i]);

scanf("%d", &n);

a=realloc(a, (5+n)\*sizeof(int));

for(i=5; i<(5+n); i++){

for(j=0; j<i; j++)

a[i]=a[i]+a[j];

}

for(i=0; i<5+n; i++) printf("%d\n", a[i]);

return 0;

} \*/

**Programma 6.5**

/\*MATRICI: Si legga dall'input una matrice di interi di dimensione 4\*3 (4 righe, 3 colonne). I valori dati in input sono ordinati per riga (i primi 3 interi sono i valori della prima riga della matrice, e cosi via). Si stampi l'indice dell'ultima colonna che contiene solo multipli di 3, o si stampi -1 nel caso che nessuna colonna rispetti questa proprieta'.

input: 5 3 12 5 6 6 5 21 3 5 6 24

output: 2 \*/

#include<stdio.h>

int main(){

int a[4][3],i,j,colonna=-1, trovato=0,conta=0;

for(i=0; i<4; i++){

for(j=0; j<3; j++){

scanf("%d",&a[i][j]);

}

}

i=0, j=2;

while(j>=0 && trovato==0){

if(a[i][j]%3==0){

conta=j;

trovato=1;

}

else

j--;

}

if(trovato!=0){

i=1; j=conta;

while(i<4)

if(a[i][conta]%3!=0){

colonna=-1;

}

else if(a[i][conta]%3==0){

i++;

colonna=conta;

}

}

printf("%d\n",colonna);

return 0;

}

/\*oppure

int main(){

int a[4][3], i, j, conta=0, trovato=0, ris=0;

for(i=0; i<4; i++)

for(j=0; j<3; j++)

scanf("%d", &a[i][j]);

// for(i=0; i<4; i++){

// for(j=0; j<3; j++)

// printf("%d ", a[i][j]);

// printf("\n");

// }

i=0, j=2;

while(j>=0){

trovato=0;

for(i=0; i<4; i++){

if(a[i][j]%3==0)

trovato++;

}if(trovato==4){

ris=j;

break;

}else if(trovato!=4){

j--;

}

}

if(trovato!=4)

printf("-1\n");

else if(trovato==4)

printf("%d\n", ris);

return 0;

}

**Programma 6.6**

/\*OUTER PRODUCT: Si legga dall'input una sequenza di numeri (0 o 1) terminata dal numero -1. Si inseriscano tali numeri in un array di interi nell'ordine in cui vengono letti. L'ultimo numero letto (-1) non va inserito nell'array in quanto rappresenta unicamente un terminatore di sequenza. Si scriva una funzione prodotto\_scalare che, dato in input l'array v, contenente gli N interi precedentemente letti, restituisca in output la matrice A di N\*N interi, tale che l'elemento A[i][j] della matrice sia pari a v[i]\*v[j]. Tale matrice e' il risultato del prodotto esterno tra il vettore v e il proprio trasposto. Nel main del programma si stampi il contenuto della matrice A, assicurandosi che tutti gli elementi di ciascuna riga della matrice siano stampati su un unica riga e separati da un singolo spazio; righe diverse della matrice vanno separate con un "a capo".

input: output:

0 0000

1 0111

1 0111

1 0111

-1 \*/

#include<stdio.h>

#include<stdlib.h>

#define VALORE 100

int\*\* prodottoScalare(int \*v, int n){

int i, j;

int\*\* mat=(int\*\*)malloc(n\*sizeof(int\*));

for(i=0; i<n; i++){

mat[i]=(int\*)malloc(n\*sizeof(int));

for(j=0; j<n; j++){

mat[i][j]=v[i]\*v[j];

}

}

return mat;

}

int main(){

int \*v=(int\*)malloc(VALORE\*sizeof(int)), len=0, n, \*\*mat, i, j, \*extv;

if(v!=NULL){

do{

scanf("%d",&n);

if(n!=-1){

v[len]=n;

len++;

extv=(int\*)realloc(v,(len+1)\*sizeof(int)); /\*realloc si usa quando mi accorgo che l'array e' troppo piccolo e voglio ingrandirlo dinamicamente senza perdere le informazioni precedentemente memorizzate. v e' un puntatore ad un'area di memoria precedentemente allocata segue poi dopo la virgola la nuova dimensione dell'area di memoria\*/

v=extv;

}

}while(n!=-1);

mat=prodottoScalare(v, len);

for(i=0; i<len; i++){

for(j=0; j<len; j++){

printf("%d ",mat[i][j]);

}

printf("\n");

}

free(v);

free(mat);

}

return 0;

}

**Programma 6.7**

/\*MOLTIPLICAZIONE MATRICI: Si legga dall'input 3 interi n, m, t, poi una matrice di interi di dimensione n\*m (n righe, m colonne) e una seconda matrice di dimensione m\*t. Per ogni matrice i valori dati in input sono ordinati per riga (per la prima matrice i primi m interi sono i valori della prima riga della matrice, e cosi via). Si usi una funzione per leggere le matrici. Si scrivi una seconda funzione che moltiplichi le due matrici e restituisca la matrice risultato (n\*t). Si stampi il risultato nella funzione main.

Input: 4//n 2//m 3//t 5 3 5 6 5 21 5 6 1 0 2 2 1 1

Matrice A: 5 3 Matrice b: 1 0 2 Output: 11 3 13

5 6 2 1 1 17 6 16

5 21 47 21 31

5 6 17 6 16 \*/

#include<stdio.h>

int main(){

int n, m, t, i=0, j=0, k=0, l=0, p=0, q=0, somma=0;

scanf("%d", &n);

scanf("%d", &m);

scanf("%d", &t);

int a[n][m], b[m][t], ris[n][t]; //vengono accettati anche senza allocazione dinamica

for(i=0; i<n; i++)

for(j=0; j<m; j++)

scanf("%d", &a[i][j]);

for(i=0; i<m; i++)

for(j=0; j<t; j++)

scanf("%d", &b[i][j]);

for(i=0; i<n; i++)

for(j=0; j<t; j++)

ris[i][j]=0;

i=0; j=0;

while(p<t){

j=0;

somma=0;

while(j<m){

somma=somma+a[i][j]\*b[j][p];

j++;

}

ris[i][p]=somma;

if(i==n){

i=0;

p++;

}else

i++;

}

for(i=0; i<n; i++){

for(j=0; j<t; j++)

printf("%d ", ris[i][j]);

printf("\n");

}

return 0;

}

LEZIONE 7

**Programma 7.1**

/\*GESTORE DEL GATTILE: Si scriva un programma di gestione di un gattile che legga dall'input un intero n, quindi per n volte, legga:

\_un codice identificativo del gatto (un intero)

\_l'eta' del gatto in anni (un intero)

\_il peso del gatto in kg (un float)

\_il tipo di cibo assunto dal gatto, espresso come 0 per le crocchette, 1 per le scatolette, 2 per il tonno in scatola.

Nello stesso ordine in cui sono stati immessi i dati dei gatti si stampino, uno per linea, i codici identificativi e il tipo di cibo di tutti i gatti che hanno problemi di dieta in giovane eta', ovvero abbiano meno di 4 anni e pesino piu' della media dei gatti. A fianco del codice identificativo, si stampi, separato da uno spazio, il tipo di cibo assunto dal gatto, ovvero crocchette, scatolette oppure tonno. N.B. Si consiglia di definirsi opportunamente una struct di nome gatto, e rappresentare il tipo di cibo come una enum.

input: 5 2 13 5.0 1 6 3 6.0 0 9 2 3.0 2 1 13 4.0 2 12 1 4.6 2

output: 6 crocchette 12 tonno \*/

#include<stdio.h>

#include<stdlib.h>

typedef enum{

crocchette,/\*e' implicito che crocchette=0 e via via gli altri\*/

scatolette,

tonno

}cibo;

typedef struct{

int codice;

int eta;

float peso;

cibo svedese;

}gatto;

int main(){

int n,i=0;

float media, somma=0;

scanf("%d",&n);

gatto \*persiano=(gatto \*)malloc(n\*sizeof(gatto)); //va allocata memoria per tutti gli n gatti

//cibo \*svedese=malloc(n\*sizeof(cibo)); quest'allocazione di memoria non serve

for(i=0; i<n; i++){

scanf("%d",&(persiano[i].codice)); //una struct e' come un array!! quindi si usa l'indice se no rimani sempre li'.

