

Esercizio 1: Strutture e file

Esercizio - lettura/filtro/scrittura

- È dato un file di testo **PEOPLE.TXT** che contiene i dati di una serie di persone (non più di 20), una persona per riga
- Si vuole realizzare un programma che, una volta letti da file i dati di queste persone, ne estrapoli l'insieme di persone compatibili con una nuova persona data, salvando il risultato sul file binario **PARTNERS.DAT**
 - Due persone sono compatibili se sono di sesso diverso e la differenza di età, *riferita solo all'anno*, non supera i 5 anni

1

Esercizio 1 - Strutture dati

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

#define NUMEROPERSONE 20
#define DIMCOGNOME 21
#define DIMNOME 21

typedef struct dataStruct
{
    int giorno, mese, anno;
} Data;

typedef struct personaStruct
{
    char cognome[DIMCOGNOME], nome[DIMNOME], sesso;
    Data nascita;
} Persona;
```

2

Esercizio 1 - Lettura (1)

```
void lettura( char nomefile[],
             Persona v[],
             int* pindice)
{
    Persona x;
    Boolean more = true;
    *pindice = 0;
    FILE *f = fopen(nomefile, "r");
    if (f == NULL)
    {
        printf("Impossibile aprire file di
        ingresso");
        exit(1);
    }
    ...
```

3

Esercizio 1 - Lettura (2)

```
do
{
    more = readField(x.cognome, ';', f);
    more = more && readField(x.nome, ';', f);
    more = more && fscanf(f, "%d/%d/%d %c\n",
                        &x.nascita.giorno,
                        &x.nascita.mese,
                        &x.nascita.anno,
                        &x.sesso);

    if(more)
    {
        v[*pindice] = x;
        (*pindice)++;
    }
}
while(more);
fclose(f);
}
```

4

Esercizio 1 - Compatibili

```
Boolean compatibili(Persona p1,
                    Persona p2)
{
    Boolean compSesso, compAnno;
    compSesso = ( p1.sesso != p2.sesso );
    compAnno =
        abs(p1.nascita.anno - p2.nascita.anno)<=5;
    return compSesso && compAnno;
}
```

5

Esercizio 1 - Main (1)

```
int main()
{
    Persona elenco[NUMEROPERSONE], utente;
    int indiceElenco = 0, i;
    FILE *fbin;
    lettura("PEOPLE.TXT", elenco, &indiceElenco);
    printf("\n\nNome e cognome: ");
    gets(utente.nome);
    gets(utente.cognome);
    printf("data di nascita (gg/mm/aaaa): ");
    scanf("%d/%d/%d",
        &utente.nascita.giorno,
        &utente.nascita.mese,
        &utente.nascita.anno);
    /* sopprime il fine linea rimasto sull'input */
    scanf("%*c");
    printf("Inserire il sesso (M/F): ");
    scanf("%c%c", &utente.sesso);
    ...
}
```

6

Esercizio 1 - Main (2)

```
...
fbin = fopen("PARTNERS.DAT", "wb");
if (fbin==NULL)
{
    printf("Impossibile aprire file di uscita\n");
    exit(2);
}
printf("Compatibili con %s %s:\n", utente.nome, utente.cognome);
for (i=0; i<indiceElenco; i++)
    if (compatibili(utente, elenco[i]))
    {
        fwrite(&elenco[i], sizeof(Persona), 1, fbin);
        printf("%s %s nato(a) il %d/%d/%d\n",
            elenco[i].nome,
            elenco[i].cognome,
            elenco[i].nascita.giorno,
            elenco[i].nascita.mese,
            elenco[i].nascita.anno);
    }
fclose(fbin);
}
```

7

Esercizio 2

Gestione di una libreria

- Realizzare un programma di gestione per una libreria
- Il programma deve acquisire **da standard input una serie di libri** (con numero massimo fissato a priori)
 - NOTA: si supponga che l'acquisizione avvenga per codici ordinati in senso crescente
- Il programma deve successivamente presentare un menu a tre voci:
 - Stampa a video dell'insieme di libri
 - Ricerca e stampa di un libro a partire dal codice
 - Si usi a tal scopo la RICERCA BINARIA
 - Uscita dal programma
- Estendere poi il programma supportando il salvataggio/caricamento dei libri su/da file binario

8

Esercizio 2 - Strutture dati

```
typedef enum {CRITICAL_ERROR_LOAD, NOT_ENOUGH_SPACE, LOAD_OK}
LoadResult;
typedef enum{CRITICAL_ERROR_SAVE, SAVE_OK} SaveResult;
typedef enum{NOT_FOUNDED, FOUNDED} FindResult;

typedef struct authorStruct
{
    char name[20];
    char surname[20];
} Author;
typedef struct bookStruct
{
    int code;
    char title[20];
    Author auth;
    int copies;
} Book;
typedef Book BookShop[MAXDIM];
```

9

Esercizio 2 - Lettura (1)

```
Author loadAuthor() {
    Author a;
    printf("author name: "); scanf("%s", a.name);
    printf("author surname: "); scanf("%s", a.surname);
    return a;
}

Book loadBook() {
    Book b;
    printf("book ID: "); scanf("%d", &b.code);
    printf("book title: "); scanf("%s", b.title);
    b.auth = loadAuthor();
    printf("copies number: "); scanf("%d", &b.copies);
    return b;
}
```

10

Esercizio 2 - Lettura (2)

