Linguaggio C

author: "Nicola Ferru"

Definizione

Il linguaggio C è un linguaggio compilato, fortemente tipizzato e case sensitive. Ovvimanete essendo un linguaggio compilato richiederà una ricompilazione nel caso in cui si cambi l'OS oppute l'architettura. È un linguaggio nato per i sistemi Unix negli anni 70' e quindi ha un ottica di programmazione orientata per quello che erano gli elaboratori all'epoca e anche quello che i programmatori usavano all'epoca, seguendo il loro schema mentale, che risultava leggermente diverso dal quello dei programmatori odierni, abbituati a linguaggi orientati alla programmazione ad Oggetti e poco abbituati a gestire manualmente le risorse. Il C è stato modificato nel tampo e attualmente lo standard è il **C11** del 2011.

Architettura

I computer sono basati solitamente sulla architettura x86, più nello specifico x86_64, ma non è l'unica architettura esistente di altre un esempio è l'architettura arm e anche quella powerpc che comunque risultano diffuse e in quel caso la gestione della memoria è diversa e non è certo che le variabili vengano gestite allo stesso modo. ma è sostanzialmente non è un problema perché è possibile verificare la dimensiome delle stesse.

Programma base "**hello world**"

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

int main(){
    printf("hello world");
    return 0;
}
```

Librerie

In C viengono utilizzate delle librerie per caricare e implementare nuove funzionalità, questo consente di ottimizzare al massimo il programma, per caricare una libreria è necessario utilizzare la direttiva al preprocessore `#include<nomeLibreria>` nel caso delle librerie di sistema, mentre, nel caso di librerie locali scritte ad hoc per il programma bisogna utilizzare la direttiva `#include "nomeLibreria"`.

Commenti

Come in ogni buon linguaggio di programmazione anche il C supporta l'utilizzo dei commenti sia per singola linea sia multi-linea, i caratteri che vengono utilizzarti per indicare questo tipo di testo sono ripportati qui sotto nella tabella.

Nome funzione	Descrizione
`// testo`	Commento a linea singola
`/* testo */`	Commento multi-linea

Il commento è un istruzione che non viene interpretata come codice eseguibile dal compilatore, che lo ignorarà e passerà alla prossima istruzione.

funzioni d'input/output basilari

Nome funzione	Descrizione
`printf()`	Funzione per stampare una stringa a schermo
`scanf()`	Funzione che consente di assegnare un valore ad una variabile

queste due funzioni consentono di avere una interazione con il programma anche se non in modo persistente.

```
Il termine persistenza in informatica, il concetto di persistenza si
riferisce alla caratteristica dei dati di un programma di sopravvivere
all'esecuzione del programma stesso che li ha creati: senza questa
capacità questi infatti verrebbero salvati solo in memoria Ram venendo
dunque persi allo spegnimento del computer.
```

By Wikipedia

esempi

```c
    // con testo statico
    printf("testo");
    // con testo dimanico
    int c=4;
    printf("%d",c);
    // acquisizione di un valore
    scanf("%d",&c);
```

Variabili

Queste sono le variabili primitive presenti all'interno del C che compongono anche le variabili complesse e anche le strutture dati.

Nome	Descrizione
int	Numero intero
float	Numero reale

Nome	Descrizione
char	Carattere alfanumerico
double	Numero reale "esteso"
long	Numero intero "esteso"
short	Numero intero "ridotto"
void	Variabile nulla che viene utilizzata normamente per le funzioni che non devono rendere un valore.

- *esteso* - consente di inserire un valore più grande all'interno della variabile, ma occupa più spazio in memoria
- *ridotto* - permette di inserire un valore più piccolo rispetto alla variabile standard ma pesa meno memoria e quindi risulta più ottimizzato se non è necessario l'utilizzo di un **int** completo.

Operatore d'assegnamento

l'operatore d'assegnamento è un operatore che viene utilizzato per assegnare un dato valore all'interno di una variabile. viene espresso dal carattere `=`, non va confuso con l'operatore di confronto che viene espresso con `==` perché altrimenti sorgono dei problemi seri.

Esempio:

```
int i=0; // in questo caso il valore all'interno della variabile è stato sostituito con il valore 0
i=4; // adesso all'interno di i non è resente più 0 ma è presente il 4. Il valore 0 è stato eliminato definitivamente.
```

Ovviamente questo metodo funziona per le variabili primitive non per quelle composte.

Casting

Il casting è un pratica che consente convertire il contenuto di una variabile di un determinato tipo in quello di un altra differente, questa pratica consente di svolgere delle operazioni che altrimenti non sarebbero possibili, come il poter avere il resto tra due interi trasformandolo in un numero relativo e quindi permette di avere un valore più preciso.
Esempio:

