**Programma 01**

//il programma stampa due colonne in cui nella prima ci sono i numeri fino a 25 e nella seconda il loro quadrato

#include<stdio.h>

int main(){

int i=0;

printf("\t Number \t\t Square of Number\n\n");

for(i=0; i<=25; ++i)

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

return 0;

}

//oppure

/\*Questo programma utilizza la funzione square per calcolare il quadrato dei numeri fino a 10.\*/

#include<stdio.h>

int square(int);

int main(){

int x;

for(x=1; x<=10; x++)

printf("Il numero %d e' quadrato di %d\n",square(x),x);

return 0;

}

int square(int y){

return y\*y;

}

**Programma 02**

//stampa due colonne, nella prima i numeri fino a 25 e nella seconda la radice di essi, con approssimazione a 4 numeri dopo la virgola, se non specificato sarebbe stato di 6 numeri dopo la virgola.

#include <stdio.h>

#include <math.h> //serve per usare la sqrt()

int main(){

int i=0;

printf("\t Number \t\t Square Root of Number\n\n");

for(i=0; i<=25; ++i)

printf("\t %d \t\t\t %.4lf \n",i, sqrt((double) i)); //la SquareRootFunction vuole un numero in double con segnaposto %lf

return 0;

}

**Programma 03**

//spiegazione del comportamento comandi break e continue

#include <stdio.h>

int main(){

int value;

while(scanf("%d",&value)==1 && value!=0){

if(value<0){

printf("Illegal value \n");

break; /\* Abandon the loop \*/

}

if(value>100){

printf("Invalid value \n");

continue; /\* Skip to start loop again \*/

}

} /\* end while value != 0 \*/

return 0;

}

**Programma 04**

//stampa i numeri da 0 a 4 tabulati per 5 volte

#include <stdio.h>

int main(){

int x,y,table[5][5];

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

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

printf("%d\t",y);

printf("\n");

}

return 0;

}

**Programma 05**

#include <stdio.h>

void stat(){

int auto\_var=0;

static int static\_var = 0; //dichiarazione variabile statica

printf("auto = %d, static = %d \n",auto\_var, static\_var);

++auto\_var; //Clearly the auto\_var variable is created each time.

++static\_var; //The static\_var is created once and remembers its value.

}

int main(){

int i;

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

stat();

return 0;

}

**Programma 06**

//come funziona la strlen()

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

int strlengg(char s[]){ //no strlen perche' gia' presente in libreria -> deve avere nome diverso

char \*p = s;

while(\*p!='\0')

p++; //usa un puntatore per restituire il numero di caratteri, poteva usare

return p-s; //anche un intero

}

/\*int strlengg(char s[]){

int i=0, conta=0;

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

conta++;

s[i++];

}

return conta;

}

int main(){

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

fgets(str,20,stdin);

int risultato=strlengg(str)-1; //togliendo lo spazio finale di chiusura

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

return 0;

}

**Programma 07**

//il programma calcola l’area di un rettangolo

#include<stdio.h>

#include<math.h>

int main(){

double base, height, area;

printf("Enter the sides of rectangle: \n");

scanf("%lf %lf", &base, &height);

area=base\*height;

printf("Area of Rectangle=%.2lf\n", area);

return 0;

}

//il programma calcola l’area di un cerchio

#include<stdio.h>

#define PI (3.1415f)

int main(){

float r; //raggio del cerchio

float a; //area del cerchio

scanf("%f",&r);

a=PI\*r\*r;

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

return 0;

}

**Programma 08**

//Il programma stampa su schermo Ciao Mondo!

#include <stdio.h>

int main(){

printf("Ciao Mondo!\n");

return 0;

}

**Programma 09**

#include <stdio.h>

#define MATRICOLA 536295

int main(){

int a;

a=19;

printf("Ciao Mondo,ho %d anni e il mio numero di matricola e'%d\n",a,MATRICOLA);

scanf("%d",&a);

return 0;

}

**Programma 10**

#include <stdio.h>

int main(){

int i;

i=1;

printf("Ciao mondo. Vediamo fino a quanto so contare:\n");

i=i+1;

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

i++;

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

i++;

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

i+=2;

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

printf("temo di non arrivare nemmeno a %d\n",--i);

return 0;

}

**Programma 11**

#include<stdio.h>

int main(){

printf("Immettere un anno: ");

int n;

scanf("%d",&n);

if(n%400==0 ||(n%100!=0 && n%4==0)){

printf("Bisestile si!\n");

