Corso di Laurea in Scienze dell'Informazione di Cesena
Sistemi per l'Elaborazione dell'Informazione 1
a.a. 1999/2000

Appunti dalle Lezioni

Il linguaggio C

Ciclo di lezioni tenute da:

Vittorio Ghini
e-mail **vittorio@csr.unibo.it**

Letture Consigliate:

Brian W. Kernighan, Dennis M. Ritchie, "**Linguaggio C ANSI**", ed. Jackson Libri.

Caratteristiche del linguaggio C

- utilizzo frequente di chiamate a **funzioni**.
- **debole controllo sui tipi di dato**. A differenza del Pascal, il C permette di operare con assegnamenti e confronti su dati di tipo diverso, in qualche caso solo mediante un type cast (conversione di tipo) esplicito.
- **linguaggio strutturato**. Il C prevede costrutti per il controllo di flusso, quali raggruppamenti di istruzioni, blocchi decisionali (if-else), selezione di alternative (switch), cicli con condizione di terminazione posta all'inizio (while, for) o posta alla fine (do) e uscita anticipata dal ciclo (break).
- **programmazione a basso livello** facilmente disponibile
- implementazione dei **puntatori** (ampio uso di puntatori per memoria, vettori, strutture e funzioni)
- **portabilità** sulla maggior parte delle architetture.
- disponibilità di **librerie standard**.
- la grande libertà messa a disposizione dai puntatori e dagli array rende facile commettere errori, soprattutto con array e puntatori.

Lo standard per i programmi C in origine era dato dalle caratteristiche messe a punto da Brian Kernighan. Al fine di rendere il linguaggio piu' accettabile a livello internazionale, venne messo a punto uno standard internazionale chiamato ANSI C (American National Standards Institute).

N.B. - Metodo di presentazione del linguaggio C

All'inizio, per poter fare qualche prova da subito, utilizzeremo alcune funzioni di libreria senza scendere nei dettagli, in attesa di specificarle meglio nelle lezioni successive.
Ad es. la funz printf, che è complicata da usare, verrà descritta solo nei caratteri fondamentali, nell'ottica di scendere nei dettagli quando avrete imparato qualcosina di più sul C.

Il primo programma in linguaggio C

Un minimo programma in C e':

```
main()
  {
  }
```

che corrisponde a un programma in Pascal:

```
program minimum;
begin
end
```

Caratteristiche di ogni programma C:

- **Ogni programma C deve contenere una e una sola funzione main()**, che rappresenta il programma principale, ed l punto di inizio dell'esecuzione del programma.

- La parentesi graffa aperta **{** indica l'inizio di un blocco di istruzioni, e corrisponde al **begin** del pascal.
- La parentesi graffa chiusa **}** indica la fine di un blocco di istruzioni, e corrisponde al **end** del pascal.
- Per ogni parentesi graffa aperta **{** deve essercene una chiusa **}**.

- I commenti possono essere posti ovunque
- L'inizio del commento è dato dalla coppia **/***
- La fine del commento è indicata da ***/**
- Non si puo' inserire un commento in un altro (vietato l'annidamento dei commenti.

Ad esempio:

```
/* Esempio di programma in C con problemi nei commenti */
main()
{
/* Un commento */                                    ESATTO
/* Commento /* Ancora un commento */ QUI ERRORE */
}
```

Il seguente esempio e' un programma che produce l'output sullo schermo della frase "Hello World":

```
#include <stdio.h>

   int main()
   {
        printf("Hello World \n");
        exit(0);
   }
```

- In C, ci vuole un punto e virgola **;** dopo ogni istruzione.
- L'istruzione "**printf(...)**" chiama la funzione printf che visualizza cio' che gli viene passato come argomento (\n indica l'andata a capo), e ricorda la write del Pascal.

- L'istruzione "**exit(0)**" chiama la funzione exit che termina il programma e restituisce al sistema operativo il codice 0.

- In C non esiste la distinzione che esiste in pascal tra funzioni e procedure, in C sono tutte funzioni, che possono restituire un qualche risultato oppure no. Le chiamate ad ogni funzione in C si effettuano chiamando il nome della funzione seguita dalle parentesi tonde, aperta e chiusa, all'interno delle quali vengono passati i parametri necessari. Anche se la funzione non richiede nessun parametro, il suo nome deve essere seguito dalle parentesi tonde aperta e chiusa.

Sviluppo di un Programma in linguaggio C

Lo sviluppo di un programma in C richiede le seguenti fasi:
Scrittura, Compilazione, Esecuzione e Debugging.