```
int a=42;
int b=3;
float ris=(float)a%b;
```

Il tipo lo si specifica tra parentesi tonde prima delle operazione di cui si vuole adattare il risultato.

Consigli

Dare sempre un nome valido alle variabili, perché così il programma risulta più leggibile dopo averlo scritto, cioè quando si scrive un programma non lo si fa mai di fila, molte volte è necessario prendere una pausa quindi se si danno dei nomi senza senso alle variabili diventarà più difficile interpretare l'utilizzo della stessa.

Contenuto delle variabili

Quando viene dichiarata una variabile senza averla inizializzata con un valore, il contenuto che si trova al suo interno è ""ignoto"", cioè il valore all'interno allo spazio di memoria è quello del programma che la stava utilizzando prima quindi non è gestibile e risulta totalmente random per il programmatore. Comunque per verificare la dimensione in memoria di una variabile bisogna utilizzare il `sizeof()`, non in tutti i computer è identico perché dipende dall'architettura.

Limite delle variabili

le variabili avendo un valore fisso hanno un limite nella rapresentazione del dato, quindi per forza di cose è necessario conoscere questo limite per evitare troncature del risultato, perché ovviamente il sistema non ti avvisa del problema ma il risultato perde di precisione.

Rapresentazione nello standard input

rapresentazione	tipo variabile
`%d`	interi
`%f`	float / double
`%e`	decimali, in notazione esponenziale
`%c`	caratteri

Operatori logici e relazionali

Come tutti i linguaggi di programmazione, possiede una parte legata all'algebra booleana e anche agli aspetti logici come AND e l'OR e relazionali come maggiore, minore e uguale, anche le funzioni di comparazione.

Simboli	funzione
`==`	Comparazione
`&&`	AND logico
`||`	OR logico
`!=`	Differenza
`<`	minore
`>`	maggiore
`<=`	minore\uguale

Simboli	funzione
>=	maggiore\uguale

Operatori aritmetici

Simboli	funzione
+	somma
-	sottrazione
/	divisione
%	modulo (resto divisione)

Operatori composti

Simboli	funzione
+=	somma
-=	sottrazione
/=	divisione
%=	modulo (resto divisione)

esempio

```
cont+=5
// invece di
count=count+5
```

occhio alle variabili non inizializzate perché non sono gestibili quindi non si può sapere che valore possa assumere il risultato.

Contizioni "**i casi**"

In C è possibile verificare dei casi, con l'utilizzo della funzione `if` che consente di valutare una determinata valutare una condizione e nel caso sia prevista una condizione alternativa va utilizzato la funzione `else` e poi l'opzione alternativa.

esempio

```
if(x!=0)
  printf("il valore è maggiore di 0, perché il valore è %d",x);
else printf("il numero è 0");
```

switch case

ovviamente esistendo casi con più possibilità esiste una funzione che consente aquisendo una variabile o una condizione di gestire diversi casi più uno "`default`" che viene scelto nel caso in cui il contenuto della variabile in questione non sia stato previsto.

```
int x;
scanf("%d",&x);
switch (x) {
  case (x>5):
  // istruzioni
  break;
  case (x-4>=0):
  // istruzioni
  break;
  default:
  // istruzione predefinita
  break;

