const int k=3;

const char NOMEFILE[]="file.txt";

**struct aereo\_t**{

char codice[11];

bool atterrato;

};

**struct aeroporto\_t**{

int M;

aereo\_t (\*piste)[k];

};

**void inizializza\_Aeroporto(aeroporto\_t &p, int M){**

if(p.M!=0)

delete[] p.piste;

p.M=M;

p.piste = new aereo\_t[p.M][k];

for(int i=0;i<p.M;i++)

for(int j=0;j<k;j++)

p.piste[i][j].atterrato=false;

}

**void salva\_stato\_piste(aeroporto\_t &p, bool file, ostream &stream){**

if (file){

stream<<p.M<<endl;

}

for(int i=0;i<p.M;i++){

stream<<i<<" - ";

for(int j=0;j<k;j++)

if(p.piste[i][j].atterrato==false)

stream<<". ";

else

stream<<p.piste[i][j].codice<<" ";

stream<<endl;

}

stream<<endl;

}

**bool salva\_stato(aeroporto\_t &p**){

for(int i=0;i<p.M;i++){;

for(int j=0;j<k;j++)

if(p.piste[i][j].atterrato==false)

f<<". ";

else

f<<p.piste[i][j].codice<<" ";

f<<endl;

}

f.close();

return true;

}

**bool carica\_stato(aeroporto\_t &p**){

ifstream f(NOMEFILE);

if(!f)

return false;

int M;

f>>M;

inizializza\_Aeroporto(p,M);

char trattino, indice;

char tmp[11];

for(int i=0;i<p.M;i++){

f>>indice>>trattino;

for(int j=0;j<k;j++){

f>>tmp;

if(strcmp(tmp,".")==0){

p.piste[i][j].atterrato=false;

p.piste[i][j].codice[0]='\0';

}

else{

p.piste[i][j].atterrato=true;

strcpy(p.piste[i][j].codice, tmp);

}

}

}

f.close();

return true;

}

**bool salva\_stato(aereoporto\_t &p){**

ofstream f(NOMEFILE);

if(!f)

return false;

int tot=p.M\*k;

f.write(reinterpret\_cast<const char \*>(&p.M), sizeof(int));

f.write(reinterpret\_cast<const char \*>(p.porto), sizeof(aereoporto\_t)\*tot);

f.close();

return true;

}

**bool carica\_stato(aereoporto\_t &p){**

ifstream f(NOMEFILE);

if(!f)

return false;

int M;

f.read(reinterpret\_cast<char \*>(&M), sizeof(int));

inizializza(M, p);

int tot=p.M\*k;

f.read(reinterpret\_cast<char \*>(p.porto), sizeof(aereoporto\_t)\*tot);

f.close();

return true;

