```
******************Rand******************
#include <ctime>
srand(time(0));
rand()%(max-min+1)+min;
******************Funzioni char******************
// 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) ;
******************Funzioni stringhe******************
#include <cstring>
strcpy(stringa1, stringa2)      //copia il contenuto di stringa2 in stringa 1
strncpy(stringa1, stringa2, n)  // copia i primi n caratteri di stringa2 in stringa 1
strcat(stringa1, stringa2)      // concatena il contenuto di stringa2 a stringa1
strcmp(stringa1, stringa2)      // confronta stringa2 con stringa1: 0 (uguali), >0 (stringa1 è maggiore di stringa2), <0 (viceversa)
strlen(stringa)                 //ritorna lunghezza stringa, si ferma al '\0'
```

```
******************Aeroporto******************
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;

const int K = 3;
int num_piste;
const char NOMEFILE[] = "aeroporto.dat";

struct aereo{
    char id[11];
};

struct pista{
    int num_aerei = 0;
    aereo vett_aerei[K];
};

pista* Inizializza_Aeroporto(int M){
    if(M>=0){
        pista *piste = new pista[M];
        for(int i=0; i<M; i++){
            for(int j=0; j<K; j++){
                piste[i].vett_aerei[j].id[0] = '.';
                piste[i].vett_aerei[j].id[1] = '\0';
            }
        }
        return piste;
    }
    return 0;
}

bool Aggiungi_aereo_a_pista(char ID_Aereo[], int NR_pista, pista* piste){
    if(piste[NR_pista].num_aerei == K){
        for(int i=0; i<num_piste; i++){
            if(piste[i].num_aerei < K){
                for(int j=0; j<K; j++){
                    if(piste[i].vett_aerei[j].id[0] == '.'){
                        strcpy(piste[i].vett_aerei[j].id, ID_Aereo);
                        piste[i].num_aerei++;
                        return true;
                    }
                }
                return false;
            }
        }
    }
    for(int j=0; j<K; j++){
        if(piste[NR_pista].vett_aerei[j].id[0] == '.'){
            strcpy(piste[NR_pista].vett_aerei[j].id, ID_Aereo);
            piste[NR_pista].num_aerei++;
            return true;
        }
    }
    return false;
}

void Stampa_stato_piste(pista* piste){
    for(int i=0; i<num_piste; i++){
        cout << i << " - ";
        for(int j=0; j<K; j++){
            if(piste[i].vett_aerei[j].id[0] == '.')
                cout << '.';
            else
                cout << piste[i].vett_aerei[j].id;
            cout << " ";
        }
        cout << endl;
    }
```

```
bool Rimuovi_aereo(char ID_Aereo[], pista* piste){
    if(strlen(ID_Aereo) <= 0)
        return false;
    for(int i=0; i<num_piste; i++){
        for(int j=0; j<K; j++){
            if(!strcmp(piste[i].vett_aerei[j].id, ID_Aereo)){
                piste[i].vett_aerei[j].id[0] = '.';
                piste[i].num_aerei--;
                return true;
            }
        }
    }
    return false;
}

bool Salva_stato(pista* piste){
    ofstream f1(NOMEFILE);
    if(!f1)
        return false;


    f1 << num_piste << endl;
    for(int i=0; i<num_piste; i++){
        f1 << piste[i].num_aerei << endl;              //CON FILE DI TESTO
        for(int j=0; j<K; j++){
            f1 << piste[i].vett_aerei[j].id << endl;
        }
    }


    f1.put(num_piste);
    f1.write(reinterpret_cast<char *>(piste), sizeof(pista)*num_piste*3);       //CON FILE BINARIO
    f1.close();
    return true;
}

pista* Carica_stato(pista* piste){
    ifstream f2(NOMEFILE);
    if(!f2)
        return piste;


    f2 >> num_piste;
    delete[] piste;
    piste = new pista[num_piste];
    for(int i=0; i<num_piste; i++){                    //CON FILE DI TESTO
        f2 >> piste[i].num_aerei;
        for(int j=0; j<K; j++){
            f2 >> piste[i].vett_aerei[j].id;
        }
    }


    num_piste = f2.get();
    delete[] piste;
    piste = new pista[num_piste];
    f2.read(reinterpret_cast<char *>(piste), sizeof(pista)*num_piste*3);   //CON FILE BINARIO

    f2.close();
    return piste;
}

bool Compatta_piste(pista* piste){
    for(int i=0; i<num_piste; i++){
        if(piste[i].num_aerei != 3){
            for(int j=0; j<K; j++){
                if(j+1 <= 2 && piste[i].vett_aerei[j].id[0] == '.' && piste[i].vett_aerei[j+1].id[0] != '.'){
                    strcpy(piste[i].vett_aerei[j].id, piste[i].vett_aerei[j+1].id);
                    piste[i].vett_aerei[j+1].id[0] = '.';
                    piste[i].vett_aerei[j+1].id[1] = '\0';
                }
            }
        }
    }
    return true;
}
```