scanf("%d",&(persiano[i].eta)); //oppure ..(persiano+i->codice));

scanf("%f",&(persiano[i].peso));

somma=somma+(persiano[i].peso);

scanf("%u",&(persiano[i].svedese));

}

media=(float)somma/n;

i=0;

do{

if((persiano[i].eta)<4&&(persiano[i].peso)>media){

printf("%d ",persiano[i].codice);

if((persiano[i].svedese)==0){

printf("crocchette\n"); //la parola gli va scritta! se lascio ...printf("%d\n",persiano[i].svedese)

} //stampa il numero ma non lo trasforma

else{

if((persiano[i].svedese)==1){

printf("scatolette\n");

}

else{

if((persiano[i].svedese)==2)

printf("tonno\n");

}

}

}

i++;

}while(i<n);

return 0;

}

/\*oppure

#include<stdio.h>

#include<stdlib.h>

typedef enum \_cibo{

crocchette, scatolette, tonno

}cibo;

typedef struct \_gatto{

int codice;

int anni;

float peso;

cibo gustoso;

}gatto;

int main(){

int n,i;

float somma=0, media=0;

scanf("%d", &n);

gatto \*miao=(gatto \*)malloc(n\*sizeof(gatto));

for(i=0; i<n; i++){

scanf("%d", &(miao[i].codice));

scanf("%d", &(miao[i].anni));

scanf("%f", &(miao[i].peso));

somma=somma+miao[i].peso;

scanf("%u", &(miao[i].gustoso));

}

media=(float)somma/n;

for(i=0; i<n; i++){

//printf("%d %d %.2f %d\n", miao[i].codice, miao[i].anni, miao[i].peso, miao[i].gustoso);

if(miao[i].anni<4 && miao[i].peso>media){

printf("%d ", miao[i].codice);

if(miao[i].gustoso==0) printf("crocchette\n");

if(miao[i].gustoso==1) printf("scatolette\n");

else if(miao[i].gustoso==2) printf("tonno\n");

}

}

return 0;

} \*/

**Programma 7.2**

/\*STUDENTI: Si scriva un programma di gestione studenti che legga dall'input un intero n, quindi per n volte, si legga uno studente, definito come:

\_una matricola (un intero)

\_un'eta' (un intero)

\_il sesso (un intero, 0 per maschio, 1 per femmina)

Dopodiche' si legge un intero m. A questo punto:

\_Stampare la media e la varianza dell'eta' degli studenti maschi (uno per riga, 2 cifre decimali). Stampare media e varianza uguale a 0 se non ci sono studenti maschi.

\_Stampare la media e la varianza dell'eta' delle studentesse (uno per riga, 2 cifre decimali). Stampare media e varianza uguale a 0 se non ci sono studentesse.

\_Stampare le seguenti informazioni, una per riga:

\_eta' dello studente con matricola m

\_sesso dello studente con matricola m. Anziche' stampare l'intero, stampare il carattere 'M' se lo studente e' maschio, 'F' se e' femmina.Se lo studente m non e' presente, stampare -1 sia per l'eta' che per il sesso.

Ricordiamo che la varianza e' definita come: var=((x1 -mu)^2 + (x2 -mu)^2 +...+(xn -mu)^2)/(N-1) dove N e' il numero degli studenti maschi, x1,x2,...,xn sono le eta' degli studenti maschi, e mu e' la media delle eta' degli studenti maschi (usare la stessa formula con i dovuti cambiamenti per la varianza delle studentesse). N.B. si consiglia di definirsi opportunatamente una struct di nome studente ed un enumerazione per definire il sesso dello studente.

input: 3 112345 19 1 223456 20 0 998724 27 0 223456

output: 23.50 24.50 19.00 0.00 20 M \*/

#include<stdio.h>

#include<stdlib.h>

#define VALORE 100

int i=0;

typedef enum{

maschio,

femmina

}sesso;

typedef struct{

int matricola;

int eta;

sesso specie;

}studente;

int main(){

float mediam,mediaf,sommam=0,sommaf=0, varianzam=0, varianzaf=0;

int n, contam=0,contaf=0, \*extetam, \*extetaf, trovato=0;

int \*etam=malloc(100\*sizeof(int));

int \*etaf=malloc(100\*sizeof(int));

scanf("%d",&n);

studente \*universitario=malloc(n\*sizeof(studente));

for(i=0; i<n; i++){

scanf("%d",&(universitario[i].matricola));

scanf("%d",&(universitario[i].eta));

scanf("%u",&(universitario[i].specie));

if((universitario[i].specie)==0){

etam[contam]=universitario[i].eta;

contam++;

extetam=realloc(etam,(contam+1)\*sizeof(int));

etam=extetam;

sommam=sommam+universitario[i].eta;

}

else if((universitario[i].specie)==1){

etaf[contaf]=universitario[i].eta;

contaf++;

extetaf=realloc(etaf,(contaf+1)\*sizeof(int));

sommaf=sommaf+universitario[i].eta;

}

}

int m;

scanf("%d",&m);

if(contam==0){

printf("%.2f\n",0.00);

printf("%.2f\n",0.00);

}

else if(contam!=0){

mediam=(float)sommam/contam;

printf("%.2f\n",mediam);

if(contam==1){

printf("%.2f\n",0.00);

}

else if(contam!=1){

for(i=0; i<contam; i++){

varianzam+=((etam[i]-mediam)\*(etam[i]-mediam))/(contam-1);

}

printf("%.2f\n",varianzam);

}

}

if(contaf==0){

printf("%.2f\n",0.00);

printf("%.2f\n",0.00);

}

else if(contaf!=0){

mediaf=(float)sommaf/contaf;

printf("%.2f\n",mediaf);

if(contaf==1){

printf("%.2f\n",0.00);

}

else if(contaf!=1){

for(i=0; i<contaf; i++){

varianzaf+=((etaf[i]-mediaf)\*(etaf[i]-mediaf))/(contaf-1);

}

printf("%.2f\n",varianzaf);

}

}

i=0;

while(i<n && trovato==0){

if((m==universitario[i].matricola)&&(universitario[i].specie==0)){

printf("%d\n",universitario[i].eta);

printf("M\n");

trovato=1;

}

else if((m==universitario[i].matricola)&&(universitario[i].specie==1)){

printf("%d\n",universitario[i].eta);

printf("F\n");

trovato=1;

}

i++;

}

if(trovato==0){

printf("-1\n");

printf("-1\n");

}

return 0;

}

/\* oppure

#include<stdio.h>

#include<stdlib.h>

typedef enum \_sesso{

maschio, femmina

}sesso;

typedef struct \_studente{

int matricola;

int eta;

sesso genere;

}studente;

int main(){

int n, i, m, contam=0, contaf=0, j, trovato=0;

float sommam=0, sommaf=0, media=0, varianza;

scanf("%d", &n);

studente \*universitario=(studente \*)malloc(n\*sizeof(studente));

for(i=0; i<n; i++){

scanf("%d", &(universitario[i].matricola));

scanf("%d", &(universitario[i].eta));

scanf("%u", &(universitario[i].genere));

if(universitario[i].genere==0){

sommam=sommam+universitario[i].eta;

contam++;

}else if(universitario[i].genere==1){

sommaf=sommaf+universitario[i].eta;

contaf++;

}

}

scanf("%d", &m);

if(contam==0)

printf("0.00\n0.00\n");

else if(contam!=0){

media=(float)sommam/contam;

printf("%.2f\n", media);

j=0, i=0;

if(contam==1)

printf("0.00\n");

else if(contam>1){

while(i<contam){

if(universitario[j].genere==0){

varianza+=((universitario[j].eta-media)\*(universitario[j].eta-media))/(contam-1);

i++;

}

j++;

}

printf("%.2f\n", varianza);

}

}

if(contaf==0)

printf("0.00\n0.00\n");

else if(contaf!=0){

media=0, varianza=0;

media=(float)sommaf/contaf;

printf("%.2f\n", media);

j=0, i=0;

if(contaf==1)

printf("0.00\n");

else if(contaf>1){

while(i<contaf){

if(universitario[j].genere==1){

varianza+=((universitario[j].eta-media)\*(universitario[j].eta-media))/(contaf-1);

i++;

}

j++;

}

printf("%.2f\n", varianza);

}

}

i=0;

while(i<n && trovato==0){

if(universitario[i].matricola==m){

printf("%d\n", universitario[i].eta);

if(universitario[i].genere==0) printf("M\n");

else if(universitario[i].genere==1) printf("F\n");

trovato=1;

}i++;

}

if(trovato==0) printf("-1\n-1\n");

return 0;

} \*/

**Programma 7.3**

/\*GESTIONE PRL: Si scriva un programma per la gestione degli esercizi del corso di PRL. Si legga da tastiera un intero n, dopodiche' per n volte si legga:

\_il numero della lezione (un intero)

\_l'identificatore dell'esercizio (un intero)

\_il numero di studenti che hanno svolto l'esercizio (un intero)

\_le matricole degli studenti che hanno svolto l'esercizio (una per riga, ogni matricola e' un intero).