}

else

printf("No bisestile!\n");

return 0;

}

**Programma 12**

//Il programma stampa su schermo il numero digitato sommato con il numero di Nepero, inoltre quando si compila con gcc si deve usare: nome.c -o nome.o -lm perché in uso la funzione di libreria math.h

#include<stdio.h>

#include<math.h>

int main(){

int n;

scanf("%d",&n);

float somma=0;

somma=n+M\_E; //sommo con il numero di nepero

printf("Il risultato e':%f\n",somma);

return 0;

}

**Programma 13**

//Questo programma genera un numero casuale da 0 a 100 (in un intervallo)

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

int main(){

int i;

//La funzione time con argomento NULL restituisce i secondi trascorsi dal 1.1.1970 (epoch). Ad ogni esecuzione (purche' non troppo vicine) otterremo un seme diverso.

srand(time(NULL));

i=rand()%100+1;

printf("Il numero casuale generato e':%d\n",i);

return 0;

}

//oppure

//Questo programma genera un numero casuale molto grande

#include<stdio.h>

#include<stdlib.h>

int main(){

int numero=0;

//inizializzando il seme della funzione srand() ad un valore diverso da NULL non serve la funzione #include<time.h>

srand(42);

//il numero casuale cosi generato assume valori molto grandi, se vogliamo invece un numero compreso tra un intervallo tra 0 e L dobbiamo usare l'operatore modulo %L.

numero=rand();

printf("Il numero casuale e' %d\n",numero);

return 0;

}

//oppure

//Questo programma genera un numero casuale molto grande

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

int main(){

int numero=0;

srand(time(NULL));

numero=rand();

printf("Il numero casuale generato e':%d\n",numero);

return 0;

}

**Programma 14**

//Scrivere un programma che genera gli elementi di un array con interi casuali da 0 a 100 (in un intervallo)

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

int main(){

int i, \*a=malloc(10\*sizeof(int));

//Inizializzo il generatore di numeri casuali con seme time(NULL) ovvero la funzione time con argomento NULL

srand(time(NULL));

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

a[i]=rand()%100+1; //se voglio un intero tra 0 e 99: i=rand()%100;

printf("Il numero casuale generato alla posizione %d e':%d\n",i, a[i]);

}

return 0;

}

**Programma 15**

/\*Esempio di richiamo di una funzione. Le direttive del preprocessore possono essere scritte in qualsiasi parte del programma anche se e' buon uso metterle all'inizio. La dichiarazione di una funzione avviene prima della sua definizione, quest'ultima puo' avvenire anche dopo il main anche se e' buon uso farle insieme. Non dimenticarsi il return in una funzione.\*/

#include<stdio.h>

int somma(int x, int y){

return (x+y);

}

int main(){

printf("%d\n",somma(40,2));

return 0;

}

**Programma 16**

//elevamento a potenza

#include<stdio.h>

int power(int n, int m){

if(n==0)

return 1; //qualsiasi elemento elevato alla 0 restituisce 1

else

return m\*power(n-1,m);

}

int main(){

int x, y;

scanf("%d %d",&x,&y);

printf("%d\n",power(y,x)); //stampa x^y

return 0;

}

**Programma 17**

//ordinamento di 3 valori

#include<stdio.h>

int main(){

int n, m, o, i, min, medio, mass;

printf("Scrivi 3 valori: ");

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

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

if(n<m && n<o)

min=n;

if(m<n && m<o)

min=m;

if(o<n && o<m)

min=o;

if((n>m && n<o) || (n<m && n>o))

medio=n;

if((m>n && m<o) || (m<n && m>o))

medio=m;

if((o>n && o<m) || (o<n && o>m))

medio=o;

if(n>m && n>o)

mass=n;

if(m>n && m>o)

mass=m;

if(o>n && o>m)

mass=o;

}

printf("Eccoli in ordine crescente: %d %d %d\n", min, medio, mass);

printf("Eccoli in ordine decrescente: %d %d %d\n", mass, medio, min);

return 0;

}

**Programma 18**

/\*SOTTOARRAY DI SOMMA MASSIMA

Dato un array a di n interi (positivi e negativi) individuare il sottoarray di somma massima.