}

*const char nomeFile[] = "lotteria.txt";*

*const int lunghezzaNome = 15;*

***struct Biglietto {***

*char nome[lunghezzaNome];*

*int codice;*

*};*

***struct Lotteria*** *{*

*int numeroBiglietti;*

*int maxDim;*

*Biglietto \* biglietti;*

*};*

***void initLotteria(Lotteria &l){***

*l.numeroBiglietti = 0;*

*l.maxDim = 1;*

*l.biglietti = new Biglietto[l.maxDim];*

*}*

***void aumentaBigliettiMax(Lotteria &l){***

*int dim = l.maxDim;*

*Biglietto \* tmpArray = new Biglietto[dim];*

*for (int i=0; i<dim; i++)*

*tmpArray[i] = l.biglietti[i];*

*delete[] l.biglietti;*

*l.maxDim \*= 2;*

*l.biglietti = new Biglietto[l.maxDim];*

*for (int i=0; i<dim; i++)*

*l.biglietti[i] = tmpArray[i];*

*delete[] tmpArray;*

*}*

***bool codicePresente(Lotteria &l, int c****){*

*//trova se il codice è già presente*

*for(int i=0; i<l.numeroBiglietti; i++){*

*if(l.biglietti[i].codice == c)*

*return true;*

*}*

*return false;*

*}*

***bool vendi\_biglietto(Lotteria &l, char nome[], int c****){*

*//controllo sul raggiungimento della dimensione massima*

*if(l.numeroBiglietti == l.maxDim)*

*aumentaBigliettiMax(l);*

*if(!codicePresente(l, c)){*

*l.biglietti[l.numeroBiglietti].codice = c;*

*strcpy(l.biglietti[l.numeroBiglietti].nome, nome);*

*l.numeroBiglietti++;*

*return true;*

*}*

*else*

*return false;*

*}*

***void scrivi\_biglietti(Lotteria &l, ostream &stream, bool file){***

*if(file){*

*stream << l.numeroBiglietti << endl;*

*stream << l.maxDim << endl;*

*}*

*for(int i=0; i<l.numeroBiglietti; i++){*

*stream << l.biglietti[i].nome << '\t' << l.biglietti[i].codice << endl;*

*}*

*}*

***void carica\_biglietti(Lotteria &l){***

*ifstream f(nomeFile);*

*delete[] l.biglietti;*

*f >> l.numeroBiglietti;*

*f >> l.maxDim;*

*l.biglietti = new Biglietto[l.maxDim];*

*for(int i=0; i<l.numeroBiglietti; i++){*

*f>>l.biglietti[i].nome;*

*f>>l.biglietti[i].codice;*

*}*

*f.close();*

*}*

*int main()*

*{*

*Lotteria lot;*

*initLotteria(lot);*

*srand(time(0));*

*case 1:*

*char nome[lunghezzaNome];*

*int c;*

*cout << "nome: ";*

*cin >> nome;*

*cout << "codice: ";*

*cin >> c;*

*vendi\_biglietto(lot, nome, c);*

*/\**

*if(vendi\_biglietto(lot, nome, c))*

*cout << "venuduto\n";*

*else*

*cout << "codice già presente\n";*

*\*/*

*break;*

*case 2:*

*scrivi\_biglietti(lot, cout, false);*

*break;*

*case 3:*

*{*

*ofstream f(nomeFile);*

*if(!f)*

*cerr << "Errore nell'apertura del file\n";*

*scrivi\_biglietti(lot, f, true);*

*f.close();*

*}*

*break;*

*const int MAX\_L = 10;*

***struct coda****{*

*int tot\_ele = 0, primo = 0, ultimo = 0;*

*int v[MAX\_L];*

*};*

***void inizializza\_coda(coda &c){***

*for(int i = 0; i < MAX\_L; i++)*

*c.v[i] = 0;*

*}*

***void stampa\_coda(coda &c){***

*if(c.tot\_ele == 0)*

*return;*

*if(c.primo <= c.ultimo)*

*for(int i = c.primo; i <= c.ultimo; i++)*

*cout<<c.v[i]<<" ";*

*else{*

*for(int i = c.primo; i < MAX\_L; i++)*

*cout<<c.v[i]<<" ";*

*for(int i = 0; i <= c.ultimo; i++)*

*cout<<c.v[i]<<" ";*

*}*

*cout<<"Primo: "<<c.primo<<endl;*

*cout<<"Ultimo: "<<c.ultimo<<endl;*

*cout<<"Elementi totali: "<<c.tot\_ele<<endl;*

*cout<<endl;*

*/\**

*for(int i = 0; i < MAX\_L; i++){*

*cout<<c.v[i]<<" ";*

*}*

*cout<<endl;*

*cout<<"Primo: "<<c.primo<<endl;*

*cout<<"Ultimo: "<<c.ultimo<<endl;*

*cout<<"Elementi totali: "<<c.tot\_ele<<endl;*

*\*/*

*}*

***bool inserisci\_testa(coda &c, int elem){***

*if(c.tot\_ele == 0){*

*c.v[0] = elem;*

*c.tot\_ele++;*

*return true;*

*}*

*else if(c.tot\_ele != MAX\_L){*

*if(c.primo == 0)*

*c.primo = MAX\_L - 1;*

*else*

*c.primo = c.primo - 1;*

*c.v[c.primo] = elem;*

*c.tot\_ele++;*

*return true;*

*}*

*return false;*

*}*

***bool inserisci\_coda(coda &c, int elem){***

*if(c.tot\_ele == 0){*

*c.v[0] = elem;*

*c.tot\_ele++;*

*return true;*

*}*

*else if(c.tot\_ele != MAX\_L){*

*c.ultimo = (c.ultimo+1) % MAX\_L;*

*c.v[c.ultimo] = elem;*

*c.tot\_ele++;*

*return true;*

*}*

*return false;*

*}*

***int estrai\_testa(coda &c){***

*if(c.primo == (MAX\_L - 1)){*

*c.primo = 0;*

*c.tot\_ele--;*

*if(c.tot\_ele == 0)*

*c.primo = c.ultimo = 0;*

*return c.v[MAX\_L - 1];*

*}*

*else if(c.primo == 0 && c.tot\_ele == 1){*

*c.tot\_ele--;*

*if(c.tot\_ele == 0)*

*c.primo = c.ultimo = 0;*

*return c.v[c.primo];*

*}*

*c.tot\_ele--;*

*c.primo++;*

*return c.v[c.primo-1];*

*}*

*/\**

*Includendo il file di intestazione <ctype>, si posso utilizzare una*

*serie di funzioni che operano sui caratteri. Tali funzioni prendono*

*in ingresso un valore intero, e ritornano un valore intero.*

*Il valore preso in ingresso è interpretato come il codice ASCII di un*

*carattere (in ogni caso si puo passare a tali funzioni anche un valore di tipo*

*char in ingresso, perchè, come vedremo in seguito, tale valore viene*

*convertito implicitamente nel tipo int). A seconda della specifica funzione,*

*il valore di ritorno può rappresentare il codice ASCII di un carattere,*

*oppure un valore logico (0 per falso, 1 per vero). Alcune di queste*

*funzioni sono:*

*\*/*

*// Ritorna un valore diverso da 0 se c e' alfanumerico, 0 altrimenti*

***int isalnum(int c) ;***

*// Ritorna un valore diverso da 0 se c e' alfabetico, 0 altrimenti*

***int isalpha(int c) ;***

*// Ritorna un valore diverso da 0 se c e' una cifra decimale, 0 altrimenti*

***int isdigit(int c) ;***

*// Ritorna un valore diverso da 0 se c e' una cifra esadecimale, 0 altrimenti*

***int isxdigit(int c) ;***

*// Ritorna un valore diverso da 0 se c e' una lettera minuscola, 0 altrimenti*

***int islower(int c) ;***

*// Ritorna un valore diverso da 0 se c e' una lettera maiuscola, 0 altrimenti*

***int isupper(int c) ;***

*// Se c è la codifica di una lettera maiuscola, restituisce la codifica della*

*// corrispondente lettera minuscola, altrimenti ritorna c*

***int tolower(int c) ;***

*// Se c è la codifica di una lettera minuscola, restituisce la codifica della*

*// corrispondente lettera maiuscola, altrimenti ritorna c*

***int toupper(int c) ;***

***RANDOM***

srand(time(0));

rand() %(max-min+1)+min;

Immagine che contiene testo

Descrizione generata automaticamente

Immagine che contiene testo

Descrizione generata automaticamente

Immagine che contiene testo

Descrizione generata automaticamente

Immagine che contiene testo

Descrizione generata automaticamente

***&*** *non si usa per i vettori*