A questo punto si legga un intero m. Si stampino quindi il numero della lezione e l'identificatore di tutti igli esercizi svolti dallo studente con matricola m, uno per riga, nel formato [NumeroLezione].[Identificatore]. Ad es. per indicare che lo studente con matricola m ha risolto soltanto il secondo esercizio della terza lezione, si stampera' la stringa 3.2. Se lo studente non ha risolto nessun esercizio, non bisogna stampare niente.

input: 2 1 1 3 452341 998721 998124 1 2 1 998721 998721

output: 1.1 1.2 \*/

#include<stdio.h>

#include<stdlib.h>

int i=0;

typedef struct{

int lezione;

int esercizio;

int studenti; //numero studenti che hanno svolto l'esercizio

int \*matricola;

}prl;

int main(){

int n, contas=0,j,m;

scanf("%d",&n);

prl \*laboratorio=malloc(n\*sizeof(prl));

for(i=0; i<n; i++){

scanf("%d",&laboratorio[i].lezione);

scanf("%d",&laboratorio[i].esercizio);

scanf("%d",&laboratorio[i].studenti);

if(laboratorio[i].studenti!=0){

laboratorio[i].matricola=malloc((laboratorio[i].studenti)\*sizeof(int));

for(j=0; j<laboratorio[i].studenti; j++){

scanf("%d",&laboratorio[i].matricola[j]);

}

}

}

scanf("%d",&m);

for(i=0; i<n; i++){

for(j=0; j<laboratorio[i].studenti; j++){

if(m==laboratorio[i].matricola[j]){

printf("%d",laboratorio[i].lezione);

printf(".");

printf("%d\n",laboratorio[i].esercizio);

}

}

}

return 0;

}

/\*oppure

#include<stdio.h>

#include<stdlib.h>

typedef struct \_esercizio{

int lezione;

int ide;

int num;

int matricola[100];

}esercizio;

int main(){

int n, i, m, trovato=0, j;

scanf("%d", &n);

esercizio \*prl=(esercizio \*)malloc(n\*sizeof(esercizio));

for(i=0; i<n; i++){

scanf("%d", &(prl[i].lezione));

scanf("%d", &(prl[i].ide));

scanf("%d", &(prl[i].num));

for(j=0; j<prl[i].num; j++)

scanf("%d", &(prl[i].matricola[j]));

}

scanf("%d", &m);

i=0;

while(i<n && trovato==0){

for(j=0; j<prl[i].num; j++){

if(prl[i].matricola[j]==m)

printf("%d.%d\n", prl[i].lezione, prl[i].ide);

}

i++;

}

return 0;

} \*/

LEZIONE8

**Programma 8.1**

/\*PILA: Le liste concatenate permettono di creare facilmente altre strutture dati, come ad es: le pile. Partendo dalla definizione di Il\_node data a lezione realizzare le seguenti funzioni:

\_ Il\_node \*push(Il\_node \*head, int v) per aggiungere in cima alla lista un nuovo nodo contenente il valore v. Restituisce il puntatore alla nuova testa

\_ Il\_node \*pop(Il\_node \*head) per rimuovere il nodo in cima alla lista. Restituisce il puntatore alla nuova lista (fate attenzione al caso in cui la lista sia vuota: in questo caso il metodo non deve fare nulla).

\_ void print(Il\_node \*head) per stampare il contenuto attuale della lista.

Il programma dovra' acquisire da tastiera una lista di numeri interi. L'operazione da eseguire dipende dal valore acquisito:

\_ se il valore e' strettamente maggiore di 0, inserisce il valore in lista mediante la funzione push

\_ se il valore e' pari a 0 richiama la funzione pop

\_ se il valore e' strettamente minore di 0 termina stampando il contenuto corrente della lista.

input: 4 1 0 10 8 0 0 2 -1

output: 2 4 \*/

#include<stdio.h>

#include<stdlib.h>

typedef struct Il\_node\_struct{

int info;

struct Il\_node\_struct \*next;

}Il\_node;

Il\_node \*push(Il\_node \*head, int v){ //aggiungere in cima un nuovo nodo

Il\_node \*new\_node=(Il\_node \*)malloc(sizeof(Il\_node));

new\_node->info=v;

new\_node->next=head;

head=new\_node;

return head;

}

Il\_node \*pop(Il\_node \*head){ //per rimuovere il nodo piu' giovane dalla lista

/\*Il\_node \*succ; //in questo modo rimuove il nodo piu' vecchio della lista

if(head==NULL || head->next==NULL)

return;

succ=(head->next)->next;

free(head->next);

head->next=succ;

return head;\*/

if(head==NULL)

return head;

else{

Il\_node \*corr=head->next;

free(head);

return corr;

}

}

void print(Il\_node \*head){

while(head!=NULL){

printf("%d\n",head->info);

head=head->next;

}

}

int main(){

int n;

Il\_node \*head=NULL;

scanf("%d",&n);

while(n>0 || n==0){

if(n>0){

head=push(head, n);

scanf("%d",&n);

}

else if(n==0){

head=pop(head);

scanf("%d",&n);

}

}

print(head);

return 0;

}

/\*struct el{int info; struct el \*next;}; //oppure non come da consegna

typedef struct el \*Lista;

void push(Lista \*head, int n){

Lista new=(Lista)malloc(sizeof(struct el)), corr=\*head;

new->info=n;

new->next=(\*head);

(\*head)=new;

}

void pop(Lista \*head){

if((\*head)!=NULL){

Lista corr=\*head;

(\*head)=(\*head)->next;

free(corr);

}

}

void print(Lista head){

while(head!=NULL){

printf("%d\n",head->info);

head=head->next;

}

}

int main(){

Lista head=NULL;

int n;

scanf("%d", &n);

while(n>=0){

if(n!=0) push(&head, n);

else if(n==0) pop(&head);

scanf("%d", &n);

}

print(head);

// printf("\n");

return 0;

} \*/

**Programma 8.2**

/\*Si implementi una lista concatenata che contenga, come dati, interi positivi. Si implementino anche tre funzioni per:

\_aggiungere un elemento in fondo alla lista

\_aggiungere un elemento in cima alla lista

\_dato un valore positivo v, eliminare la prima occorrenza di v nella lista (non modificare la lista nel caso che non contenga v)

Quindi scrivere un programma che legga degli interi. Per ogni interi letto (e nello stesso ordine dell'input) si applichi una delle seguenti modifiche:

\_se il valore letto v e' strettamente negativo, rimuovere il primo elemento uguale a |v| dalla lista (non modificare la lista nel caso che questa non contenga |v|)

\_se il valore letto v e' positivo e pari, aggiungerlo in cima alla lista

\_se il valore letto v e' positivo e dispari, aggiungerlo in fondo alla lista

\_se il valore letto e' 0, terminare l'esecuzione del programma stampando, in ordine, gli elmenti della lista.

input: 4 5 2 -4 -5 -3 9 2 0

output: 2 2 9 \*/

#include<stdio.h>

#include<stdlib.h>

typedef struct Il\_node\_struct{

int info;

struct Il\_node\_struct \*next;

}Il\_node;

Il\_node \*addf(Il\_node \*head, int v){

Il\_node \*corr=head;

Il\_node \*new\_node=malloc(sizeof(Il\_node));

new\_node->info=v;

new\_node->next=NULL;

if(head==NULL){

new\_node->next=head;

head=new\_node;

}

else{

while(corr->next!=NULL){

corr=corr->next;

}

corr->next=new\_node;

}

return head;

}

Il\_node \*addc(Il\_node \*head, int v){

Il\_node \*new\_node=malloc(sizeof(Il\_node));

new\_node->info=v;

new\_node->next=head;

head=new\_node;

return head;

}

Il\_node \*elim(Il\_node \*head,int v){

Il\_node \*prec=NULL, \*corr=head;

int trovato=0;

while(corr!=NULL && trovato==0){

if(corr->info==v)

trovato=1;

else{

prec=corr; //avanzo con i puntatori fino a che trovo la prima occorrenza di v

corr=corr->next;

