PUNTATORI A FUNZIONE

mercoledì 21 dicembre 2022 11:03

PUNTATORI A FUNZIONE

LA FUNZIONE definisce l'indirizzo in nexoria **della prima ISTRUZIONE**;

QUINDI la SUNZIONE È **CARATTERIZZATA** DA **UN** INDIRIZZO.

**Posso** prendere l'indirizzo della **prima** ISTRUZIONE di una funzione e METTE RIO IN UNA VARIABILE **?** Si!

IN **UN** PUNTO del **codire,** Il **puntatore** a funzione, Hi **permette** di **duamorce** une **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.

PARAMETRI

ESEMPIO:

DichiARAZIONE di una **funzione**: TIPO\_RESTITUITO **NONE.** FUNZIONE **(PARANDERY**, MAAND (F2...) **PUNTATORE** A **FUNZIONE:** TIPO\_RESTITURTO (\*NOME\_PTR\_A\_FUNZIONE) **(PARANDERI**, MAAndeF2...)

FUNCTION()

{

**WORK**

~

**"**PTR\_FUN

ון

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 **9**, DOUBLE 8)

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);

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.

allora **potremo assegnare la funzione somma al puntatore ptrf** cosi:

**ptrf** = **somma;**

ESEMPIO...

**#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**

**#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;

J

double **(**\*ptrff []) (double**,** double)

=

**ARRAY** di puzzatore a funzioni **che** RITORNOmo un double e Richiedono due double **come** for ametri.

**{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);

*1*

=

D= ptrff [2] (A,B)**;**

ptrff [2] **divisione;**

**printf(**"Valore della divisione

2

%f", D);|

门口

**PIRFF** [o] SOMMA

PIRFF [1] SOTTRAZION

PIRFF ***[*2]** Divisione

}

**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;

ESEMPIO PROFESSORE

#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++**)**

{

=

array[j**]**

=

array[i];

**size\_t** j i **rand()** / (**RAND\_MAX /** (n − i**)** + 1**); int** t **=** array[j];

array[i] = t;

}

}

}

**void print\_array**(char \*header**, int** \*array, **size\_t** n**)**

{

**int** i;

**printf(**"%s**:** ", header);

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

**puts(**""**);**

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

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);

ELEMENTS.

ABENde la funcioue compare, che accerta

CONST void deel vatte

COMPATTA

**gick** SORT STARdard

100 Co

}

**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);**

COMPATTA

PUNTOJare **a fuente**!

**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**;**

**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 contenenta IN **UN** PONTATORE**, PUNTATORI** PASSANO DA REGISTRI IN **Cui** dico **AL Processore che** deve **INTERPRETare** IL CONTENUTO del Registro come ISTAZIONE.

E una **cosa** simile al **coolice** assembly **per** gestire lo switch,

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

Carichiamo é **indirizzo** di una funzione in VN **Registro,** Quero l'indirizzo della **PIKA ISTRUZIONo** di Una funzione**,** e chiediamo al **processore di** Eseguire Una call ***all* istruzione Puntata dal** contenuto **di RAX**.

**\* =** prendi Il contenuto di RAX, **INTERFACtalo** come un indirizzo, scrivo sulles stacke

Modifica RIP al contenuto di RAX, e **scrivi** su stack L'Indirizzo ***di* RITORNO**;

SU

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