-input = array a di interi

-output = la somma del sottoarray di somma massima

Realizzare la soluzione ottima sapendo che:

-la somma dei valori in ogni prefisso dl sottoarray ottimo e' positiva, se cosi' non fosse potremmo eliminare tale prefisso ottenendo un sottoarray di somma maggiore (assurdo)

-il valore immediatamente precedente al primo valore del sottoarray ottimo e' negativo, se cosi' non fosse potremmo aggiungere tale valore ottenendo un sottoarray di somma maggiore (assurdo)

\*/

#include<stdio.h>

#include<stdlib.h>

//soluzione cubica

/\*

int sommamassima(int a[], int dim){

int max=a[0], i, j, k, somma=0;

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

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

somma=0;

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

somma=somma+a[k];

if(somma>max)

max=somma;

}

}

return max;

}

\*/

//soluzione cubica

/\*

int sommamassima(int a[], int dim){

int max=a[0], i, j, somma=0;

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

somma=0;

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

somma=somma+a[j];

if(somma>max)

max=somma;

}

}

return max;

}

\*/

//soluzione lineare

int sommamassima(int a[], int dim){

int max=a[0], i, somma=0;

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

if(somma>0)

somma=somma+a[i];

else

somma=a[i];

if(somma>max)

max=somma;

}

return max;

}

int main(){

int n, i;

scanf("%d", &n);

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

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

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

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

return 0;

}

**Programma 19**

//ARITMETICA DEI PUNTATORI

#include<stdio.h>

#include<string.h> //ci sono le funzioni per manipolare le stringhe

#include<stdlib.h> //per array dinamici

int main(){

int tabella[4]={10,20,30,40}; //e' un array di 4 interi

int \*p;

p=&tabella[0]; //oppure

p=tabella;

//p e' un puntatore al primo elemento dell'array, oppure essendo l'array un puntatore costante, p e' un puntatore al primo elemento di una sequenza fissata di elementi. Di fatto un'array e' un puntatore particolare: non possiamo associare ad esempio ad un array il valore di un altro indirizzo

p=p+1; //se incrementiamo il puntatore

printf("%d\n",\*p); //il risultato NON e' di incrementare di 1 il valore del puntatore, bensi' di un numero di bytes equivalenti all'ampiezza di un intero, cioe' 4. In questo modo la variabile puntatore puntera' al secondo elemento dell'array(questo perche' gli elementi di un array sono memorizzati consecutivamente). Quindi con i puntatori si scorrono gli elementi di un array.

p=p-1; //se decrementiamo il puntatore

printf("%d\n",\*p); //analogamente, l'array viene percorso all'indietro

int \*puntatore1=&tabella[0];

int \*puntatore2=&tabella[3];

//la sottrazione tra puntatori conta gli elementi che separano i puntatori,avra' sempre tipo intero

int i=puntatore2-puntatore1;

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

//analogamente se dovessimo sommare puntatore1 con 3 otterremo puntatore2

int \*a[10]; //array di 10 puntatori a int

int (\*p)[10]; //puntatore ad un array di 10 int

int \*f(int); // funzione che prende un int e restituisce un puntatore a int

int (\*p)(int); //puntatore a funzione che prende e restituisce un int

int \*a[](int); //array di funzioni che prendono un int e restituiscono un puntatore a int

int (\*a[])(int); //array di puntatori a funzioni che prendono e restituiscono un int

int (\*p)[](int); //puntatore ad un array di funzioni che prendono e restituiscono un int

char stringa[7]={'O','r','n','e','l','a','\0'};//specificando la dimensione

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

char string[]={'L','a','g','o',' ','B','o','r','a','c','i','f','e','r','o','\0'};//non specificando la dimensione

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

char \*s;

char \*token; //sottostringa

char \*temp="prima|seconda|terza|quarta";

s=(char \*)malloc((strlen(temp)+1)\*sizeof(char));

strcpy(s,temp); //realizza la copia della stringa s nella temp compreso il carattere terminatore

while((token=strsep(&s,"|"))!=NULL)