}

}

if(trovato==1){

if(prec==NULL){

head=corr->next;

free(corr);

}

else if(prec!=NULL){

prec->next=corr->next;

free(corr);

}

}

return head;

}

void print(Il\_node \*head){

while(head!=NULL){

printf("%d\n",head->info);

head=head->next;

}

}

int main(){

Il\_node \*head=NULL;

int n,t;

scanf("%d",&n);

while(n!=0){

if(n<0){

t=n\*(-1); //altrimenti non elimina quelli di segno positivo!

head=elim(head,t);

scanf("%d",&n);

}

if(n>0 && n%2==0){

head=addc(head,n);

scanf("%d",&n);

}

else if(n>0 && n%2!=0){

head=addf(head,n);

scanf("%d",&n);

}

}

print(head);

return 0;

}

/\*oppure

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

struct el{int info; struct el \*next;};

typedef struct el \*Lista;

void addtesta(Lista \*head, int n){

Lista new=(Lista)malloc(sizeof(struct el)), corr=\*head;

new->info=n;

new->next=(\*head);

(\*head)=new;

}

void addfondo(Lista \*head, int n){

Lista new=(Lista)malloc(sizeof(struct el)), corr=\*head;

new->info=n;

new->next=NULL;

if((\*head)==NULL){

new->next=(\*head);

(\*head)=new;

}else if((\*head)!=NULL){

while(corr->next!=NULL)

corr=corr->next;

corr->next=new;

}

}

void delete(Lista \*head, int n){

if((\*head)!=NULL){

Lista corr=\*head, prev=NULL;

int trovato;

while(corr!=NULL && trovato==0){

if(corr->info==abs(n)) trovato=1;

else if(corr->info!=n){

prev=corr;

corr=corr->next;

}

}

if(trovato==1){

if(prev==NULL){

(\*head)=(\*head)->next;

free(corr);

}else if(prev!=NULL){

prev->next=prev->next->next;

free(corr);

}

}

}

}

void print(Lista head){

while(head!=NULL){

printf("%d\n",head->info);

head=head->next;

}

}

int main(){

Lista head=NULL;

int n;

scanf("%d", &n);

while(n!=0){

if(n<0) delete(&head, n);

else if(n>0){

if(n%2==0) addtesta(&head, n);

else if(n%2!=0) addfondo(&head, n);

}

scanf("%d", &n);

}

print(head);

// printf("\n");

return 0;

} \*/

**Programma 8.3**

/\*Scrivere un programma che acquisisca da tastiera una successione di numeri interi positivi e li inserisca in maniera ordinata (crescente) all'interno di una lista. La lista puo' contenere duplicati. L'acquisizione da tastiera termina quando si incontra il primo valore negativo (che non va inserito in lista) e viene stampato il contenuto corrente della lista.

input: 4 5 7 1 2 6 9 10 -1

output: 1 2 4 5 6 7 9 10 \*/

#include<stdio.h>

#include<stdlib.h>

typedef struct Il\_node\_struct{

int info;

struct Il\_node\_struct \*next;

}Il\_node;

Il\_node \*ordina(Il\_node \*head, int v){

Il\_node \*new\_node=malloc(sizeof(Il\_node));

Il\_node \*prec=NULL, \*corr=head;

int trovato=0;

new\_node->info=v;

new\_node->next=NULL;

if(head==NULL){

new\_node->next=head;

head=new\_node;

}

else if(head!=NULL){

while(corr!=NULL && trovato==0){

if(corr->info>=v){

trovato=1;

}else if(corr->info<v){

prec=corr;

corr=corr->next;

}

}

if(trovato==1){

new\_node->next=corr;

if(prec!=NULL){

prec->next=new\_node;

}

else if(prec==NULL){

head=new\_node;

}

}

else if(trovato==0){

if(prec!=NULL){

prec->next=new\_node;

}

else if(prec==NULL){

head=new\_node;

}

}

}

return head;

}

void print(Il\_node \*head){

while(head!=NULL){

printf("%d\n",head->info);

head=head->next;

}

}

int main(){

Il\_node \*head=NULL;

int n;

scanf("%d",&n);

while(n>0){

head=ordina(head,n);

scanf("%d",&n);

}

print(head);

return 0;

}

/\*oppure

#include<stdio.h>

#include<stdlib.h>

struct el{int info; struct el \*next;};

typedef struct el \*Lista;

void addordine(Lista \*head, int n){

Lista new=(Lista)malloc(sizeof(struct el)), corr=\*head, prev=NULL;

new->info=n;

new->next=NULL;

if((\*head)==NULL){

new->next=(\*head);

(\*head)=new;

}else if((\*head)!=NULL){

int trovato=0;

while(corr!=NULL && trovato==0){

if(corr->info<n){

prev=corr;

corr=corr->next;

}else if(corr->info>=n) trovato=1;

}

if(trovato==1){

if(prev==NULL){

new->next=(\*head);

(\*head)=new;

}else if(prev!=NULL){

new->next=corr;

prev->next=new;

}

}else if(trovato==0) prev->next=new;

}

}

void print(Lista head){

while(head!=NULL){

printf("%d\n",head->info);

head=head->next;

}

}

int main(){

Lista head=NULL;

int n;

scanf("%d", &n);

while(n>=0){

addordine(&head, n);

scanf("%d", &n);

}

print(head);

// printf("\n");

return 0;

} \*/

**Programma 8.4**

/\*Scrivere un programma che acquisisca da tastiera 2 successioni di interi positivi, entrambe terminate da un numero negativo, e le inserisca in 2 liste distinte. Realizzare poi una funzione che, prese in input le 2 liste, restituisca una nuova lista contenente l'intersezione ordinata tra le due: non vi devono quindi essere elementi ripetuti e i valori devono essere memorizzati in ordine crescente. si stampi il contenuto della lista intersezione. N.B. si possono organizzare le 2 liste in input come meglio si ritiene necessario.

input: 4 5 6 7 -1 10 9 7 6 5 -1

output: 5 6 7 \*/

#include<stdio.h>

#include<stdlib.h>

typedef struct Il\_node\_struct{

int info;

struct Il\_node\_struct \*next;

}Il\_node;

Il\_node \*aggordinati(Il\_node \*head, int v){ //aggiunge gli elementi in maniera ordinata e senza duplicati per agevolare la ricerca

Il\_node \*prec=NULL, \*corr=head; //degli elementi intersezione

int trovato=0, dupli=0;

Il\_node \*new\_node=malloc(sizeof(Il\_node));

new\_node->info=v;

new\_node->next=NULL;

if(head==NULL){

new\_node->next=head;

head=new\_node;

}

else if(head!=NULL){

while(corr!=NULL && trovato==0 && dupli==0){

if(corr->info>v){

trovato=1;

}

else{

if(corr->info==v)

dupli=1;

else if(corr->info<v){

prec=corr;

corr=corr->next;

}

}

}

if(trovato==1){

if(prec!=NULL){

new\_node->next=corr;

prec->next=new\_node;

}

else if(prec==NULL){

new\_node->next=corr;

head=new\_node;

}

}

else if(trovato==0 && dupli==0){

new\_node->next=corr;

prec->next=new\_node;

}

}

return head;

}

Il\_node \*interseca(Il\_node \*head1, Il\_node \*head2){

Il\_node \*head3=NULL, \*corr1=head1, \*corr2=head2;

if((head1==NULL || head2==NULL))

return NULL;

else{

while(corr1!=NULL && corr2!=NULL){

if(corr1->info==corr2->info){

head3=aggordinati(head3,corr1->info);

corr1=corr1->next;

corr2=corr2->next;

}

else{

if(corr1->info<corr2->info)

corr1=corr1->next;

else if(corr1->info>corr2->info)

corr2=corr2->next;

}

}

return head3;

}

}

void print(Il\_node \*head){

while(head!=NULL){

printf("%d\n",head->info);

head=head->next;

}

}

int main(){

Il\_node \*head1=NULL,\*head2=NULL,\*head=NULL;

int n,m;

scanf("%d",&n);

while(n>0){

head1=aggordinati(head1,n);

scanf("%d",&n);

}

scanf("%d",&m);

while(m>0){

head2=aggordinati(head2,m);

scanf("%d",&m);

}

head=interseca(head1,head2);

print(head);

return 0;

