Fondamenti di Informatica L-B (L-Z)
Esercitazioni

A.A. 2007/08

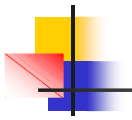
Tutor: Barbara Pettazzoni

barbara.pettazzoni@studio.unibo.it

Informazioni Utili

- Orario delle Esercitazioni
  - Martedì 14 - 16 Gruppo L – Q
  - Martedì 16 - 18 Gruppo R – Z
- Ricevimento
  - Al termine delle esercitazioni in laboratorio e, dopo termine delle lezioni, previo appuntamento tramite email
- Contatti
  - e-mail: barbara.pettazzoni@studio.unibo.it
  - Web Site: http://www-db.deis.unibo.it/Courses/FIL-B
  - Universibo: https://www.universibo.unibo.it

2



Perché usare UniversiBO?

- Il vostro dubbio puó essere il dubbio di vostri colleghi
- Se io rispondo sul forum, la risposta non é solo tra me e chi mi ha mandato la mail…ma é a disposizione di tutti.
- Per lo stesso ragionamento, puó essere che qualcun altro abbia giá chiesto quello che volete chiedere…
- …e quindi la risposta potrebbe essere giá sul forum ☺

3

Strumenti Utilizzati

- LCC-Win32 (Win 95 e successivi)
  - Ambiente integrato per lo sviluppo di software in C
  - In laboratorio, accessibile tramite il collegamento 'wedit'
  - Software gratuito, scaricabile, con relativo manuale, dai siti
    - ftp://ftp.cs.virginia.edu/pub/lcc-win32/lccwin32.exe
    - ftp://ftp.cs.virginia.edu/pub/lcc-win32/manual.exe
  - Molto utile è il tutorial messo a disposizione nel sito di LCC che presenta esempi di programmazione C sia a livello elementare che a livello avanzato
    - ftp://ftp.cs.virginia.edu/pub/lcc-win32/tutorial.pdf

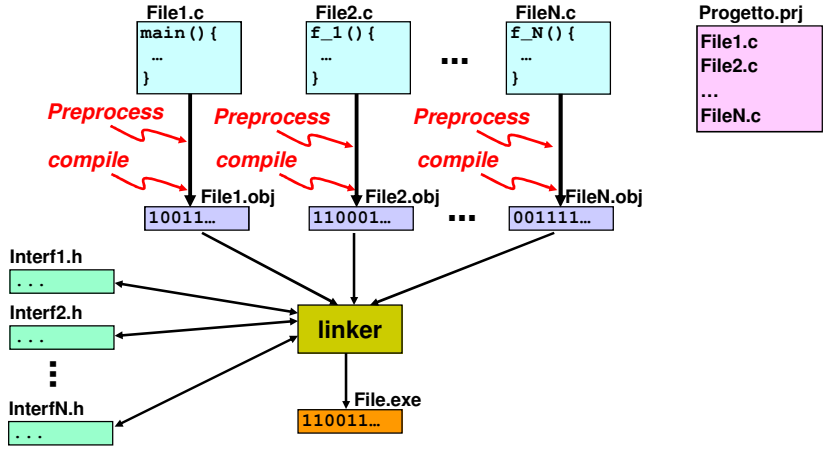
4

Creazione e gestione di un progetto

Provate ad utilizzare per loggarvi:
Username: lab3_numerocomputer (c'é scritto sopra il computer)
Password: come lo username!

Editing, compilazione, esecuzione

- Un **progetto** è un programma che, *durante la fase di editing*, è stato strutturato su più file



Editing, compilazione, esecuzione (2)

Precompilazione: la fase di precompilazione è eseguita dal preprocessore C ed ha il compito di espandere alcune forme abbreviate come ad esempio sostituire alle costanti presenti nel codice i loro valori settati attraverso la *define*

Compilazione: nella fase di compilazione vengono cercati eventuali errori presenti nel codice sorgente e tale codice viene tradotto in istruzioni scritte in linguaggio Assembler. Tali istruzioni non sono ancora eseguibili in quanto il compilatore lascia in sospeso tutte le funzioni che sono invocate nel programma, ma che in esso non sono definite (es: *scanf* – *printf* presenti nella libreria *stdio*)

Linker: nella fase di link tali funzioni vengono cercate nelle librerie indicate attraverso il comando *#include* e se vengono trovate vengono collegate al programma, altrimenti viene restituito un messaggio di errore.