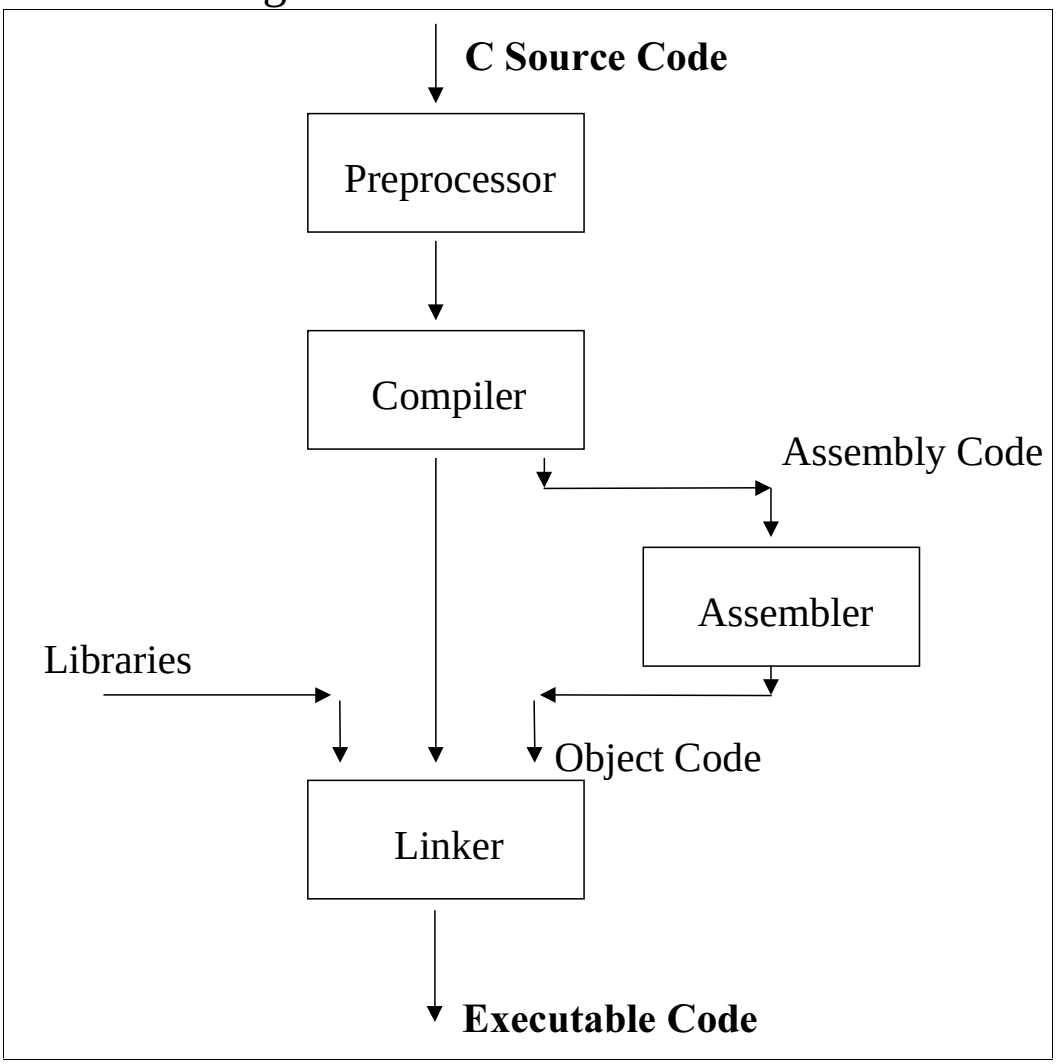
- **Scrittura del Programma in Linguaggio C:**
  - Per creare un file contenente un programma C si usa un qualsiasi text editor quale "edit" "notepad" in pc con sistemi operativi dos-windows, oppure vi, emacs, xedit in host con s.o. unix.
  - Il nome del file deve avere l'estensione "**.c**", ad esempio, prog.c.
  - Il contenuto, ovviamente, deve rispettare la sintassi C.
- **Compilazione:**
  - La compilazione di un programma scritto in C si effettua mediante programmi detti **compilatori** quali Microsoft C Compiler (cl.exe) in sistemi dos-windows, o Gnu C Compiler (gcc) in sistemi Unix.
  - Questi compilatori in realtà svolgono 3 funzioni: **preprocessing**, **compilazione vera e propria** (eventualmente passando da uno step intermedio che consiste nella generazione di file in codice assembly) per generare dei moduli oggetto, **linking** (collegamento) dei vari moduli oggetto e delle eventuali librerie.
  - queste tre funzioni, in ambiente unix sono solitamente eseguite da tre programmi diversi (preprocessor, compiler, linker) che vengono attivati da un programma di gestione anch'esso comunemente detto compilatore. In ambienti microsoft spesso preprocessing e compilazione sono eseguiti da uno stesso programma, ed il linking è effettuato da un altro programma chiamato dal compilatore. Esiste inoltre un utility (il make) che permette di eseguire le varie fasi solo per quei files che sono stati modificati più recentemente (in particolare i files modificati dopo l'esecuzione dell'ultima compilazione), limitando in tal modo il lavoro del compilatore.