}

/\*oppure

#include<stdio.h>

#include<stdlib.h>

struct el{int info; struct el \*next;};

typedef struct el \*Lista;

void add(Lista \*head, int n){

Lista new=(Lista)malloc(sizeof(struct el)), corr=\*head, prev=NULL;

new->info=n;

new->next=NULL;

if((\*head)==NULL){

new->next=(\*head);

(\*head)=new;

}else if((\*head)!=NULL){

int trovato=0;

while(corr!=NULL && trovato==0){

if(corr->info<n){

prev=corr;

corr=corr->next;

}else if(corr->info>=n) trovato=1;

}if(trovato==1){

if(prev==NULL){

new->next=(\*head);

(\*head)=new;

}else if(prev!=NULL){

new->next=corr;

prev->next=new;

}

}else if(trovato==0) prev->next=new;

}

}

int conta(Lista head, int n, int init){

Lista corr=head, prev=NULL;

int i=1, conta=0;

while(i<init){

corr=corr->next;

i++;

}prev=corr;

while(prev!=NULL){

if(prev->info==n) conta++;

prev=prev->next;

}

return conta;

}

int trova(Lista head, int n){

int trovato=0;

while(head!=NULL && trovato==0){

if(head->info==n) trovato=1;

head=head->next;

}

return trovato;

}

void intersezione(Lista \*head2, Lista \*head1, Lista \*head){

if((\*head)!=NULL){

Lista corr=\*head;

int inizio=1;

while(corr!=NULL){

if((conta((\*head), corr->info, inizio)==1)&&(trova((\*head1), corr->info)==1)) add(&(\*head2), corr->info);

inizio++;

corr=corr->next;

}

}

}

void print(Lista head){

while(head!=NULL){

printf("%d\n",head->info);

head=head->next;

}

}

int main(){

Lista head=NULL, head1=NULL, head2=NULL;

int n, m;

scanf("%d", &n);

while(n>=0){

add(&head, n);

scanf("%d", &n);

}

scanf("%d", &m);

while(m>=0){

add(&head1, m);

scanf("%d", &m);

}

intersezione(&head2, &head1, &head);

print(head2);

// printf("\n");

// print(head);

// printf("\n");

// print(head1);

return 0;

} \*/

LEZIONE 9

**Programma 9.1**

/\*Scrivere un programma che legga da tastiera un numero N e stampi l'N-esimo numero di Fibonacci calcolato con la formula di Binet. Secondo questa formula, l'N-esimo numero di Fibonacci puo' essere espresso come: F(N)=(fi^N -(-fi)^-N)/(5)^(1/2) con fi=(1+5^(1/2))/2. N.B. Arrotondare all'intero piu' vicino.

input: 10 output: 55 \*/

#include<stdio.h>

#include<math.h>

void binet(int v){

double fi=(1+sqrt(5))/2;

double ris=(pow(fi,v)-pow(fi\*-1,v\*-1))/(sqrt(5));

printf("%.0f\n",ris);

}

/\*

void fib(int valore){ //senza binet

int risultato;

if(valore==0)

risultato=0;

if(valore==1)

risultato=1;

else{

int indice=1;

int prec=0;

int corr=1;

while(indice<valore){

int t=prec+corr;

prec=corr;

corr=t;

indice++;

}

risultato=corr;

}

printf("%d\n",risultato);

}\*/

int main(){

int n;

scanf("%d",&n);

binet(n);

return 0;

}

**Programma 9.2**

/\*Si realizzi un programma C che prenda in ingresso una stringa S (contenente anche spazi, di al piu' 40 caratteri) e un carattere c. Il programma deve stampare in uscita il numero di occorrenze di c in S. N.B. c'e' differenza tra maiuscole e minuscole.

input: Hello World o output: 2 \*/

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

void contaoccorrenze(char \*s, char c){

int i=0, conta=0;

while(s[i]!='\0'){

if(s[i]==c){

conta++;

i++;

}

else if(s[i]!=c){

i++;

}

}

printf("%d\n",conta);

}

int main(){

char \*s=malloc(sizeof(char));

fgets(s,40,stdin); //legge al piu' 40 caratteri o comunque fino a \n compreso, compresi gli spazi tra le parole

char c;

scanf("%c",&c);

contaoccorrenze(s,c);

return 0;

}

**Programma 9.3**

/\*Si realizzi un programma C che prenda in ingresso una stringa S (di al piu' 40 caratteri e contenente anche spazi) e stampi la stessa stringa invertita. BONUS(no extra punti): farlo in place, ossia senza allocare/utilizzare spazio aggiuntivo per la stringa invertita ma modificando direttamente la stringa letta.

input: Stringa invertita output: atitrevni agnirtS \*/

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

void invertita(char \*s){

int i=0, lunghezza=0;

while(s[i]!='\0'){ //equivale a strlen(s);

lunghezza++;

i++;

}

for(i=lunghezza-2; i>=0; i--){

printf("%c",s[i]);

}

printf("\n");

}

int main(){

char \*s=malloc(sizeof(char));

fgets(s,40,stdin);

invertita(s);

return 0;

}

**Programma 9.4**

/\*Si realizzi un programma che legga una parola da tastiera e stampi 1 se la parola e' palindroma, 0 altrimenti. Ricordiamo che una parola e' palindroma se letta a rovescio rimane identica. N.B. la parola e' lunga al piu' 40 caratteri.

input: ingegni output: 1 \*/

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

void palindroma(char \*str){

int i=0, j=0, trovato=0;

char str2[strlen(str)];

i=strlen(str)-2;

do{

for(j=0; j<strlen(str); j++){

str2[j]=str[i];

i--;

}

}while(i>=0);

i=0;

while(i<=strlen(str)-2 && trovato==0){

if(str[i]!=str2[i])

trovato=1;

else

i++;

}

if(trovato==1)

printf("0\n");

else if(trovato==0)

printf("1\n");

}

int main(){

char \*s=malloc(sizeof(char)), \*s2=malloc(sizeof(char));

int i=0;

fgets(s,40,stdin);

palindroma(s);

return 0;

}

**Programma 9.5**

/\*Si realizzi un programma che legga da tastiera un intero n e successivamente n parole (una per riga). Il programma deve poi stampare prima tutte le parole palindrome (nell'ordine in cui sono state lette) e successivamente tutte le restanti parole (sempre in ordine di apparizione). N.B Ogni singola parola e' lunga al piu' 40 caratteri, e per allocare dinamicamente un array di stringhe dobbiamo prima allocare un array di char\*, e per ciascuna posizione dell'array, dobbiamo allocare lo spazio sufficiente per mantenere una stringa.

input: 5 anna prova otto ingegni ciao

output: anna otto ingegni prova ciao \*/

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

/\*

#define MAXLEN 1000

typedef struct Il\_node\_struct{ //non funziona se fatto con le liste

char \*info;

struct Il\_node\_struct \*next;

}Il\_node;

Il\_node \*insert(Il\_node \*head, char v[]){ //per inserire un el. in coda scorriamo l’intera lista

Il\_node \*corr=head;

Il\_node \*new\_node=malloc(sizeof(Il\_node));

new\_node->info=v;

new\_node->next=NULL;

if(head==NULL) //se la lista e’ vuota, gestire sempre con attenzione il caso della

return new\_node; //lista vuota

while(corr->next!=NULL) //scorriamo la lista fino a trovarne il nodo con il campo next

corr=corr->next; //cosi vado avanti con i nodi

corr->next=new\_node; //alla fine l’ultimo nodo a cui arrivo e’ l’ultimo

return head; //ritorna il puntatore al nodo head che restituisce la nuova lista

}

void print(Il\_node \*head){

while(head!=NULL){

printf("%s\n",head->info);

head=head->next;

}

} \*/

int palindroma(char str[]){

int i=0, j=0, trovato=0;

char str2[strlen(str)];

i=strlen(str)-2;

do{

for(j=0; j<strlen(str); j++){

str2[j]=str[i];

i--;

}

}while(i>=0);

i=0;

while(i<=strlen(str)-2 && trovato==0){

if(str[i]!=str2[i])

trovato=1;

else

i++;

}

return trovato;

}

/\*

void \*\*Make\_Array(int space){ //Crea un'array di puntatori, che puntano a delle stringhe

Il\_node \*head1=NULL,\*head2=NULL;

int i;

char \*\*array=(char\*\*)malloc(space\*sizeof(char\*));

for(i=0; i<space+1; i++){

array[i]=(char\*)malloc(40\*sizeof(char));

fgets(array[i],40,stdin);

if(palindroma(array[i])==0){

head1=insert(head1,array[i]);