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

char \*p="abc"; //Le costanti di tipo stringa o stringa letterale vengono scritte tra doppi apici e vengono trattate come puntatori, quest'ultimi sono delle variabili che contengono l'indirizzo di memoria di un'altra variabile, in questo caso il puntatore e' la variabile p che contiene l'indirizzo di memoria di un carattere, o meglio, di una costante di tipo stringa. Mentre per conoscere l'indirizzo di una variabile si utilizza l'operatore & tramite il quale accediamo indirettamente alla variabile p. Per dereferenziare (accedere alla locazione di memoria puntata) si utilizza l'operatore \*

printf("%s %s %s\n",p,p+1,p+2);

int i=42;

int \*puntatore=&i; //assegnamento della variabile puntatore l'indirizzo di i

\*puntatore=1; //modifica indirettamente il contenuto della variabile i

printf("%d\n",i); //al termine il valore di i e' 1 sebbene non sia stato modificato direttamente

//questi tre modi seguenti sono equivalenti:

char arr1[]="abc";

char arr2[4]="abc";

char arr3[]={'a','b','c','\0'}; //in questo caso l'array non e' costante per cui e' liberamente modificabile.Una stringa e' un array di caratteri che contiene la sequenza di caratteri che forma la stringa seguita dal carattere speciale di fine stringa: '\0'. Se quest'ultimo carattere non c'e' e' un array

char \*buffer=(char \*)malloc(50\*sizeof(char));

scanf("%s",buffer); // Con questa istruzione vengono letti da input i caratteri in sequenza fino a trovare il primo carattere bianco, oppure un tab, un interlinea ecc. Questi caratteri vengono messi dentro il vettore buffer, al posto del carattere bianco, viene posto il carattere'\0'. Il vettore deve essere sufficientemente grande da contenere tutti i caratteri letti. NON si usa &buffer ma direttamente buffer! perche' buffer essendo di tipo char\* e' gia' un indirizzo!!

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

return 0;

}

**Programma 20**

/\*Viene dato un array di dimensione n contenente non ordinati tutti gli interi compresi tra 1 e n+1 ad eccezione di uno di essi e si vuole stabilire l'elemento mancante. Sono possibili almeno 4 soluzioni aventi le seguenti complessita':

-in tempo O(n^2) in spazio O(1)

-in tempo O(nlog(n)) in spazio O(n)

-in tempo O(n) in sapzio O(n)

-in tempo O(n) in spazio O(1) NON RISOLTO

Input: un array di interi il cui primo elemento denota la dimensione es: 8 3 4 9 2 7 1 8 6

Output: l'elemento mancante es: 5 \*/

#include<stdio.h>

#include<stdlib.h>

//soluzione 1

//essendo tempo quadratico questo permette l'ordinamento con l'Insertion Sort, oppure con Selection Sort, non possiamo qui usare il Counting Sort che seppur sia lineare in tempo, ha pero' costo in spazio non costante

int \*InsertionSort(int a[], int dim){

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

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

k=a[j];

i=j-1;

while(i>=0 && a[i]>k){

a[i+1]=a[i];

i--;

}

a[i+1]=k;

}

return a;

}

void swap(int \*a, int \*b){

int temp=\*a;

\*a=\*b;

\*b=temp;

}

int \*SelectionSort(int a[], int dim){

int i=0, j=0, minimo=0;

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

minimo=i; //all'INIZIO il minimo e' la posizione del PRIMO elemento, questo per effettuare lo swap

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

if(a[j]<a[minimo])

minimo=j;

}

swap(&a[i], &a[minimo]);

}

return a;

}

//soluzione 2

//uso il MergeSort perche' ha i costi su detti

void Merge(int a[], int sx, int cx, int dx){

int i=sx, j=cx+1, k=0, b[999];

while(i<=cx && j<=dx){

if(a[i]<=a[j]){

b[k]=a[i];

i++;

}else{

b[k]=a[j];

j++;

}

k++;

}

while(i<=cx){

b[k]=a[i];

i++;

k++;

}

while(j<=dx){

b[k]=a[j];

j++;

k++;

}

for(k=sx; k<=dx; k++)

a[k]=b[k-sx];

}

int \*MergeSort(int a[], int sx, int dx){

int cx=0;

if(sx<dx){

cx=(sx+dx)/2;

MergeSort(a, sx, cx);

MergeSort(a, cx+1, dx);

Merge(a, sx, cx, dx);

}

return a;

}

//soluzione 3

//uso CountingSort

int \*CountingSort(int a[], int dim){

int i, max=a[0], k=0;

for(i=0; i<dim; i++){ //ricerca del massimo

if(a[i]>max)

max=a[i];

}

k=max+1;

int \*c=(int \*)calloc(k,sizeof(int)); //gli elementi di c sono cosi' tutti inizializzati a 0

int \*b=(int \*)malloc((dim+1)\*sizeof(int));