COMPILATORI A LINEA DI COMANDO		
\ ambiente fase \	DOS-WINDOWS (Microsoft)	UNIX-LINUX (GNU)
gestione	nmake.exe	make
preprocessing	cl.exe	cpp
compilazione	cl.exe	gcc
linking	link.exe	ld

- **Esecuzione e debugging**:
  - L'eseguibile ottenuto mediante compilazione e link viene eseguito per verificarne il funzionamento e scoprire eventuali errori.
  - N.B. Un programma sviluppato in C che sembra funzionare dopo la prima compilazione ha sicuramente degli errori nascosti !!!!
  -
- Esistono in commercio i cosiddetti "ambienti di sviluppo integrati" quali ad es il Microsoft Visual C, che nascondono all'utente questi aspetti della compilazione, mediante un'interfaccia visuale, e che consentono di effettuare il debugging in maniera più facile, seguendo passo a passo l'esecuzone del programma.

Il Modello di Compilazione per il C

Gli step essenziali della compilazione per un programma sviluppato in C sono rappresentati nel seguente schema:



- NOTA BENE: Il passaggio attraverso il linguaggio assembly solitamente non è necessario, e viene utilizzato solo a scopo di debugging.

- **Il Preprocessore**

Il preprocessore prende in input un file di codice sorgente C e:

1) processa le cosiddette **direttive al preprocessore**, che sono essenzialmente:
   - **Inclusione di file**
     #include <nomefile>
     prende il file nomefile e lo inserisce al posto della direttiva
   - **Definizione di Macro**
     #define SIMBOLO DefinizioneDiSimbolo
     definisce una Macro, e da quel momento, ogni volta che il preprocessore incontra SIMBOLO lo sostituisce con DefinizioneDiSimbolo. Es. #define LENGTH 100
   - **Compilazione Condizionale**
     #ifdef SIMBOLO
     istruzioni C /* eseguite se prima esiste #define SIMBOLO */
     #else
     istruzioni C /* eseguite se prima NON esiste #define SIMBOLO */
     #endif

2) elimina i commenti contenuti nel sorgente.

3) genera così un nuovo file senza commenti e senza più necessità di effettuare sostituzioni, pronto per essere processato dal compilatore.

Un esempio di come lavora il processore:

prima	prima	dopo il passaggio col processore
/* file **header.h** */ extern int i; extern double f; /* fine header.h */	/* file **prova.c** */ #include "header.h" /* size vettore */ #define SIZE 10 int vettore[SIZE];	/* file **prova.E** */ extern int i; extern double f; int vettore[10];

- **Il Compilatore.**

Il compilatore prende in input il codice ottenuto come output dal preprocessore, e crea il codice assembly, cioè un codice che (semplificando) è mappabile 1 a 1 con il codice macchina ma è scritto in una forma umanamente comprensibile.

- **L'Assemblatore.**

Prende in input il codice assembly, e crea il codice oggetto, cioè del codice macchina (comprensibile per il calcolatore) in cui ancora esistono dei riferimenti non risolti.

- **Il Linker.**

detto anche collegatore, si occupa di prendere in input i vari files di codice oggetto, sia quelli generati dall'assemblatore, o direttamente dal compilatore a partire al codice del programmatore, sia quello delle librerie disponibili, e di generare il file eseguibile, collegando tutti i riferimenti a variabili e funzioni da un file all'altro.

In particolare si occupa di collegare, senza farlo vedere all'utente, una porzione di codice oggetto, reso disponibile come libreria, che:

- **appronta l'ambiente per il processo quando questo va in esecuzione** e
- **passa il controllo alla funzione main** che l'utente deve avere definito.

Questa porzione di codice si occupa ad esempio, ma non solo, di consegnare al programma i parametri passati tramite la riga di comando, e questo senza che l'utente se ne debba occupare.

Le librerie sono importanti perchè contengono il codice che esegue funzioni quali ad es le funzioni matematiche e le funzioni di I/O.

In un certo senso, le librerie del C, assomigliano alle **Unit del Pascal**, che, come ricorderete vengono accedute mediante la clausola **uses**, e che mettono a disposizione funzioni procedure e variabili.

Non tutte le librerie sono disponibili su tutte le architetture di calcolatori. Il ristretto insieme di funzioni definito da ANSI viene però reso disponibile dalla maggior parte dei compilatori.

Molti compilatori mettono a disposizione librerie proprietarie, che se da un lato facilitano la programmazione all'utente, dall'altro lato rendono difficile il porting del programma, cioè il trasferimento del programma su una architettura diversa.

E' necessario in genere **indicare quali librerie il linker deve utilizzare**, passando al linker stesso alcuni flag nella linea di comando.
Alcune librerie sono utilizzate per default, altre devono essere esplicitamente nominate.

Gli ambienti integrati in genere nascondono o almeno facilitano la soluzione delle problematiche relative alla scelta delle librerie da utilizzare.