}

else if(palindroma(array[i])==1){

head2=insert(head2,array[i]);

}

}

print(head1);

print(head2);

} \*/

int main(){

int space,i=0,j=1,ctr,z,c;

scanf("%d\n",&space);

char \*\*array;

//Make\_Array(space);

/\*int i=0, j=0;

for(i=0; i<space+1; i++){

printf("%s",array[i]);

}\*/

//First\_Palindrome(array,space);

//Destroy\_Array(space,array);

array=(char\*\*)malloc(space\*sizeof(char\*));

while(i<space){

array[i]=malloc(40\*sizeof(char));

fgets(array[i], 40, stdin);

i++;

}

i=0;

char \*sup;

while(j<space){

//se al posto di 0 metto 1 e al posto di 1 metto 0 stampa prima le non palindrome e poi le palindrome

if(palindroma(array[i])==0){

i++;

j++;

}

else{

if(palindroma(array[i])==1 && palindroma(array[j])==1)

j++;

else if(palindroma(array[i])==1 && palindroma(array[j])==0){

z=j-i;

c=j;

while(z>0){

sup=array[c];

array[c]=array[c-1];

array[c-1]=sup;

c--;

z--;

}

j++;

i++;

}

}

}

i=0;

while(i<space){

printf("%s",array[i]);

i++;

}

return 0;

}

**Programma 9.6**

/\*Si realizzi una funzione str3cat che accetta come input 3 stringhe (ciascuna lunga al piu' 40 caratteri) e ritorni una nuova stringa ottenuta come la concatenazione, separata da spazi, delle tre. Il programma deve leggere 3 stringhe e stampare in uscita le 3 stringhe seguite dal risultato dell'invocazione di str3cat sulle stesse.

input: ciao a tutti output: ciao a tutti ciao a tutti \*/

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#define MAXLEN 1000

void str3cat(char \*str1, char \*str2, char \*str3){

printf("%s\n",str1);

printf("%s\n",str2);

printf("%s\n",str3);

printf("%s %s %s\n",str1,str2,str3);

}

int main(){

char s1[MAXLEN],s2[MAXLEN],s3[MAXLEN];

scanf("%s",s1);

scanf("%s",s2);

scanf("%s",s3);

str3cat(s1,s2,s3);

return 0;

}

**Programma 9.7**

/\*Scrivere un programma che legga da tastiera un numero n compreso tra 20 e 89 e stampi il suo valore espresso come stringa.

input: 23 output: ventitre

input: 81 output: ottantuno \*/

#include<stdio.h>

#include<stdlib.h>

void funzione(int value){

int i=0;

char \*\*decine=(char \*\*)malloc(9\*sizeof(char \*));

for(i=0; i<9; i++){

decine[0]="dieci";

decine[1]="dieci";

decine[2]="venti";

decine[3]="trenta";

decine[4]="quaranta";

decine[5]="cinquanta";

decine[6]="sessanta";

decine[7]="settanta";

decine[8]="ottanta";

}

char \*\*unita=(char\*\*)malloc(10\*sizeof(char\*));

for(i=0; i<10; i++){

unita[0]="zero";

unita[1]="uno";

unita[2]="due";

unita[3]="tre";

unita[4]="quattro";

unita[5]="cinque";

unita[6]="sei";

unita[7]="sette";

unita[8]="otto";

unita[9]="nove";

}

int newvalue=0;

if(value/10 != 0){

newvalue=value/10;

printf("%s", decine[newvalue]);

}if(value%10!=0){

newvalue=value%10;

printf("%s", unita[newvalue]);

}

printf("\n");

}

int main(){

int n;

scanf("%d", &n);

printf("%d\n", n);

funzione(n);

return 0;

}

LEZIONE 10

**Programma 10.1**

/\*Dato lo scheletro di codice qui riportato (e che puo' essere scaricare dagli allegati), lo si completi aggiungendo l'implementazione delle funzioni richieste.

#include<stdio.h>

#include<stdlib.h>

int \*leggiInput(int \*dim); //funzioni da implementare:

int \*eliminaDup(int \*vect, int dim, int \*dim\_nodup);

int ugualeASomma(int \*vect, int dim);

int \*maggioreDeiSuccessivi(int \*vect, int dim);

int main(){

int \*input, \*nodup, \*results, dim, dim\_nodup, i;

input=leggiInput(&dim); //legge l'input

printf("Stampa dei valori in input: (%d valori)\n",dim);

for(i=0; i<dim; i++)

printf("%d\n",input[i]);

nodup=eliminaDup(input,dim,&dim\_nodup); //elimina i duplicati

printf("Stampa dei valori senza duplicati: (%d valori)\n",dim\_nodup);

for(i=0; i<dim\_nodup; i++)

printf("%d\n",nodup[i]);

printf("Risultato di ugualeASomma: %d\n", ugualeASomma(nodup,dim\_nodup)); //esegue ugualeASomma

results=maggioreDeiSuccessivi(nodup,dim\_nodup); //esegue maggioreDeiSuccessivi

printf("Risultato maggioreDeiSuccessivi:\n");

for(i=0; i<dim\_nodup; i++)

printf("%d\n", results[i]);

return 0;

}

Le funzioni devono implementare le seguenti operazioni:

-leggiInput: legge un numero n e successivamente n interi, inserendoli in un array nell'ordine in cui vengono letti. La funzione restituisce l'array e scrive, nell'indirizzo puntato da dim, il valore n, ovvero la dimensione dell'array;

-eliminaDup: la funzione accetta come parametri un array di interi vect e la sua dimensione dim. Deve creare e restituire un nuovo array di interi contenente solo la prima occorrenza degli elementi contenuti nell'array iniziale (mantenendo il loro ordine). Nell'indirizzo puntato da dim\_nodup viene memorizzata la dimensione del nuovo array;

-ugualeASomma: la funzione accetta come parametri un array di interi vect e la sua dimensione dim. Restituisce 1 se viene soddisfatta la proprieta': esiste i appartenente a [0,dim) tale che vect[i]=alla sommatoria per j appartenente a [0,dim) e j!=i del vect[j] oppure 0 in caso contrario;

-maggioreDeiSuccessivi: la funzione accetta come parametri un array di interi vect e la sua dimensione dim. Ritorna un array di interi, dove ciascuna posizione i assume valore 1 se vale la seguente proprieta': per ogni j appartenente a [i+1,dim) tale che vect[i]>vect[j] mentre assume valore 0 in caso contrario. Si noti che per l'ultimo elemento dell'array la condizione e' sempre soddisfatta.

input: 6 2 2 3 10 4 1

output: Stampa dei valori in input: (6 valori) 2 2 3 10 4 1 Stampa dei valori senza duplicati: (5 valori) 2 3 10 4 1 Risultato di ugualeASomma: 1 Risultato maggioreDeiSuccessivi: 0 0 1 1 1 \*/

#include<stdio.h>

#include<stdlib.h>

// Funzioni da implementare

int \*leggiInput(int \*dim){

scanf("%d", &(\*dim)); //oppure scanf("%d", dim);

int \*a=(int \*)malloc((\*dim)\*sizeof(int)), i;

for(i=0; i<\*dim; i++) scanf("%d", &a[i]);

return a;

}

int \*eliminaDup(int \*vect, int dim, int \*dimnodup){

if(dim!=0){

\*dimnodup=1;

int \*a=(int \*)malloc((\*dimnodup)\*sizeof(int)), i, j, conta=0;

a[0]=vect[0];

for(i=1; i<dim; i++){

for(j=0; j<\*dimnodup; j++) if(vect[i]==a[j]) conta++;

if(conta==0){

(\*dimnodup)++;

a=realloc(a,(\*dimnodup)\*sizeof(int));

a[(\*dimnodup)-1]=vect[i];

}

conta=0;

}

return a;

}

}

int ugualeASomma(int \*vect, int dim){

//per calcolare questa funzione non posso ordinare l'array

int i=0, somma=0, j, trovato=0;

while(i<dim && trovato==0){

for(j=0; j<dim; j++) if(i!=j) somma=somma+vect[j];

if(vect[i]==somma) trovato=1;

else{

somma=0;

i++;

}

}

return trovato;

}

int \*maggioreDeiSuccessivi(int \*vect, int dim){

int i, j, \*a=(int \*)malloc(dim\*sizeof(int)), trovato=0;

for(i=0; i<dim-1; i++){

j=i+1;

while(j<dim && trovato==0){

if(vect[i]>vect[j]) j++;

else trovato=1;

