Puntatori

Informatica A - 24/10/2024

Esempio - Calcolo del cubo

```c
#include <stdio.h>

int cubeByValue(int n); // prototipo

int main(void) {
   int number = 5; // inizializza number

   printf("The original value of number is %d", number);
   number = cubeByValue(number); // passa number per valore a cubeByValue
   printf("\nThe new value of number is %d\n", number);
}

// calcola e restituisci il cubo di un argomento intero
int cubeByValue(int n) {
   return n * n * n; // restituisci il cubo di n
}
```

```
The original value of number is 5
The new value of number is 125
```

```c
#include <stdio.h>

void cubeByReference( int *nPtr); // prototipo di funzione

int main(void) {
   int number = 5; // inizializza number

   printf("The original value of number is %d", number);
   cubeByReference(&number); // passa l'indirizzo di number a cubeByReference
   printf("\nThe new value of number is %d\n", number);
}

// eleva al cubo *nPtr; di fatto modifica number in main
void cubeByReference( int *nPtr ) {
   *nPtr = *nPtr * *nPtr * *nPtr; // calcola il cubo di *nPtr
}
```

```
The original value of number is 5
The new value of number is 125
```

Esercizio 1

Per ciascuna delle seguenti richieste, scrivi una singola linea di codice che faccia quanto indicato.
(Assumi che value1 e value2 siano due long int e che value1 sia stato inizializzato a 200000)

a)	Definisci la variabile lPtr come puntatore di un oggetto di tipo long	➔	long *lPtr
b)	Assegna l'indirizzo di value1 al puntaote lPtr	➔	lPtr = &value1
c)	Print il valore dell'oggetto cui lPtr punta	➔	printf("%ld\n", *lPtr)
d)	Assegna il valore dell'oggetto cui lPtr punta a value2	➔	value2 = *lPtr
e)	Print il valore di value2	➔	printf("%ld\n", value2)
f)	Print l'indirizzo di value1	➔	printf("%p\n", &value1)
g)	Print l'indirizzo racchiuso in lPtr. E' lo stesso valore di value 1?	➔	printf("%p\n", lPtr). // si

Esercizio 2

Ci sono errori nel seguente frammento di codice? Cosa fa?

```c
#include <stdio.h>
#define SIZE 80

void mystery1(char *s1, const char *s2); // prototype

int main(void) {
   char string1[SIZE]; // create char array
   char string2[SIZE]; // create char array

   puts("Enter two strings: ");
   scanf("%39s%39s" , string1, string2);
   mystery1(string1, string2);
   printf("%s", string1);
}

// What does this function do?
void mystery1(char *s1, const char *s2) {
   while (*s1 != ) {
      ++s1;
   }

   for (; *s1 = *s2; ++s1, ++s2) {
      ; // empty statement
   }
}
```

Mancano i check !='\0' in while e for

Concatena due stringhe

Esercizio 3

Ci sono errori nel seguente frammento di codice? Cosa fa?

```c
#include <stdio.h>
#define SIZE 80

size_t mystery2(const char *s); // prototype

int main(void) {
   char string[SIZE]; // create char array

   puts("Enter a string: ");
   scanf("%79s", string);
   printf("%d\n", mystery2(string));
}

// What does this function do?
size_t mystery2(const char *s) {
   size_t x;

   // loop through string
   for (x = 0; *s != ; ++s) {
      ++x;
   }

   return x;
}
```

Mancano i check !='\0' in for e %d in print sarebbe %zu

size_t tipo di carattere usato per dimensioni/bit (size_of)

Controlla lunghezza della stringa

Esercizio 4

Ci sono errori nel seguente frammento di codice? Cosa fa?

```c
#include <stdio.h>
#define SIZE 80

int mystery3(const char *s1, const char *s2); // prototype

int main(void) {
   char string1[SIZE]; // create char array
   char string2[SIZE]; // create char array

   puts("Enter two strings: ");
   scanf("%79s%79s", string1 , string2);
   printf("The result is %d\n", mystery3(string1, string2));
}
int mystery3(const char *s1, const char *s2) {
   int result = 1;

   for (; *s1 !=  && *s2 != ; ++s1, ++s2) {
      if (*s1 != *s2) {
         result = 0;
      }
   }

   return result;
}
```