```
LoadResult loadBookShop(BookShop shop, int* num, int maxdim)
{
    int i = 0;
    do
    {
        if(i > maxdim)
            return NOT_ENOUGH_SPACE;
        printf("Inserisci un libro\n");
        shop[i] = loadBook();
        i++;
        printf("Altro libro (s/n)? ");
        getchar();
    }
    while(getchar() == 's');
    *num = i;
    return LOAD_OK;
}
```

11

Esercizio 2 - Stampa

```
void printAuthor(Author a)
{
    printf("author: %s %s\n", a.name, a.surname);
}

void printBook(Book b)
{
    printf("code: %d\n", b.code);
    printf("title: %s\n", b.title);
    printAuthor(b.auth);
    printf("copies: %d\n", b.copies);
    printf("-----\n");
}

void printBookShop(BookShop shop, int dim)
{
    int i;
    for(i = 0; i < dim; i++)
        printBook(shop[i]);
}
```

12

Esercizio 2 - Comparazione

```
int compare(Book b1, Book b2)
{
    return b1.code - b2.code;
}
```

13

Esercizio 2 - Ricerca binaria

```
int binarySearch(BookShop shop, int dim, int toSearch)
{
    int midPos = dim / 2;
    Book bookToSearch;
    bookToSearch.code = toSearch;
    if(compare(shop[midPos], bookToSearch) == 0)
        return midPos;
    if(midPos == 0) return INT_MIN;
    if(compare(shop[midPos], bookToSearch) > 0)
        return binarySearch(shop, midPos, toSearch);
    else
    {
        int startPos = midPos + 1;
        return startPos + binarySearch(&shop[startPos],
                                       dim - startPos,
                                       toSearch);
    }
}
```

14

Esercizio 2 - Ricerca libro

```
FindResult findCode(BookShop shop, int dim)
{
    int code, pos;
    printf("Insert code: ");
    scanf("%d", &code);
    pos = binarySearch(shop, dim, code);
    if(pos < 0)
        return NOT_FOUNDED;
    printBook(shop[pos]);
    return FOUNDED;
}
```

15

Esercizio 2 - Main (1)

```
int main()
{
    BookShop shop;
    int loaded;
    char resp;
    LoadResult lRes;
    FindResult fRes;
    lRes = loadBookShop(shop, &loaded, MAXDIM);
    handleLoadResult(lRes);
    if(lRes != OK)
        return 1;
    ...
}
```

16

Esercizio 2 - Main (2)

```
...
do
{
    printf("p)\t print\n f)\t find\n e)\t exit\n");
    while(getchar()!='\n');
    resp = getchar();
    switch(resp)
    {
        case 'p':
            printBookShop(shop, loaded); break;
        case 'f':
            fRes = findCode(shop, loaded);
            handleFindResult(fRes); break;
        case 'e': break;
    }
}
while(resp != 'e');
return 0;
}
```

17

Esercizio 2 - Scrittura binaria

```
typedef enum{CRITICAL_ERROR_SAVE, SAVE_OK} SaveResult;

SaveResult saveBookShop(BookShop shop, int dim, char* fileName)
{
    int i;
    FILE *f = fopen(fileName, "wb");
    if(f == NULL)
        return CRITICAL_ERROR_SAVE;
    for(i = 0; i < dim; i++)
        fwrite(&shop[i], sizeof(book), 1, f);
    fclose(f);
}
```

18

Strutture e File

```
typedef enum{CRITICAL_ERROR, NOT_ENOUGH_SPACE, OK} LoadResult;

LoadResult loadBookShopFromFile( BookShop shop, int *dim,
                                char* fileName, int maxdim)
{
    int i = 0, result;
    FILE *f = fopen(fileName, "r");
    if(f == NULL)
        return CRITICAL_ERROR_LOAD;
    do
    {
        if (i > maxdim)
            return NOT_ENOUGH_SPACE;
        result = fread(&shop[i], sizeof(book), 1, f);
        i++;
    }
    while(result>0);
    *dim = i;
    return OK;
}
```

19

Esercizio 3: uso di file

Conta parole

- Si scriva un programma C che conti il numero dei caratteri, delle parole e delle linee contenute in un file di testo **PAROLE.TXT**

20

Esercizio 3 - Soluzione (1)

```
#include <stdio.h>
#include <string.h>

int main (void)
{
    char nomefile[13];
    FILE *fp;
    int caratteri = 0;
    int linee = 0;
    int parole = 0;
    char ch, prec = ' ';

    printf("Immetti il nome del file: ");
    scanf("%12s", nomefile);

    if ((fp = fopen(nomefile, "r")) == NULL) {
        printf("Errore in apertura in lettura del file %s!\n",
            nomefile);
        exit(1);
    }

    ...
}
```

21

Esercizio 3 - Soluzione (2)

```
...
while (fscanf(fp, "%c", &ch) == 1)
{
    caratteri++;
    if (ch == '\n')
        linee++;
    if (isspace(prec) && !isspace(ch))
        parole++;
    prec = ch;
}

fclose(fp);

printf("Il numero di caratteri e' %d.\n",
    caratteri);
printf("Il numero di parole e' %d.\n", parole);
printf("Il numero di linee e' %d.\n", linee);

return 0;
}
```

22