}

if(trovato==1) a[i]=0;

else a[i]=1;

trovato=0;

}

a[dim-1]=1;

return a;

}

int main(){

int \*input, \*nodup, \*results, dim, dimnodup, i;

//Legge l’input

input=leggiInput(&dim);

printf("Stampa dei valori in input: (%d valori)\n", dim);

for(i=0; i<dim; i++) printf("%d\n", input[i]);

//Elimina i duplicati

nodup=eliminaDup(input, dim, &dimnodup);

printf("Stampa dei valori senza duplicati: (%d valori)\n", dimnodup);

for(i=0; i<dimnodup; i++) printf("%d\n", nodup[i]);

//Esegue ugualeASomma

printf("Risultato di ugualeASomma: %d\n", ugualeASomma(nodup, dimnodup));

//Esegue maggioreDeiSuccessivi

results=maggioreDeiSuccessivi(nodup, dimnodup);

printf("Risultato maggioreDeiSuccessivi:\n");

for(i=0; i<dimnodup; i++) printf("%d\n", results[i]);

return 0;

}

**Programma 10.2**

/\*Dato lo scheletro di codice qui riportato, lo si completi aggiungendo l'implementazione delle funzioni richieste.

#include<stdio.h>

#include<stdlib.h>

int \*leggiInput(int \*dim); //funzioni da implementare

int \*eliminaNeg(int \*vect, int dim, int \*newdim);

float \*mediaEstremi(int \*vect, int dim);

int main(){

int \*input, \*noneg, dim, dimnoneg, i;

float \*results;

input=leggiInput(&dim); //legge l'input

printf("Stampa dei valori in input: (%d valori)\n",dim);

for(i=0; i<dim; i++) printf("%d\n", input[i]);

noneg=eliminaNeg(input, dim, &dimnoneg); //elimina i negativi

printf("Stampa dei valori senza negativi: (%d valori)\n", dimnoneg);

for(i=0; i<dimnoneg; i++) printf("%d\n",noneg[i]);

results=mediaEstremi(noneg,dimnoneg); //esegue media degli estremi

dim=dimnoneg%2==0? dimnoneg/2: dimnoneg/2+1; //calcolo dimensione dell'array results

printf("Risultato di mediaEstremi:\n");

for(i=0; i<dim; i++) printf("%3.2f\n,results[i]");

return 0;

}

Le funzioni devono implementare le seguenti operazioni:

-leggiInput: legge una sequenza di interi (positivi e negativi) terminata da uno 0 e li inserisce in un array nell'ordine in cui vengono letti. Lo 0 comparira' solo come terminatore della sequenza e non dovra' essere inserito nell'array. Si puo' assumere che la sequenza possa prevedere al massimo 20 elementi, escluso lo 0. La funzione restituisce l'array e scrive, nell'indirizzo puntato da dim, la dimensione dell'arrau letto (ovvero il numero di elementi letti da tastiera diversi da 0);

-eliminaNeg: la funzione accetta come parametri un array di interi vect, la sua dimensione dim, e un puntatore ad interi new\_dim. La funzione modifica l'array vect sostituendo ogni occorrenza di un numero negativo con il suo valore assoluto ed aggiungendo in fondo all'array un elemento contenente, come il precedente, il valore assoluto del numero negativo. Supponendo che il primo numero negativo si trovi in posizione i nell'array vect, ne consegue che il valore di vect[i] verra' trasformato in positivo, mentre verra' aggiunto un ulteriore elemento abs(vect[i])in coda all'array. L'inserimento degli elementi in coda all'array deve avvenire rispettando l'ordine dei corrispondenti numeri negativi nel vettore vect iniziale. Ad esempio, dato l'array {4, -1, 2, -3}, questo verra' trasformato nell'array {4, 1, 2, 3, 1, 3}. La funzione restituisce il nuovo array dopo il ridimensionamento e scrive, nell'indirizzo puntato da new\_dim, la dimensione del nuovo array.

-mediaEstremi: la funzione accetta come parametri un array di interi vect e la sua dimensione dim. Restituisce un array di float, il cui elemento i-esimo contiene la media tra gli elementi di vect in posizione i a partire dall'inizio e dalla fine dell'array. Ad esempio, in posizione i a partire dall'inizio e dalla fine dell'array. Ad esempio, in posizione 0 sara' contenuta la media tra il primo e l'ultimo elemento di vect, in posizione 1 tra il secondo e il penultimo, in posizione 2 tra il terzo e il terz'ultimo, etc. In caso l'array vect sia di dimensione dispari, ricopiare l'elemento centrale come ultimo valore dell'array delle medie.

input: 2 -1 4 7 -8 1 3 0

output: Stampa dei valori in input: (7 valori) 2 -1 4 7 -8 1 3 Stampa dei valori senza negativi: (9 valori) 2 1 4 7 8 1 3 1 8 Risultato di mediaEstremi: 5.00 1.00 3.50 4.00 8.00 \*/

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

int \*leggiInput(int \*dim){

int \*a=malloc(20\*sizeof(int)), i=0, dim2=0, n;

scanf("%d", &n);

while(n!=0){

a[i]=n;

i++;

scanf("%d",&n);

dim2++;

}

\*dim=dim2;

return a;

/\* //oppure

\*dim=0;

int n, \*a=(int \*)malloc((\*dim)\*sizeof(int)), i=0;

scanf("%d", &n);

while(n!=0){

(\*dim)++;

a=realloc(a,(\*dim)\*sizeof(int));

a[i]=n;

scanf("%d", &n);

i++;

}

return a;

\*/

}

int \*eliminaNeg(int \*vect, int dim, int \*newdim){

int i=0, valore=0;

for(i=0; i<dim; i++){

if(vect[i]<0){

dim++;

vect=realloc(vect,(dim)\*sizeof(int));

vect[i]=abs(vect[i]);

vect[dim-1]=abs(vect[i]);

}

}

\*newdim=dim;

return vect;

/\* //oppure

(\*newdim)=dim;

int i;

for(i=0; i<(\*newdim); i++){

if(vect[i]<0){

vect[i]=vect[i]\*(-1);

(\*newdim)++;

vect=realloc(vect, (\*newdim)\*sizeof(int));

vect[(\*newdim)-1]=vect[i];

}

}return vect;

\*/

}

float \*mediaEstremi(int \*vect, int dim){

float \*c=realloc(vect,dim\*sizeof(float));

int i=0, j=dim-1, n=0;

while(n<dim){

float a=vect[i], b=vect[j], media=(a+b)/2; //se eseguo solo media=(vect[i]+vect[j])/2 restituisce solo interi come risultati.

c[n]=media;

n++;

j--;

if(j==i)

break;

i++;

if(j==i){

c[n]=vect[j];

break;

}

}

return c;

/\* //oppure

int newdim=0, i=0, j=0, k=0;

if(dim%2==0) newdim=dim/2;

else if(dim%2!=0) newdim=(dim/2)+1;

float \*v=(float \*)malloc(newdim\*sizeof(float));

j=dim-1;

while(i<=j){

v[i]=(float)(vect[i]+vect[j])/2;

i++;

j--;

}

return v;

\*/

}

int main(){

int \*input, \*noneg, dim, dimnoneg, i;

float \*results;

//Legge l’input

input=leggiInput(&dim);

printf("Stampa dei valori in input: (%d valori)\n", dim);

for(i=0; i<dim; i++) printf("%d\n", input[i]);

//Elimina i negativi

noneg=eliminaNeg(input, dim, &dimnoneg);

printf("Stampa dei valori senza negativi: (%d valori)\n", dimnoneg);

for(i=0; i<dimnoneg; i++) printf("%d\n", noneg[i]);

//Esegue media degli estremi

results=mediaEstremi(noneg, dimnoneg);

//Calcolo dimensione dell’array results

dim=dimnoneg%2==0?dimnoneg/2:dimnoneg/2+1;

printf("Risultato di mediaEstremi:\n");

for(i=0; i<dim; i++) printf("%3.2f\n", results[i]);

return 0;

}

**Programma 10.3**

/\*GESTORE CORSA 1000 METRI: Si implementi un gestore di gare di corsa. La gara, a cui partecipa un numero variabile di atleti, consiste in una corsa lunga 1000 metri. Gli atleti partono tutti insieme e i loro tempi parziali vengono misurati ogni 100 metri. Per ogni atleta vi saranno dunque 10 misurazioni, l'ultima delle quali e' il tempo di raggiungimento del traguardo. Il programma prende in input le informazioni sugli atleti e i loro tempi parziali misurati durante una gara, ed elabora delle statistiche. Dato lo scheletro di codice qui riportato (e che puo' essere scaricato dagli allegati), lo si completi aggiungendo l'implementazione delle funzioni richieste.