Mancano i check !='\0' in for

restistuisce 0 o 1 se le due stringhe fossero uguali

si poteva fare direttamente il return 0 quando diversa

Esercizio 5

Scrivere un programma C che definita una matrice di array, per ciascuno ne mostri a video la media.

```c
#include <stdio.h>

void calcAverages(int **array, int *lenghts, int size);

int main(){
    int a1[1] = {1};
    int a2[2] = {1,2};
    int a3[3] = {1,2,3};
    int a4[4] = {1,2,3,4};
    int a5[5] = {1,2,3,4,5};

    int *ptrArray[5] = {a1, a2, a3, a4, a5};
    int lenghts[5] = {1,2,3,4,5};

    calcAverages(ptrArray, lenghts, 5);
}

void calcAverages(int **array, int *lenghts, int size){
    for (int i=0; i<size; i++){
        float somma = 0;
        for (int j=0; j<lenghts[i]; j++){
            somma = somma + array[i][j];
        }
        printf("Media array %d: %f\n",i, somma/lenghts[i]);
    }
}
```

Esercizio 6

Scrivere una funzione per la risoluzione di equazioni di secondo grado della forma $ax2 + bx + c = 0$ in base alle seguenti indicazioni. La funzione riceve tre parametri (di tipo double) passati per valore che rappresentano i tre coefficienti a, b e c e due parametri x1 e x2 (anch'essi di tipo double) passati per riferimento. La funzione ritorna un valore intero. Se l'equazione ha soluzioni reali la funzione ritorna 1 e nei parametri x1 e x2 vengono scritte le soluzioni dell'equazione. Se l'equazione non ha soluzioni reali la funzione ritorna 0 e le variabili x1 e x2 non vengono scritte.

Scrivere poi un programma che fa inserire all'utente i coefficienti di un'equazione di secondo grado e stampa le sue soluzioni se queste sono reali e un opportuno messaggio in caso contrario.

```c
#include <stdio.h>
#include <math.h>

int soluzioni(double a, double b, double c, double * x1,double * x2);

int main(void){
    printf("Inserisci i tre coefficienti dell'equazione\n");
    double a, b, c;
    scanf("%lf%lf%lf",&a, &b, &c);
    double x1, x2;
    if(soluzioni(a, b, c, &x1, &x2)){
        printf("Le soluzioni reali dell'equazione sono: %g e %g\n",    x1, x2);}
    else{
        printf("L'equazione non ha soluzioni reali\n");}}

int soluzioni(double a, double b, double c, double * x1,double * x2){
double delta=b*b-4*a*c;
int soluzioniReali=0;
if(delta>0){
    *x1=(-b+sqrt(delta))/(2*a);
    *x2=(-b-sqrt(delta))/(2*a);
    soluzioniReali=1;}
return soluzioniReali;}
```

Esercizio 7

Modificare la funzione precedente in modo che calcoli le soluzioni complesse dell'equazione. Più precisamente, la funzione avrà tre parametri (di tipo double) passati per valore che rappresentano i coefficienti a, b e c dell'equazione e quattro parametri (di tipo double) passati per riferimento. Tali parametri rappresentano la parte reale e la parte immaginaria di ciascuna delle due soluzioni.

```c
#include <stdio.h>
#include <math.h>

void soluzioni(double a, double b, double c, double * re1, double * im1, double * re2, double * im2);

int main(void){
    printf("Inserisci i tre coefficienti dell'equazione\n");
    double a, b, c;
    scanf("%lf%lf%lf",&a, &b, &c);
    double re1, im1, re2, im2;
    soluzioni(a, b, c, &re1, &im1, &re2, &im2);
    printf("Le soluzioni dell'equazione sono: %g +i %g e %g + i %g\n", re1, im1, re2, im2);}

void soluzioni(double a, double b, double c, double * re1, double * im1, double * re2, double * im2){
    double delta=b*b-4*a*c;
    if(delta>0){*re1=(-b+sqrt(delta))/(2*a);
        *re2=(-b-sqrt(delta))/(2*a);
        *im1=0;
        *im2=0;
    }else{
        *re1=-b/(2*a);
        *im1=sqrt(-delta)/(2*a);
        *re2=*re1;
        *im2=-sqrt(-delta)/(2*a);}}
```

Esercizio 8

Scrivere una funzione che calcola l'equazione di una retta nel piano passante per due punti (x1,y1) e (x2, y2). La funzione riceve come parametri i quattro valori double x1, y1, x2 e y2 (passati per valore) e due ulteriori parametri *m* e *q* (passati per riferimento) in cui la funzione deve scrivere il coefficiente angolare *m* e il termine noto *q* dell' equazione della retta passante per i due punti dati. Qualora la retta sia verticale (cioè avente equazione y=c) il valore di *m* deve essere impostato a INFINITY (tale costante è definita nella libreria math.h) e *q* deve essere uguale a c.