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

c[a[i]]++;

}

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

c[i]=c[i]+c[i-1];

}

for(i=dim-1; i>=0; i--){

b[c[a[i]]]=a[i];

c[a[i]]--;

}

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

a[i]=b[i+1];

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

}

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

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

//}

return a;

}

void cercaelemento(int a[], int dim){

//a=InsertionSort(a,dim);

//a=SelectionSort(a,dim);

//a=MergeSort(a,0,dim-1);

a=CountingSort(a,dim);

int i=0, trovato=0, risultato=0;

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

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

//i=0;

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

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

risultato=i+1;

trovato=1;

}

else if(a[i]==i+1)

i++;

}

if(trovato==0)

risultato=dim+1;

else if(trovato==1)

risultato=i+1;

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

}

int main(){

int n, i;

scanf("%d", &n);

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

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

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

}

cercaelemento(a,n);

free(a);

return 0;

}

**Programma 21**

/\*ARRAY DI STRINGHE: Scrivere un programma che legga da tastiera un array di stringhe. La prima riga dell'input contiene la dimensione n dell'array, le righe successive contengono gli elementi dell'array, una stringa per riga, ognuna di lunghezza massima 100 caratteri. \*/

#include<stdio.h>

#include<stdlib.h>

int main(){

int i, n;

scanf("%d", &n);

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

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

a[i]=(char \*)malloc(101\*sizeof(char));

scanf("%s", a[i]);

}

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

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

printf("\n");

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

free(a[i]); //LIBERARE OGNI SINGOLA STRINGA

free(a); //LIBERARE L'ARRAY DI STRINGHE

return 0;

}

**Programma 22**

//matrice

#include<stdio.h>

#include<stdlib.h>

int main(){

int N, M, i, j, \*\*A;

printf("Immettere numero colonne della matrice ");

scanf("%d",&N);

A=malloc(N\*sizeof(int\*));

printf("Immettere numero righe della matrice ");

scanf("%d",&M);

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

A[i]=malloc(M\*sizeof(int));

printf("Completare colonna %d dell'array \n", i+1);

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

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

}

}

printf("La matrice e'\n");

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

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

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

}

printf("\n");

}

return 0;

}

**Programma 23**

/\*Scrivere un programma che presi i dati da input di una matrice ne stampa il contenuto di essa, il massimo, il minimo.\*/

#include<stdio.h>

#include<stdlib.h>

int main(){

int N, M, i, j, \*\*A;

printf("Immettere numero colonne della matrice ");

scanf("%d",&N);

A=malloc(N\*sizeof(int\*));

printf("Immettere numero righe della matrice ");

scanf("%d",&M);

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

A[i]=malloc(M\*sizeof(int));

printf("Completare colonna %d dell'array \n", i+1);

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

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

}

}

printf("La matrice e'\n");

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

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

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

}

printf("\n");

}

int max=A[0][0], min=A[0][0];

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

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

if(A[i][j]>max)

max=A[i][j];

else if(A[i][j]<min)

min=A[i][j];

}

}

printf("Il massimo e': %d\n",max);

printf("Il minimo e': %d\n",min);

return 0;

}

**Programma 24**

/\*Questo programma permette di inserire i valori di un'array dinamico, quindi da tastiera si immette la dimensione e si completa con i rispettivi valori dell'array.\*/

#include<stdio.h>

#include<stdlib.h>

int main(){

int N,i;

int \*ptr;

printf("Immettere la dimensione dell'array ");

scanf("%d",&N);

ptr=malloc(N\*sizeof(int));

if(ptr!=NULL){

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

printf("Immettere il valore della posizione %d nell'array ",i+1);

scanf("%d",(ptr+i));

}

printf("Hai inserito i valori: ");

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

printf("%d ", ptr[i]);

}

printf("\n");

}

return 0;

}

**Programma 25**

//il programma stampa su schermo usando argomenti nel main

#include<stdio.h>

//il terzo argomento e' un array di stringhe che contiene i valori delle variabili d'ambiente.