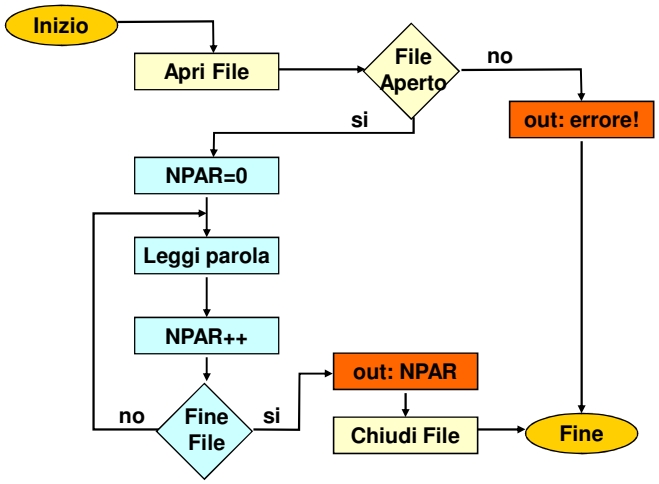
7

Problema

- Scrivere un programma che conta le parole contenute in un file
  - Individuazione dell'algoritmo risolutivo
  - Creazione del progetto "ContaParole.prj"
  - Creazione del file "ContaParole.c" e inserimento nel progetto
  - Scrittura del codice che realizza l'algoritmo
  - Compilazione
  - Collegamento (linking) e creazione dell'eseguibile
  - Test di correttezza (esecuzione)

8

Algoritmo Risolutivo

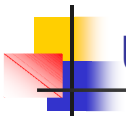


9

Un breve richiamo sui file in C

- **Il tipo FILE è una struttura definita in header standard <stdio.h>,** che l'utente non ha necessità di conoscere nei dettagli (che spesso cambiano da un compilatore all'altro)
- **L'utente** definisce e usa, nei suoi programmi, solo dei ***puntatori a FILE***
- Dopo l'apertura, il programma **opera sul file *utilizzando la variabile (puntatore)* che lo rappresenta**: il sistema operativo provvederà a effettuare l'operazione richiesta sul file associato a tale simbolo
- Al **termine**, la corrispondenza dovrà essere *soppressa:* **operazione di *chiusura del file***

10



Un breve richiamo sui file in C (2)

Per aprire un file si usa la funzione:

```
FILE* fopen(char fname[], char modo[])
```

modo specifica ***come*** aprire il file:

- **r apertura in lettura (read)**
- **w apertura in scrittura (write)**
- **a apertura in aggiunta (append)**

seguita opzionalmente da:

- **t apertura in modalità testo (default)**
- **b apertura in modalità binaria (che noi non tratteremo)**

11

Un breve richiamo sui file in C (3)

Il valore restituito da fopen() è

- **un *puntatore a FILE, da usare in tutte le successive operazioni* sul file**
- **NULL in caso l'apertura sia fallita**

Controllare il valore di uscita della fopen() è l'unico modo per sapere se il file è stato aperto correttamente

Per chiudere un file si usa la funzione:

```
int fclose(FILE*)
```

che restituisce 0 se tutto è andato bene o EOF in caso di ERRORE

12

Struttura base di ContaParole.c

```
/* ContaParole.c */

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

main()
{
   FILE *fp;
   int num_par;
   /* Eventuale definizione altre variabili */

   fp = fopen("Nomefile.txt","rt");
   //Controllo della corretta apertura del file
   if (fp==NULL)
   {
      printf("Errore nell'apertura del file.\n");
      exit(1);
   }
   ...
   //Stampa a video del risultato
   printf("Il file contiene %d parole.\n",num_par);
   fclose(fp);
}
```