Scrivere poi un programma che chiede all'utente di inserire le coordinate di due punti nel piano e stampa l'equazione della retta passante per i due punti dati.

```c
#include <stdio.h>
#include <math.h>

void retta(double x1, double y1, double x2, double y2, double * m, double* q);

int main(void){
    printf("Inserisci le coordinate di un punto nel piano\n");
    double x1, y1;
    scanf("%lf%lf",&x1, &y1);
    printf("Inserisci le coordinate di un secondo punto\n");
    double x2, y2;
    scanf("%lf%lf",&x2, &y2);
    double m, q;
    retta(x1, y1, x2, y2, &m, &q);
    printf("L'equazione della retta per i due punti dati e'\n");
    if(m==INFINITY){
        printf("x=%g\n",q);}
    else if(m==0){
        printf("y=%g\n",q);}
    else {
        printf("y=%g x + %g\n", m, q);}}

void retta(double x1, double y1, double x2, double y2, double * m, double* q){
    if(x1==x2){ //retta verticale
        *m=INFINITY;
        *q=x1;
    }else{
        *m=(y2-y1)/(x2-x1);
        *q=-(*m)*x2+y2;}}
```

Esercizio 9

Scrivere una funzione che prende come parametro un array di interi e restituisce il numero di elementi che sono maggiori del loro successore. Si operi sugli elementi dell'array tramite l'aritmetica dei puntatori invece che tramite gli indici.

```c
#include <stdio.h>
#include <math.h>

int contaInversioni(int* aPtr, int dim);

int main(void){
    printf("Quanti elementi vuoi inserire?\n");
    int n;
    scanf("%d",&n);
    int a[n];

    for(int i=0;i<n;i++){
        printf("Inserisci l'elemnto in posizione %d ", i);
        scanf("%d",&a[i]);}

    int c=contaInversioni(a, n);
    printf("Il numero di elementi maggiore del proprio successore e':%d\n", c);}

int contaInversioni(int* aPtr, int dim){
    int count=0;
    for(int i=0;i<dim-1;i++)
    if(*(aPtr+i)>*(aPtr+i+1)){
        count++;}
    return count;}
```

Esercizio 10 (1)

Scrivere un programma C che definita una matrice 4x4, estragga e stampi a video la sottomatrice 2x2 la cui somma dei valori è massima.

```c
int main(){
    int mat[SIZE][SIZE] = {
        {1, 2, -1, 2},
        {-3, 21, 5, 1},
        {1, -2, 1, 3},
        {1, 4, 5, 6},
    };
    int fCol, fRow;

    printf("Matrice origianle:\n");
    stampaMatrice(mat);

    trovaSottomatrice(mat);

    trovaSottomatrice2(mat, &fRow, &fCol);
    printf("La sottomatrice con somma massima è: \n");
    for (int i=0; i<2; i++){
        for (int j=0; j<2; j++){
            printf("%2d ",mat[fRow+i][fCol+j]);}
        printf("\n");}}

void stampaMatrice(int mat[SIZE][SIZE]){
    for(int i=0; i<SIZE; i++){
        for (int j=0; j<SIZE; j++){
            printf("%2d ",mat[i][j]);}
        printf("\n");}}
```

Esercizio 10 (2)

Scrivere un programma C che definita una matrice 4x4, estragga e stampi a video la sottomatrice 2x2 la cui somma dei valori è massima.

```c
void trovaSottomatrice(int mat[SIZE][SIZE]){
    int maxSum = 0;
    int finalCol = 0, finalRow = 0;
    for (int i=0; i<SIZE-1; i++){
        for (int j=0; j<SIZE-1; j++){
            int sum = mat[i][j]+mat[i+1][j]+mat[i][j+1]+mat[i+1][j+1];
            if (sum>maxSum){
                maxSum = sum;
                finalRow = i;
                finalCol = j;}}}
    printf("La sottomatrice con somma massima è: \n");
    for (int i=0; i<2; i++){
        for (int j=0; j<2; j++){
            printf("%2d ",mat[finalRow+i][finalCol+j]);}
        printf("\n");}}

void trovaSottomatrice2(int mat[SIZE][SIZE], int *finalRow, int *finalCol){
    int maxSum = 0;
    *finalCol = 0;
    *finalRow = 0;
    for (int i=0; i<SIZE-1; i++){
        for (int j=0; j<SIZE-1; j++){
            int sum = mat[i][j]+mat[i+1][j]+mat[i][j+1]+mat[i+1][j+1];
            if (sum>maxSum){
                maxSum = sum;
                *finalRow = i;
                *finalCol = j;}}}}
```

Esercizio 11

Si scriva una funzione ricorsiva che prende in input due array della stessa dimensione A e B, i cui elementi sono solo numeri 0 ed 1 e calcola la loro distanza di Manhattan.