int main(int argc, char \*argv[], char \*env[]){

argc=6;

int i;

argv[0]="Nel mio programma\0";

argv[1]="la stringa\0";

argv[2]="e' passata\0";

argv[3]="al\0";

argv[4]="programma\0";

argv[5]="\0";

printf("Ho ricevuto %d argomenti\n",argc-1);

printf("Questi argomenti sono: \n");

for(i=0; i<=argc-1; i++)

printf("%s\n",argv[i]);

printf("Questi argomenti sono al contrario: ");

for(i=5; i>=0; i--)

printf("%s\n",argv[i]);

//stampa a video la lista delle variabili ambiente

//for(i=0; env[i]!=NULL; i++)

//printf("%s\n",env[i]);

return 0;

}

//oppure

#include<stdio.h>

int main(int argc, char \*argv[]){

argc=1;

int i, j;

argv[0]="Ornela\0";

for(i=0; i<=argc-1; i++)

printf("%s\n",argv[i]);

return 0;

}

**Programma 26**

//Il Crivello di Erastotene trova tutti i numeri primi in un intervallo da 0 fino a un massimo

#include<stdio.h>

#define MAXVALORE 101

int main(){

int tabella[MAXVALORE],i,j; //devo usare due indici

for(i=2; i<MAXVALORE; i++)

tabella[i]=1; //assegno 1 a tutti i valori da 2 in poi

for(i=2; i<MAXVALORE; i++){

if(tabella[i]==1){

printf("%d\n",i); //cosi li stampo uno sotto l'altro

for(j=2; i\*j<MAXVALORE; j++) //se il secondo indice e' multiplo del primo ritorna 0

tabella[i\*j]=0;

}

}

printf("\n");

return 1; //cosi mi restituisce solo i numeri che non sono multipli di una altro numero

}

//oppure

/\*Questo programma stampa 0 se n e' primo altrimenti 1, (esiste un numero i appartenente a [2,n/1] tale che n e' divisibile per i).A priori e' impossibile sapere quanti valori di i saranno testati quindi si usa while\*/

#include<stdio.h>

int main(){

int n,i;

scanf("%d",&n);

i=2;

while(n%2 && i<n){

i++;

}

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

return 0;

}

//oppure

/\*Questo programma legge 10 interi e per ognuno stampa 0 se n e' primo altrimenti 1.\*/

#include<stdio.h>

int main(){

int n,i,j;

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

scanf("%d",&n);

i=2;

while(n%i && i<n){

i++;

}

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

}

return 0;

}

**Programma 27**

/\*programma che legge l'eta da tastiera e stabilisce se sei minorenne oppure maggiorenne\*/

#include<stdio.h>

int main(){

int eta;

printf("Inserire la tua eta in anni ");

scanf("%d",&eta);

if(eta>=18)

printf("Birra e alcool per te!!\n");

else

printf("Sei ancora minorenne.\n");

return 0;

}

**Programma 28**

/\*Questo programma stampa su schermo la media dell'array eta costituito da tali valori.\*/

#include<stdio.h>

int main(){

int eta[]={23, 24, 17, 27, 25, 24, 24}, n=7, i, somma=0;

double media=0;

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

somma=somma+eta[i];

media=(double)somma/n;

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

return 0;

}

**Programma 29**

/\*l'esercizio non termina oppure non esegue niente perche' la guardia del for e' sempre verificata, mentre per uscire dal ciclo for la guardia deve diventare falsa\*/

#include<stdio.h>

int main(){

int i;

for(i=0; i%2==1; i+=2)

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

return 0;

}

//oppure

#include<stdio.h>

int main(){

int i;

for(i=0; i<=0; i--)

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

return 0;

}

**Programma 30**

#include<stdio.h>

int main(){

int n;

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

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

return 0;

}

**Programma 31**

/\*Questo programma legge i valori da tastiera dell'array e li ristampa nell'ordine inverso a come sono stati letti.\*/

#include<stdio.h>

int main(){

int arr[7], i;

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

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

for(i=6; i>=0; i--)

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

return 0;

}

**Programma 32**

//il programma stampa i numeri pari fino a 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 33**

/\*Questo programma legge i numeri interi positivi da tastiera finche' non ne trova uno negativo. A priori e' impossibile sapere quanti valori saranno letti, ma sappiamo che ne va letto almeno uno, quindi si usa do-while.\*/

#include<stdio.h>

int main(){

int n;

do{

scanf("%d",&n);

}while(n>=0);

return 0;

}

**Programma 34**

/\*Questo programma legge interi da terminale finche' la somma non supera 100, quindi stampa la somma. A priori e' impossibile sapere quanti numeri saranno letti, quindi si usa while\*/