Suggerimento

- Utilizzare la funzione di libreria

```
int fscanf(FILE* fp, string-format, char* buffer);
```

- Una possibile modifica:
  - Modificare "ContaParole.c" in modo da visualizzare a schermo le parole conteggiate, separate da uno spazio

Una possibile soluzione:ContaParole.c

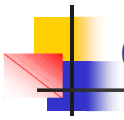
```
#include <stdio.h>
#include <stdlib.h>
main()
{
   FILE *fp;
   int num_par=0;
   int nletti;    /* Char letti dalla fscanf */
   char buffer[512];

   fp = fopen("Intervista.txt","rt");
   if(fp==NULL) { printf("Errore nell'apertura del file.\n");exit(1); }

   while( !feof(fp) )
   {
      nletti = fscanf(fp, "%s",buffer);
      if(nletti>0) num_par++;
   }

   printf("\nIl file contiene %d parole.\n",num_par);
   fclose(fp);
}
```

15

ContaParole.c: Commenti

- `void exit(int stato);`
  - Procedura della libreria `<stdlib.h>`
  - Usata per indicare una terminazione anomala
  - Invocata in qualsiasi punto del programma (anche all'interno di una funzione) termina l'esecuzione restituendo il valore indicato in `stato` (tipicamente diverso da 0)

16



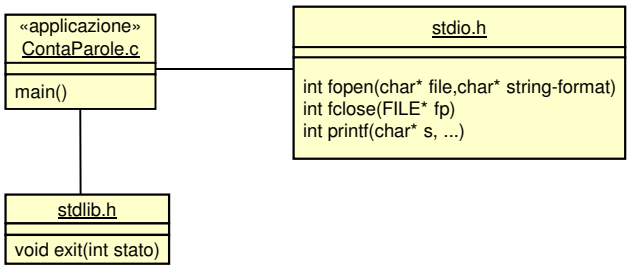
ContaParole.c: Commenti

- Nella soluzione la maggior parte del lavoro è svolto dalla funzione di libreria *fscanf(...)*
  - L'implementazione risulta **semplice**
  - Il codice è **leggibile**
- Programmare usando funzioni e procedure vuol dire *scomporre un problema difficile in sotto-problemi* più facilmente risolvibili

17

Rappresentazione Visuale

- Scrivere un programma che conta le parole contenute in un file

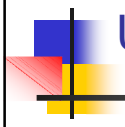


18

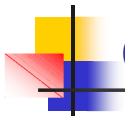
Next Time:

- Progettare il tipo di dato astratto **Contatore**
  1. Definire **cos'è** un contatore
  2. Definire le **operazioni** relative ad un contatore
  3. Definire la **rappresentazione interna** dell'ADT contatore
  4. Scrivere l'**interfaccia** dell'ADT contatore
  5. **Implementare** l'ADT in modo che realizzi l'interfaccia specificata
- Scrivere un programma che, utilizzando le funzionalità offerte dall'ADT Contatore, conteggi le parole contenute in un file

19



Una possibile soluzione alternativa



ContaParole.c: Soluzione Alternativa

```
#include <stdio.h>
#include <stdlib.h>
main()
{
   FILE *fp;
   int num_par=0;
   int nuova_par=0;    // Flag di inizio parola
   char c;

   fp = fopen("NomeFile.txt","rt");
   if (fp==NULL) {printf("Errore nell'apertura del file.\n");exit(1); }

   while( (c=getc(fp)) != EOF )
   {
      if( c==' ' || c=='\n' || c=='\t') nuova_par=0;
      else if(nuova_par==0)
            {
               nuova_par=1;
               num_par++;
            }
   }
   printf("Il file contiene %d parole.\n",num_par);
   fclose(fp);
}
```