PUNTATORI A FUNZIONE
mercoledì 21 dicembre 2022 11:03

PUNTATORI A FUNZIONE

LA FUNZIONE definisce l'INDIRIZZO IN MEMORIA della PRIMA ISTRUZIONE;
QUINDI la FUNZIONE È CARATTERIZZATA DA UN INDIRIZZO.

Posso PRENDERE L'INDIRIZZO della PRIMA ISTRUZIONE di una FUNZIONE e METTERLO IN UNA VARIABILE? SI!

IN UN PUNTO del codice, il PUNTATORE a FUNZIONE, MI PERMETTE di CHIAMARE una FUNZIONE:

OVERLOADING DELLE FUNZIONI!
SOVRACCARICO una funzione!

Un **puntatore a funzione** è un tipo di puntatore in C, C++, D, e altri linguaggi di programmazioni stile C. Quando viene dereferenziato esso invoca una funzione, passandole zero o più argomenti come ad una funzione normale. Nei linguaggi di programmazione come il C, i puntatori a funzione possono essere usati per semplificare il codice fornendo un modo semplice per eseguire codice in base a parametri determinati a run-time.

In C è possibile utilizzare dei puntatori a funzioni, ovvero delle variabili a cui possono essere assegnati gli indirizzi in cui risiedono le funzioni, e tramite questi puntatori a funzione, le funzioni puntate possono essere chiamate all'esecuzione.

ESEMPIO:

PARAMETRI

Dichiarazione di una funzione: TIPO_RESTITUTO NOME_FUNZIONE (PARAMDEF1, PARAMDEF2...)

Puntatore a funzione: TIPO_RESTITUTO (*NOME_PTR_A_FUNZIONE) (PARAMDEF1, PARAMDEF2...)

```
FUNCTION ( )
{
    WORK
}
```

*PTR_FUN
=
&FUNCTION

ad es, la seguente dichiarazione definisce un puntatore a funzione che punta a funzioni le quali prendono come argomenti due double, e restituiscono un double (void).

```
DOUBLE (*PTRF) (DOUBLE g, DOUBLE f)
```

Il C tratta i nomi delle funzioni come se fossero dei puntatori alle funzioni stesse.

Quindi, **quando vogliamo assegnare ad un puntatore a funzione l'indirizzo di una certa funzione dobbiamo effettuare un operazione di assegnamento del nome della funzione al nome del puntatore a funzione**

Se ad es. consideriamo la funzione dell'esempio precedente:
double somma(double a, double b);

allora **potremo assegnare la funzione somma al puntatore ptrf** così:
ptrf = somma;

Analogamente, **l'esecuzione di una funzione mediante un puntatore che la punta,viene effettuata con una chiamata in cui compare il nome del puntatore come se fosse il nome della funzione**, seguito ovviamente dai necessari parametri.

ESEMPIO ...

```c
#include <stdbool.h>
#include <stdint.h>
#include <stdio.h>

double somma( double a, double b) ; /* dichiarazione */


int main()
{

    double A=10 , B=29, C;
    double (*ptrf) ( double g, double f);
    ptrf = somma;
    C = ptrf (A,B); /* chiamata alla funz. somma */
    printf("Valore della somma = %f",C);
}

double somma( double a, double b) /* definizione */
{
    return a+b ;
}
```

ESEMPIO CON ARRAY DI PUNTATORI A FUNZIONI

```c
#include <stdbool.h>
#include <stdint.h>
#include <stdio.h>

double somma( double a, double b) ; /* dichiarazione */
double sottrazione (double a, double b);/* dichiarazione */
double divisione (double a, double b);/* dichiarazione */


int main()
{

    double A=10 , B=29, C,S,D;

    double (*ptrff[])(double, double) = { somma, sottrazione, divisione };
    //definisce un array di puntatori a funzioni
    ptrff[0] = somma;
    C = ptrff[0] (A,B); /* chiamata alla funz. somma */
    printf("Valore della somma = %f",C);

    ptrff[1] = sottrazione;
    S = ptrff[1] (A,B);
    printf("Valore della sottrazione = %f",S);

    ptrff[2] = divisione;
    D = ptrff[2] (A,B);
    printf("Valore della divisione = %f", D);

}


double somma( double a, double b)
{
    return a+b ;
}
double sottrazione (double a, double b)
{
    return a-b;
}
double divisione (double a, double b)
{
    return a/b;
}
```

ARRAY di puntatori a funzioni che ritornano un double e richiedono due double come parametri.

