Esercitazione Capitolo 1:

Visibilità, funzioni e gestione della memoria

1. Ricorsione - C - HASH

Si scriva una funzione hash in C che calcola l'hash intero di una stringa in questo modo: fa la somma di tutti i caratteri moltiplicati per la loro posizione all'interno delle stringa (con la posizione iniziale = 1).

Al solito, scrivi tre versioni: una non ricorsiva, una ricorsiva senza tail recursion e una ricorsiva con tail recursion.

Specifica esattamente i parametri che passi alla procedura, il tipo di passaggio utilizzato e il loro significato.

Scrivi anche un main di esempio in cui chiami la funzione con stringa “ab” e assegna il risultato ad una variabile globale HASH_AB. Prova a chiamare tutte e tre le versioni della funzione.

Record di attivazione

Disegna il record di attivazione per tutte e tre le versioni fino alla massima estensione del record di attivazione. Nel caso di tail recursion, spiega nel codice quali ottimizzazioni hai adottato o potresti adottare.

Per ognuno dei seguenti esercizi scrivi tre versioni: una non ricorsiva, una ricorsiva senza tail recursion e una ricorsiva con tail recursion. Inoltre disegna il record di attivazione.

---

2. Somma dei numeri pari

Dato un array di interi calcolare la somma dei numeri pari

3. Palindroma

Data una stringa restituisce se palindroma (usa bool)

4. Divisibile per 8

un numero è divisibile per 8 se termina con tre zeri o se è divisibile per 8 (chiamata ricorsiva) il numero formato dalle sue ultime 3 cifre
(puoi usare %8 solo per controllare che sia divisibile un numero con 3 cifre)

5. Divisibile per 7

un numero con più di due cifre è divisibile per 7 se la differenza del numero ottenuto escludendo la cifra delle unità e il doppio della cifra delle unità è 0, 7 o è un numero divisibile per 7.
per es. 95676 è divisibile per 7 se lo è il numero 9567-2*6=9555; questo è divisibile per 7 se lo è il numero 955-2*5=945; questo è divisibile per 7 se lo è 94-2*5=84 che è divisibile per 7 dunque lo è anche il numero 95676
(non puoi usare %7 tranne che per numeri di max due cifre ma puoi usare %10)

TEMA ESAME. Ricorsione - C – funzione vocalizza

Si scriva una funzione vocalizza in C che prende una stringa come array di char (assumi per semplicità minuscoli) e restituisce una stringa uguale abbreviata ignorando tutte le consonanti. Ad esempio vocalizza(“angelo”)=“aeo”

Al solito, scrivi tre versioni: una non ricorsiva, una ricorsiva senza tail recursion e una ricorsiva con tail recursion. Specifica esattamente i parametri che passi alla procedura, il tipo di passaggio utilizzato e il loro significato con commenti prima della funzione.

Scrivi anche un main di esempio in cui chiami la funzione con “aio” e assegna il risultato ad una variabile globale RESULT. Prova a chiamare tutte e tre le versioni della funzione (e stampare il risultato per controllare che il tuo codice funziona).

Se necessario usa free dei puntatori.

Disegna il record di attivazione (su carta o su un file calc con estensione ods) per tutte e tre le versioni fino alla massima estensione del record di attivazione. Nel caso di tail recursion, spiega nel codice quali ottimizzazioni hai adottato o potresti adottare.

Per concatenare stringhe puoi usare strcat oppure sprintf

strcat

Description

The C library function char *strcat(char *dest, const char *src) appends the string pointed to by src to the end of the string pointed to by dest.

Declaration

char *strcat(char *dest, const char *src)

Parameters

dest -- This is pointer to the destination array, which should contain a C string, and should be large enough to contain the concatenated resulting string.

src -- This is the string to be appended. This should not overlap the destination.

Return Value

This function returns a pointer to the resulting string dest.

sprintf

Description

The C library function int sprintf(char *str, const char *format, ...) sends formatted output to a string pointed to, by str.

Declaration

int sprintf(char *str, const char *format, ...)

Parameters

str − This is the pointer to an array of char elements where the resulting C string is stored.

format − This is the String that contains the text to be written to buffer. It can optionally contain embedded format tags that are replaced by the values specified in subsequent additional arguments and formatted as requested. Format tags prototype: %[flags][width][.precision][length]specifier, as explained below −

Ad esempio:

```
sprintf(b, "%c%s", str[0], vocalizza_ric(str+1));
```

Passaggio di parametri

Passaggio puntatore a puntatore - int

Scrivi una funzione che prende una variabile intera nel main come puntatore a puntatore e lo incrementa di 1. Prova a stampare il valore prima e dopo la chiamata della funzione. Disegna il record di attivazione

Passaggio puntatore a puntatore – string

Scrivi una funzione che prende una stringa e modifica la prima lettera e mette “A”. Usa i puntatori a puntatori Stampa la stringa prima e dopo la chiamata. Disegna il record di attivazione.

Passaggio per riferimento

Alcune volte si usa il passaggio per riferimento per ottenere indietro un risultato.

Ad esempio al funzione:

```
void somma(int x, int y, int&z){
        // esegui la somma tra x e y e mettila in z
}
```

Può essere usata per ottenere il risultato della somma tra x e y. Prova a scrivere un main in cui fai la somma tra due numeri e leggi il risultato in una variabile ausiliaria passata come z a somma.