Esercizi 3AIIN 26.03.2021

- Salva in Moodle un file zip che contiene solo i pgm sorgenti.
- Formato nome file: **CognomeNomeGGMMAAAA.estensione**

dove GGMMAAAA è la data di assegnazione del lavoro.

- Terminata ogni lezione di laboratorio devi caricare in Moodle il lavoro svolto.
- A casa eventualmente puoi finirlo e/o correggerlo. Hai tempo una settimana.
- Anche chi è assente il giorno della consegna è tenuto a svolgere il lavoro assegnato.
- Verranno fatti controlli a campione e sicuramente durante le interrogazioni.

Esercizio n.1 VETTORI – STRINGHE - RECORD - FUNZIONI

Dato un vettore di record contenente informazioni su pittori avente la seguente struttura:

```
#define MAX_PITTORI 3
#define DIM_STR_NOME 20+1
#define DIM_STR_GENERE 15+1

typedef struct{
      int giorno;        //1<=giorno<=31
      int mese;          //1<=mese<=12
      int anno;          //anno > 1900
} data;

typedef struct{
       char nomePittore[DIM_STR_NOME];        //chiave primaria
       char genere[DIM_STR_GENERE];           //es.: paesaggista, ritrattista,
…
       data dataNascita;
}pittore;
```

L'attributo genere è una chiave secondaria.

Nel **main()**:

```
pittore artisti[MAX_PITTORI];
int numPittori = 0;
```

RICORDA 1

L'attributo nomePittore essendo **chiave primaria** identifica univocamente un pittore nell'insieme dei pittori (vettore di record), cioè non esistono due pittori che hanno lo stesso nome.

Una **chiave secondaria** può invece identificare più di un record nel vettore. Nel nostro caso ci possono essere più pittori appartenenti allo stesso genere.

RICORDA 2

1) Una volta individuati i sottopgm, determina i parametri, determina poi la loro direzionalità (I, O, I/O) (vedi slide) e infine stabilisci le modalità di passaggio dei parametri (vedi slide).
2) Ogni sottopgm che svolgere un'unica funzione logica (cioè fare un'unica cosa).
3) Ogni sottoprogramma non deve contenere istruzioni di I/O, salvo casi particolari (es. inserimento/stampa di dati in/da una struttura dati).

Scrivere un pgm C++ che esegua interattivamente (si utilizzi un menù) le seguenti funzioni:

1) inserimento (in coda) di un pittore nella struttura dati **insertPittore()**. La procedura inserimento utilizza una procedura **leggiPittore()** che legge i dati di un pittore e restituisce (return) il pittore letto. Non controllare che il campo nomePittore (lo leggi nel main()) sia chiave primaria, lo supponi vero. Utilizza una funzione **controllaData()** che controlla la data di nascita (solo questi controlli: 0<=giorno<=31, 1<=mese<=12, anno>1900). La funzione insertPittore() ritorna 1 se l'inserimento è andato bene, 0 altrimenti.
2) stampa i dati di tutti i pittori **stampaPittori()**. Per ogni pittore la procedura richiama un'altra procedura che stampa i dati del pittore di un pittore (**stampa Pittore()**).
3) ricerca di un pittore per nome (la funzione **ricercaPittore()** ritorna l'indice se presente, -1 se non presente). Nel main() poi attraverso l'indice stampi le informazioni sul pittore (usa stampaPittore() che trovi descritta più avanti). Questa è una ricerca per chiave primaria.
4) cancellazione (fisica) di un pittore: l'utente, nel main(), inserisce il nome del pittore (chiave primaria). Il sottopgm **cancellaPittore()** usa il sottogm **ricercaPittore()**. Prima di cancellare visualizza i dati del pittore (usa **stampaPittore()** e chiedi conferma). La procedura cancellaPittore() ritorna 1 se la cancellazione è stata effettuata, 0 altrimenti.
5) ricerca per genere (lo leggi nel main(), è una chiave di ricerca secondaria): **ricercaGenere()**. Al suo interno per ogni pittore trovato richiama stampaPittore(). Se non ci sono pittori del genere richiesto visualizza un opportuno messaggio.
6) stampa ordinata in modo crescente dei pittori per nome: **stampaOrdinataNome()** (usa: **sortNome()** + stampaPittori()).

```
int menu() {
int scelta;

	cout<<"GESTIONE PITTORI: "<<endl<<endl;

	cout<<"1) Inserimento di un pittore: "<<endl;
	cout<<"2) Stampa i dati di tutti i pittori: "<<endl;
	cout<<"3) Ricerca di un pittore per nome: "<<endl;
	cout<<"4) Cancellazione di un pittore per nome: "<<endl;
	cout<<"5) Ricerca dei pittori per genere: "<<endl;
	cout<<"6) Stampa ordinata dei pittori per nome: "<<endl;
	cout<<"0) Fine lavoro "<<endl<<endl;

	cout<<"Digita la funzione scelta: ";
	cin>> scelta;

	return scelta;
}
```

Esercizio n.2 Dichiarazione di tipi e definizione di variabili

Fai una o più dichiarazioni di *tipi* (typedef, struct, enum) per rappresentare la seguente realtà:

- Un **edificio** ha un massimo 20 piani. Ogni piano contiene massimo 40 uffici. Prevedi la possibilità di avere un numero di piani e di uffici inferiore alle dimensioni massime. Tutti i piani hanno lo stesso numero di uffici.
- Ogni **ufficio** ha un'**esposizione**: NORD, NORDEST, EST, SUDEST, SUD, … e un impiegato.
- Ogni **impiegato** ha un nome (stringa di 25 caratteri), un cognome (stringa di 30 caratteri), una categoria (numero intero) e uno stipendio (numero reale).

Usa le costanti.

Definisci poi una *variabile* **torre** di tipo edificio.

Scrivi poi un'istruzione C++ cha assegna il tuo cognome all'impiegato che si trova al piano n.10 nell'ufficio n.5.