}
```

Cicli "interogazioni"

L'iterazione è l'atto di ripetere un processo con l'obiettivo di avvicinarsi a un risultato desiderato. Ogni ripetizione del processo è essa stessa definita un'iterazione, e i risultati di una sono utilizzati come punto di partenza per quella successiva. Diffuso è l'utilizzo negli algoritmi e nella programmazione in ambito informatico, ma anche in campi come il project management.
By Wikipedia

Teorema di Jacopini-Bohm

Nella programmazione informatica, l'iterazione, chiamata anche ciclo o con il termine inglese loop, è una struttura di controllo, all'interno di un algoritmo risolutivo di un problema dato, che ordina all'elaboratore di eseguire ripetutamente una sequenza di istruzioni, solitamente fino al verificarsi di particolari condizioni logiche specificate.

Assieme alla sequenza o blocco e alla selezione è una delle tre strutture fondamentali per la risoluzione algoritmica di un dato problema secondo il Teorema di Böhm-Jacopini. Esistono varie forme di iterazione; le più conosciute sono il MENTRE (in inglese: while..do), il RIPETI (in inglese: repeat..until o do..while), ed il PER (comunemente detto ciclo for). Si può dire che l'iterazione è l'anello forte della programmazione che consente di automatizzare portando a termine un processo, al quale non basta la semplice esecuzione in sequenza di istruzioni.

Il cosiddetto "**ciclo infinito**", o "**loop infinito**", è un caso di iterazione dovuto solitamente ad un errore di programmazione che manda in stallo l'esecuzione del programma, mentre in alcune tecniche di programmazione soprattutto con microcontrollori è utilizzato in maniera voluta per iterare infinitamente all'interno del programma.

while & do/while

Il while e il do/while sono molto simili tra loro, nel caso del primo controlla la condizione all'inizio, mentre, il secondo controlla la conzione alla fine. questa funzione è molto utilizzata ed è l'unico modo per le più svariate operazioni, quindi bisogna avere una totale padronanza del

for

Logica dei puntatori

i puntattori vengono utilizzati in modo esplicito nel C, per gestire la mamoria, infatti, esistono le variabili dedicate, che vengono dichiarate come le altre variabili ma con un * davanti al nome della stessa `int *i`. Sono fondamentali per l'utilizzo degli array dinamici e anche per la gestione dei file, proprio per la loro natura. Una delle funzioni che utilizza un puntatore è proprio la ruimentale scanf(), ogni variabile possiede il suo pountatore, per accedere al puntatore invece che al contenuto della stessa è necesario mettere davanti al nome il carattere `&`.

Gestione dei file

Come in tutti i linguaggi strutturati, il `C` ha la possibilità di gestire dei file, ovviamente è sempre il C, quindi le operazioni vanno svolte manualmente. E soprattutto richiedono una certa attenzione, perché il rischio è quello di sovrascrivere qualche documento in portante che una volta perso non lo si recupera, è perso definitivamente.

Variabili dedicate

Nome variabile	Descrizione
FILE	questo tipo di variabile indira un file fisico presente sul device d'archiviazione in questione

Effettivamente la cosa importate in questi casi è fornire un percorso valido in cui salvare lo stesso.

Funzioni dedicate

Nome funzione	Descrizione
`fprintf()`	Funzione per stampare una stringa in un file
`fscanf()`	Funzione che consente di assegnare un valore da file
`fopen()`	Funzione che consente di aprire un file
`fclose()`	Funzione che consente di chiudere un file

esempi

Operazione di scrittura su file

```
/* librerie standard */
```

```c
int main() {
  FILE *fd;
  int x=-32;

        /* apre il file in scrittura */
  fd=fopen("scrivi.txt", "w");
  if( fd==NULL ) {
    perror("Errore in apertura del file");
    exit(1);
  }


        /* scrive il numero */
  fprintf(fd, "%d\n", x);


        /* chiude il file */
  fclose(fd);

  return 0;
}
```

Operazione di recupero dei dati da un file

```c
int main(){
  FILE *pf;
  char nome_file[128];
  printf("Inserisci il nome di un file (percorso completo): ");
  scanf ("%s" , nome_file );
  pf = fopen(nome_file , "r"); if (pf) {
      fseek(pf, SEEK_END);
      printf ("%s è lungo %ld bytes" , nome_file , ftell (pf ));
      fclose( pf );
  }else{
    printf("%s non esiste !", nome_file );
  }
  return 0;
}
```

Consiglio

Il consiglio è di scrivere una funzione esterna al `main` per gestire queste operazioni, perché così si possono gestire al meglio le eccezioni e quindi si può impedire all'utente finale di fare potenziali danni al suo stesso sistema delimitando le operazioni possibili guidandolo nel gio sto modo. Perché l'utente non sa cosa sta facendo.

ATTENZIONE!!!!

Quando si apre un file bisogna sempre e comunque chiuderlo quando si finisce di esequire una determinata operazione, altrementi si rischia creare degli effettivi problemi, uno dei tanti è che il file risulta in utilizzo finché il programma è aperto anche se non è più necessario, per di più rischi di danneggiare lo stesso scrivendoci per errore e tanti altri problemi logici che non possono essere espressi in due righe.

Le funzioni

Le funzioni vengono utilizzate in programmazione per poter scomporre un problema complesso in tanti più piccoli più semplici da gestire e da scrivere. La funzione può essere di diversi tipi, infatti, proprio come una variabile condivide gli stessi tipi proprio per il fatto che la stessa deve rendere qualcosa indietro che sia un intero o un numero reale, l'importante è che tutto venga gestito nel migliore dei modi e venga scritto modo più ordinato possibile.

esempio

funzione che rende una media

```
float media (int v[]) {
  int i,
  float ris=0;
  for (i=0;i<N;i++)
    ris+=(float)v[i];
    ris/=N;
  return ris;
}
```