#include<stdio.h>

int main(){

int somma=0, n;

while(somma<100){

scanf("%d",&n);

somma=somma+n;

}

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

return 0;

}

**Programma 35**

//questo programma stampa la tabellina del 3 fino a 10

#include<stdio.h>

int main(){

int i;

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

printf("%d\n",i\*3);

}

return 0;

}

**Programma 36**

**//FUNZIONE COMPARE DEL QSORT**

//ordinare un array di interi in modo non decrescente con la funzione qsort

#include<stdio.h>

#include<stdlib.h>

int confrontaInt(const void \*a, const void \*b){

return (\*(int \*)a - \*(int \*)b);

}

int main(){

int values[]={10, 20, 1, 7, 24, 9}, n=6, i;

qsort(values, n, sizeof(int), confrontaInt);

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

printf("%d ",values[i]);

printf("\n");

return 0;

}

//oppure

//ordinare un array di stringhe in ordine lessicografico

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

int compare(const void \*a, const void \*b){

char \*\*a1=(char \*\*)a;

char \*\*a2=(char \*\*)b;

return strcmp(\*a1, \*a2);

}

int main(){

int n, i;

scanf("%d",&n);

char \*\*values=(char \*\*)malloc(n\*sizeof(char \*));

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

values[i]=(char \*)malloc(101\*sizeof(char));

scanf("%s", values[i]);

}

qsort(values, n, sizeof(char \*), compare);

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

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

printf("\n");

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

free(values[i]);

free(values);

return 0;

}

**Programma 37**

/\*ELEMENTO MAGGIORITARIO: Un array A contiene n interi, uno di essi e' detto elemento maggioritario se occorre in A almeno (n/2+ 1) volte. Si vuole un algoritmo che identifichi l’elemento maggioritario, se presente, in tempo O(n) utilizzando O(1) spazio aggiuntivo. L’algoritmo deve stampare N se non e' presente alcun elemento maggioritario.

-Caso particolare: si assuma che tutti gli elementi, ad esclusione eventualmente del maggioritario, occorrano una sola volta in A.

-Caso generale: gli elementi possono avere un numero arbitrario di occorrenze.

Input 5 1 10 22 11 2

Output N

Input 5 22 10 22 11 22

Output 22 \*/

#include<stdio.h>

#include<stdlib.h>

int \*CountingSort(int a[], int dim){

int i, max=a[0], k=0;

for(i=0; i<dim; i++){ //ricerca del massimo

if(a[i]>max)

max=a[i];

}

k=max+1;

int \*c=(int \*)calloc(k,sizeof(int)); //gli elementi di c sono cosi' tutti inizializzati a 0

int \*b=(int \*)malloc((dim+1)\*sizeof(int));

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

c[a[i]]++;

}

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

c[i]=c[i]+c[i-1];

}

for(i=dim-1; i>=0; i--){

b[c[a[i]]]=a[i];

c[a[i]]--;

}

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

a[i]=b[i+1];

}

return a;

}

void cercaelemento(int a[], int dim){

a=CountingSort(a,dim);

int i=0, cx=0, contaprimo=1, contaultimo=0, trovato=0, risultato=0;

cx=dim/2;

i=cx;

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

if(a[i]!=a[i-1])

trovato=1;

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

contaprimo++;

i--;

}

}

i=cx;

trovato=0;

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

if(a[i]!=a[i+1])

trovato=1;

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

contaultimo++;

i++;

}

}

risultato=contaprimo+contaultimo;

if(risultato==cx+1)

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

else if(risultato!=cx+1)

printf("N \n");

}

int main(){

int n, i;

scanf("%d", &n);

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

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

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

}

cercaelemento(a,n);

return 0;

}

Programma 38

Programma 1

//Questo programma stampa le cifre da 0 fino a 9

#include<stdio.h>

#define CIFRE 9

int main()

{

int i;

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

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

return 0;

}

Programma 2

#include<stdio.h>

#define PRODOTTO(x,y) ((x)\*(y)) //se fosse stato (x\*y), restituiva un risultato sbagliato

int main()

{

int i,j=2;

i=PRODOTTO(j+1,3);

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

int b,c=0;

b=(c++,c\*2);

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

int d,e=0;

d=e++,e\*2; //qui l'assegnamento ha precedenza rispetto alla virgola

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

return 0;

}