#include<stdio.h>

#include<stdlib.h>

typedef struct{

int tempi[10]; //Tempi misurati ogni 100 metri

char nome[20]; //Nome dell'atleta

}corridore;

corridore \*leggiInput(int \*dim); //funzioni da implementare

int trovavincitore(corridore \*vect, int dim);

int \*tempitointevalli(corridore c);

int \*trovasprint(corridore c);

int main(){

corridore \*input;

int dim, dimsprint, vincitore, i, j, \*intervalli, \*sprint;

input=leggiInput(&dim); //legge l'input

printf("Stampa dei valori in input: (%d atleti)\n",dim);

for(i=0; i<dim; i++){

printf("Tempi per l'atleta %s:",input[i].nome);

for(j=0; j<10; j++) printf("%d",input[i].tempi[j]);

printf("\n");

}

vincitore=trovavincitore(input, dim); //trova il vincitore

printf("Nome del vincitore: %s (migliore tempo: %d secondi)\n",input[vincitore].nome,input[vincitore].tempi[9]);

for(i=0; i<dim; i++){ //converti da tempi ad intervalli

intervalli=tempitointervalli(input[i]);

printf("Intervalli per l'atleta %s:",input[i].nome);

for(j=0; j<10; j++) printf("%d",intervalli[j]);

printf("\n");

free(intervalli);

}

for(i=0; i<dim; i++){ //trova gli sprint

sprint=trovasprint(input[i]);

printf("Sprint per l'atleta %s:",input[i].nome);

for(j=0; j<10; j++) printf("%d",sprint[j]);

printf("\n");

free(sprint);

}

return 0;

}

Le funzioni devono implementare le seguenti operazioni:

-leggiInput: legge un numero n e successivamente le informazioni riguardanti n atleti, inserendoli in un array nell'ordine in cui vengono letti. Per ogni atleta, vengono letti prima il suo nome (una parola che consiste solo di caratteri senza spazi, lunga in tutto meno di 20 caratteri), quindi 10 interi, corrispondenti alle misurazioni dei tempi parziali, in secondi. La funzione retituisce l'array che contiene, per ogni atleta, una struttura corridore, contenente le informazioni lette. Inoltre, la funzione scrive, nell'indirizzo puntato da dim, il valore n, ovvero il numero di elementi dell'array restituito.

-trovavincitore: la funzione accetta come parametri un array di strutture corridore e la sua dimensione dim, e restituisce l'indice (posizione nell'array) dell'atleta che ha vinto la gara.

-tempitointervalli: la funzione accetta come parametri un corridore c. A partire dai tempi parziali del corridore, la funzione genera e restituisce un array contenente i tempi di percorrenza di ogni tratto. Ad esempio, se i tempi di percorrenza del corridore sono T=[10,20,30,40,49,61,72,82,92,102], l'array risultante sara' I=[10,10,10,10,9,12,11,10,10,10].

-trovasprint: la funzione accetta come parametro una struttura corridore. A partire dai tempi parziali, restituisce un array A contenente 10 interi di valore 0 o 1. L'elemento i-esimo di A avra' valore 1 se nel tratto i l'atleta ha fatto uno sprint, ovvero ha percorso il tratto in meno della meta' del tempo rispetto alla media degli altri intervalli. Varra' 0 altrimenti.

input: 3 output: Stampa dei valori in input: (3 atleti)

mohammed\_ali 12 24 32 45 50 60 70 81 92 97 Tempi per l'atleta mohammed\_ali: 12 24 32 45 50 60 70 81 92 97

salvio\_berluschini 12 22 30 43 48 55 60 71 82 86 Tempi per l'atleta salvio\_berluschini: 12 22 30 43 48 55 60 71 82 86

renzo\_mattei 7 15 22 32 43 54 63 76 85 110 Tempi per l'atleta renzo\_mattei: 7 15 22 32 43 54 63 76 85 110

Nome del vincitore: salvio\_berluschini (miglior tempo: 86 secondi)

Intervalli per l'atleta mohammed\_ali: 12 12 8 13 5 10 10 11 11 5

Intervalli per l'atleta salvio\_berluschini: 12 10 8 13 5 7 5 11 11 4

Intervalli per l'atleta renzo\_mattei: 7 8 7 10 11 11 9 13 9 25

Sprint per l'atleta mohammed\_ali: 0 0 0 0 1 0 0 0 0 1

Sprint per l'atleta salvio\_berluschini: 0 0 0 0 0 0 0 0 0 1

Sprint per l'atleta renzo\_mattei: 0 0 0 0 0 0 0 0 0 0 \*/

#include<stdio.h>

#include<stdlib.h>

typedef struct{

int tempi[10]; //Tempi misurati ogni 100 metri

char nome[20]; //Nome dell'atleta

}corridore;

corridore \*leggiInput(int \*dim){

int i, j;

scanf("%d", dim);

corridore \*sportivi=(corridore \*)malloc((\*dim)\*sizeof(corridore));

for(i=0; i<\*dim; i++){

scanf("%s", sportivi[i].nome);

for(j=0; j<10; j++) scanf("%d", &sportivi[i].tempi[j]);

}

return sportivi;

}

int trovavincitore(corridore \*vect, int dim){

int menotempo=1000, i, s=0;

for(i=0; i<dim; i++){

if(vect[i].tempi[9]<menotempo){

menotempo=vect[i].tempi[9];

s=i;

}

}

return s;

/\* //in modo piu' complicato

int i=0,j=0, \*max=malloc(dim\*sizeof(int)), min=0;

if(dim==1) return 0;

else if(dim>1){

for(i=0; i<dim; i++) max[i]=vect[i].tempi[9];

for(i=0; i<dim; i++) if(max[i]>max[i+1]) min=max[i+1];

i=0;

while(i<dim){

if(max[i]==min) return i;

else if(max[i]!=min) i++;

}

}

return i;

\*/

}

int \*tempitointervalli(corridore c){

//time lo devo per forza dichiarare cosi, e non time[10] altrimenti la considera una variabile

int j, \*time=calloc(1000,10\*sizeof(int));

// oppure

// int \*time=calloc(10,sizeof(int);

// int \*time=(int \*)malloc(10\*sizeof(int));

// for(i=0; i<10; i++) time[i]=0; //non e' necessario

time[0]=c.tempi[0];

for(j=1; j<10; j++) time[j]=c.tempi[j]-c.tempi[j-1];

return time;

}

int \*trovasprint(corridore c){

int i, \*a=malloc(10\*sizeof(int)), \*time=tempitointervalli(c), somma=0, media=0;

for(i=0; i<10; i++) somma=somma+time[i];

media=somma/18;

//printf("%d ",media);

// for(i=0; i<10; i++) a[i]=0; //non necessario

for(i=0; i<10; i++){

if(time[i]>media) a[i]=0;

else a[i]=1;

}

return a;

}

int main(){

corridore \*input;

int dim, dimsprint, vincitore, i, j, \*intervalli, \*sprint;

//Legge l’input

input=leggiInput(&dim);

printf("Stampa dei valori in input: (%d atleti)\n", dim);

for(i=0; i<dim; i++){

printf("Tempi per l'atleta %s: ", input[i].nome);

for(j=0; j<10; j++) printf("%d ", input[i].tempi[j]);

printf("\n");

}

//Trova il vincitore

vincitore=trovavincitore(input, dim);

printf("Nome del vincitore: %s (miglior tempo: %d secondi)\n", input[vincitore].nome, input[vincitore].tempi[9]);

//Converti da tempi ad intervalli

for(i=0; i<dim; i++){

intervalli=tempitointervalli(input[i]);

printf("Intervalli per l'atleta %s: ", input[i].nome);

for(j=0; j<10; j++)

printf("%d ", intervalli[j]);

printf("\n");

free(intervalli);

}

//Trova gli sprint

for(i=0; i<dim; i++){

sprint=trovasprint(input[i]);

printf("Sprint per l'atleta %s: ", input[i].nome);

for(j=0; j<10; j++)

printf("%d ", sprint[j]);

printf("\n");

free(sprint);

}

return 0;

}