0

1

2

ESEMPIO PROFESSORE

```c
#define ELEMENTS 6

int values[] = { 40, 10, 100, 90, 20, 25 };

typedef int (*compare_t)(const void *, const void *);

int compare(const void *a, const void *b)
{
    return (*(int *)a - *(int *)b);
}
void shuffle(int *array, size_t n)
{
    srand((unsigned int)time(NULL));
    if (n > 1)
    {
        size_t i;
        for (i = 0; i < n - 1; i++)
        {
            size_t j = i + rand() / (RAND_MAX / (n - i) + 1);
            int t = array[j];
            array[j] = array[i];
            array[i] = t;
        }
    }
}

void print_array(char *header, int *array, size_t n)
{
    int i;
    printf("%s: ", header);
    for (i = 0; i < n; i++)
        printf("%d ", values[i]);
    puts("");
}



int main()
{
    srand((unsigned int)time(NULL));
    compare_t compare_f1 = compare;
    int (*compare_f2)(const void *, const void *) = compare;
    srand(time(0));
    print_array("Original", values, ELEMENTS);
    qsort(values, ELEMENTS, sizeof(int), compare);
    print_array("Sorted with f.name", values, ELEMENTS);
```

PRENDE la funzione compare, che accetta const void * due volte.

COMPATTA

```c
    srand(time(0));
    print_array("Original", values, ELEMENTS);
    qsort(values, ELEMENTS, sizeof(int), compare);
    print_array("Sorted with f.name", values, ELEMENTS);
    shuffle(values, ELEMENTS);
    print_array("Shuffled", values, ELEMENTS);
    qsort(values, ELEMENTS, sizeof(int), compare_f1);
    print_array("Sorted with f.ptr 1", values, ELEMENTS);
    shuffle(values, ELEMENTS);
    print_array("Shuffled", values, ELEMENTS);
    qsort(values, ELEMENTS, sizeof(int), compare_f2);
    print_array("Sorted with f.ptr 2", values, ELEMENTS);

    return 0;
}
```

COMPATTA

PUNTATORE A FUNZIONE! ✓

QUICK SORT STANDARD IN C;

```
Original: 40 10 100 90 20 25
Sorted with f.name: 10 20 25 40 90 100
Shuffled: 25 90 20 10 100 40
Sorted with f.ptr 1: 10 20 25 40 90 100
Shuffled: 25 90 20 10 100 40
Sorted with f.ptr 2: 10 20 25 40 90 100
```

OUTPUT!

SOTTO AL "COFANO" DEVO CHIAMARE UNA FUNZIONE CONTENUTA IN UN PUNTATORE;

I PUNTATORI PASSANO DA REGISTRI IN CUI DICO AL PROCESSORE che deve INTERPRETARE IL CONTENUTO del REGISTRO come ISTRUZIONE.
È una cosa simile al codice assembly per gestire lo switch;

```
movq $function, %rax
call *%rax
```

Carichiamo l'INDIRIZZO di una funzione IN UN REGISTRO, OVVERO l'INDIRIZZO della PRIMA ISTRUZIONE di una funzione, e chiediamo al PROCESSORE di ESEGUIRE una call all'ISTRUZIONE PUNTATA dal CONTENUTO di RAX.

* = PRENDI IL contenuto di RAX, INTERPRETALO come un INDIRIZZO, SCRIVI SULLO STACK e MODIFICA RIP al contenuto di RAX, e SCRIVI SU STACK L'INDIRIZZO di RITORNO;

I PARAMETRI E IL VALORE DI RITORNO DOVE STANNO? NON CI IMPORTA!