La distanza di Manhattan è definita come il numero di elementi diversi nei due array.

(Ad esempio, gli array: 1,0,1,1,0,1 e 0,1,1,1,0,1 Hanno distanza di Manhattan uguale a 2)

```c
#define LEN 6
#include <stdio.h>

int manhattan_iter(int *a, int *b, int lunghezza){
    int i = 0;
    int differenze = 0;
    for(i=0;i<lunghezza;i++){
        if(a[i] != b[i]){
            differenze++;}}
    return differenze;}

int manhattan(int *a, int *b, int lunghezza){
    if(lunghezza == 0){
        return 0;
    }else{
        if(*a == *b){
            return 0 + manhattan(a+1,b+1,lunghezza-1);
        }else{
            return 1 + manhattan(a+1,b+1,lunghezza-1);}}}

int manhattan_tail(int *a, int *b, int lunghezza, int differenze){
    if(lunghezza == 0){
        return differenze;
    }else{
        if(*a == *b){
            return manhattan_tail(a+1,b+1,lunghezza-1,differenze);
        }else{
            return manhattan_tail(a+1,b+1,lunghezza-1,differenze+1);}}}
int manhattan_wrapper(int *a, int *b, int lunghezza){
    return manhattan_tail(a,b,lunghezza,0);}

int main(){
    int a[LEN] = {1,0,1,1,0,1};
    int b[LEN] = {0,1,1,1,0,1};
    printf("%d", manhattan_wrapper(a,b,LEN));}
```

Esercizio 12 (1)

Scrivere un programma C che riceva in input un numero n di stringhe e le renda tutte maiuscole.

```c
#include <stdio.h>
#include <stdlib.h> //per la malloc

char** creaArrayDiStringhe(int numStringhe, int lunghezze[]);
void tuttoMaiuscolo(char parola[]);

int main(int argc, const char * argv[]) {
    char **stringhe = NULL;
    int numStringhe;
    int *lunghezze = NULL;
    int i;
    printf("Quante stringhe vuoi inserire?\n");
    scanf("%d",&numStringhe);
    lunghezze = (int*)malloc(sizeof(int)*numStringhe);
    printf("Quanto è lunga ogni stringa?\n");
    for (i=0; i<numStringhe; i++) {
        printf("Lunghezza stringa n.%d = ",i);
        scanf("%d",&lunghezze[i]);}
    stringhe = creaArrayDiStringhe(numStringhe, lunghezze);
    for (i=0; i<numStringhe; i++){
        printf("Inserisci la stringa %d di lunghezza %d\n",i,lunghezze[i]);
        scanf(" %s",stringhe[i]);}
    for (i=0; i<numStringhe; i++){
        tuttoMaiuscolo(stringhe[i]);}
    for (i=0; i<numStringhe; i++){
        printf("%s\n",stringhe[i]);}
    for (i=0; i<numStringhe; i++)
        free(stringhe[i]);
    free(stringhe);
    free(lunghezze);
    return 0;}
```

Esercizio 12 (2)

Scrivere un programma C che riceva in input un numero n di stringhe e le renda tutte maiuscole.

```c
char** creaArrayDiStringhe(int numStringhe, int lunghezze[]){
    char **arrayDiStringhe = NULL;
    int i,j;
    arrayDiStringhe = (char**)malloc(sizeof(char*)*numStringhe);
    if (arrayDiStringhe == NULL){
        printf("Error di allocazione memoria\n");
        return NULL;}
    for (i=0; i<numStringhe; i++){
        arrayDiStringhe[i] = (char*)malloc(sizeof(char)*(lunghezze[i]+1));

        if (arrayDiStringhe[i]==NULL){
            printf("Errore di allocazione memoria\n");
            for (j=0; j<i; j++){
                free(arrayDiStringhe[j]);}
            free(arrayDiStringhe);
            return NULL;}}
    return arrayDiStringhe;}

void tuttoMaiuscolo(char parola[]){
    int i;
    for (i=0; parola[i]!='\0'; i++){
        if ((parola[i]>='a') && (parola[i]<='z')){
            parola[i] = parola[i] -'a' + 'A';
        }
    }
}
```