Programmazione I

Il Linguaggio C

Esercizi

Daniel Riccio

Università di Napoli, Federico II

17 dicembre 2021

Risultato della competizione

Posizione	Matricola	Esito	Tempo
1	N86004029	Correctness Test passed	0.002296
2	N86004428	Correctness Test passed	0.002800
3	N86004052	Correctness Test passed	0.002996
4	N86004057	Correctness Test passed	0.003262
5	N86004117	Correctness Test passed	0.010180
6	N86004077	Correctness Test passed	0.663828
7	N86004389	Correctness Test passed	0.676110
8	N86004283	Correctness Test passed	1.171358

Quesito 1-a

Qs. 1 (a) – (5 punti): si risponda alle seguenti domande riportando su un foglio bianco il numero della domanda e al lato di ciascuna numero una V se si considera la risposta vera o una F se si ritiene che la risposta sia falsa.

N.	Domanda	V/F
1	Una variabile di tipo char ammette valori nell'intervallo [0,255]	F
2	La funzione strncat(a,b) confronta i primi n caratteri delle due stringhe a e b	F
3	if(N=0){…} non esegue mai le istruzioni nelle parentesi graffe	V
4	Un record è una collezione eterogenea di variabili	V
5	La definizione di un record non può contenere altri record annidati	F
6	Nell'aritmetica dei puntatori il prodotto e la divisione sono operazioni non ammesse	F
7	Una stringa è una sequenza di caratteri terminata da un '\0'	V
8	Il nodo sentinella termina una lista doppiamente concatenata	F
9	I puntatori hanno sempre la dimensione di una parola macchina	V
10	Uno stack è una struttura ricorsiva di tipo FIFO	F
11	Il ciclo do{…}while(1); è un ciclo infinito	V
12	Il Bubble Sort ha complessità O(n) nel caso medio e $O(n^2)$ nel caso peggiore	F
13	Per i dati signed, shift logico e aritmetico non è equivalente	V
14	Il tipo long int occupa 8 byte su una macchina a 32 bit	V
15	La ricorsione con stack esplicito è meno efficiente di quella basata sullo stack di sistema	F
16	La ricerca binaria non può essere applicata su una stringa di caratteri anche se ordinata	F
17	La sequenza di fibonacci si calcola come f(n)=f(n-1) – f(n-2), con f(1)=1 e f(0)=1	F
18	La funzione malloc() restituisce un puntatore a intero	F
19	L'inserimento di un nodo in una lista single-linked richiede due puntatori nella ricerca della posizione	V
20	Il nome di un vettore è un puntatore costante	V

Quesito 1-b

Qs. 1 (b) – (3 punti): sia dato il vettore V={81, 1, 21, 11, 25, 40}. Si mostri lo stato del vettore ad ogni passo dell'algoritmo Insertion Sort.

Iterazione	V[0]	V[1]	V[2]	V[3]	V[4]	V[5]
1	81	1	21	11	25	40
2	1	81	21	11	25	40
3	1	21	81	11	25	40
4	1	11	21	81	25	40
5	1	11	21	25	81	40
6	1	11	21	25	40	81
7						
8						
9						
10						

Quesito 2

Qs. 2 – (5 punti): si indichi l'output del seguente programma.

```
#include <stdio.h>

int V[5]={1,2,3};
int main(){
  int *p=V;
  int i;

  for(i=0; *p>0; p++);

  p[i] = *(p-1) + *(p-2);

  printf("%ld %d %d\n", sizeof(p), *p, V[i]);

  return 0;
}
```

Output:

4 5 1

Quesito 3

Qs. 3 – (5 punti): si indichino gli errori, qualora presenti, nelle seguenti istruzioni.

`#include “stdio.h”;`	punto e virgola
`int main()`	Manca la parentesi graffa
`int c=32;`	corretto
`char *pc=&c;`	Tipi incompatibili
`pc=pc*1;`	corretto
`for((i=0); (i<5), (i++))`	virgola
`c = *pc + i;`	corretto
`printf(%d\n, *pc);`	Mancano i doppi apici
`return &i;`	int * non è un intero
	Manca la parentesi graffa

Quesito 4

Qs. 5 – (12 punti): Si implementi in linguaggio C il seguente programma

Scrivere un programma C che:
1) Prende da linea di comando una 10 numeri e li inserisce in un vettore V. Inoltre, chiede in input un numero x mediante una scanf()
2) Implementa la funzione ricorsiva int Cerca_Valore (int V[], int n, int x) che restituisce 1 se x compare nel vettore e 0 altrimenti.

Input e Output	
Input: 10 1 23 5 8 12 7 11 33 9 **Output:** **Inserisci il carattere da cercare: 8** **Il carattere 8 compare nel vettore**	

Soluzione

```
int cerca_valore(int V[], int n, int x) {
        if (n == 0)
                return (V[n] == x);
        else {
                if (V[n] == x)
                        return 1 + cerca_valore(V, n - 1, x);
                else
                        return cerca_valore(V, n - 1, x);
        }
}
```

Soluzione

```
int main(int argc, char *argv[])
{
	int i;
	int x = 0;
	int posizione = 0;
	int V[10];

	for (i = 0; i < 10; i++)
		V[i] = atoi(argv[i]);

	printf("Inserisci il valore da cercare: ");
	scanf("%d", &x);

	posizione = cerca_valore(V, 10, x);

	if (posizione > 0)
		printf("Il valore %d compare nel vettore\n", x);
	else
		printf("Il valore %d non compare nel vettore\n", x);

	return 0;
}
```

Traccia

Traccia 012

Scrivere un programma C che:
1) Prende dalla linea di comando un numero generico di interi (es.: 2 3 7 10)
2) Inserisce i valori in un vettore di interi allocato dinamicamente
3) Implementa la funzione ricorsiva int SommaPari(int *v, int n) che somma solo i valori pari all'interno dell'array

Input e Output

Input: 2 3 7 10
La somma degli elementi nel vettore è: 12

Soluzione

```
int SommaPari(int *A, int n) {

        if (n == 0)
                if ((A[0] % 2 == 0))
                        return A[0];
                else
                        return 0;
        else if (A[n] % 2 == 0)
                return Somma(A, n - 1) + A[n];
        else return Somma(A, n - 1);
}
```

Soluzione

```
int main(int argc, char *argv[])
{
        int *V;
        int i = 0;
        int somma = 0;

        V = (int *)malloc((argc - 1) * sizeof(int));

        for (i = 1; i < argc; i++)
                V[i - 1] = atoi(argv[i]);

        somma = SommaPari(V, argc - 2);

        printf("La somma degli elementi nel vettore è: %d", somma);

        free(V);

        return 0;
}
```

Traccia

Traccia 008

Scrivere un programma C che:
1) Definisce un nuovo tipo di variabile Polinomio come record (grado n, n+1 coefficienti float).
2) Scrivere una funzione float polydev(Polinomio p, Polinomio *pdev) che inserisce in pdev la derivata prima di p.

Input e Output

Inserisci il grado del polinomio: 3
Inserisci il coefficiente per x^0: 2
Inserisci il coefficiente per x^1: 1.8
Inserisci il coefficiente per x^2: 2.6
Inserisci il coefficiente per x^3: 4.5
Il polinomio di grado 3 è: 4.5x^3 +2.6x^2 +1.8x^1 +2
Il polinomio derivato è di grado 2 ed è: 13.5x^2 +5.2x^1 +1.8

Soluzione

```
#include <stdio.h>
#include <stdlib.h>

typedef struct Polinomio {
        int grado;
        float c[64];
}Polinomio;

void polydev(Polinomio p, Polinomio *pdev);
```

Soluzione

```
int main(int argc, char* argv[])
{
	int i = 0;
	Polinomio p, pdev;

	printf("Inserisci il grado del polinomio: ");
	scanf("%d", &(p.grado));

	for (i = 0; i < p.grado + 1; i++) {
		printf("Inserisci il coefficiente per x^%d:", i);
		scanf("%f", &(p.c[i]));
	}

	printf("Il polinomio di grado %d è: ", p.grado);
	for (i = p.grado; i > 0; i--)
		printf("%gx^%d +", p.c[i], i);
	printf("%g\n", p.c[0]);

	polydev(p, &pdev);
	printf("Il polinomio derivato è di grado %d ed è: ", pdev.grado);

	for (i = pdev.grado; i > 0; i--)
		printf("%gx^%d +", pdev.c[i], i);
	printf("%g\n", pdev.c[0]);
	return 0;
}
```

Soluzione

```
void polydev(Polinomio p, Polinomio *pdev) {

        int i = 0;

        pdev->grado = p.grado - 1;

        for (i = p.grado; i > 0; i--)
                pdev->c[i - 1] = i * p.c[i];

        return;
}
```