```
// exercicio 3
#include <iostream>
using namespace std;
#define TMAX 20
typedef struct{
    string nome;
    float media, notas[4];
} Aluno;

int leiaNLimSup(int);
void leituraReg(Aluno &); // por referencia direta
void leituraVetReg(int, Aluno []);
void calculaMediaAluno(Aluno &); // por referencia direta
void calculaMediaTurma(int, Aluno []);
void mostraReg(Aluno);
void mostraVetReg(int, Aluno []);

int main(){
    Aluno turma[TMAX];
    int n = leiaNLimSup(TMAX);
    leituraVetReg(n, turma);
    calculaMediaTurma(n, turma);
    mostraVetReg(n, turma);
    return 0;
}
int leiaNLimSup (int limSup){
    int n;
    do{
        cout << "Qtde (max" << limSup << ") : " ;
        cin>>n;
    }while(n<=0 or n>limSup);
    return n;
}
void leituraReg(Aluno &alguem){  // por referencia
    cout<<"Dados do aluno - Nome:";
    getline(cin,alguem.nome);
    alguem.media = 0;
    for (int j=0; j<4; j++){
        do{
            cout << j+1<< "a nota: ";
            cin >> alguem.notas[j];
        }while(alguem.notas[j]<0 or alguem.notas[j]>10);
    }
}
void leituraVetReg(int n, Aluno osAlunos[]){
    for (int i=0; i<n; i++){
        cin.ignore();
        leituraReg(osAlunos[i]);
    }
}
void calculaMediaAluno(Aluno &alguem){ // por referencia
    for (int j=0; j<3; j++)
        alguem.media += alguem.notas[j];
    alguem.media = alguem.media/4;
}
void calculaMediaTurma(int n, Aluno osAlunos[]){
    for (int i=0; i<n; i++)
        calculaMediaAluno(osAlunos[i]);
}
void mostraReg(Aluno alguem){
```

```
    cout<<alguem.nome<< " - " << alguem.media<< endl;
}
void mostraVetReg(int n, Aluno osAlunos[]){
    cout<<"DADOS DOS ALUNOS: Nome - Media"<<endl;
    for (int i=0; i<n; i++)
        mostraReg(osAlunos[i]);
}

// exercicio 4
#include <iostream>
#include <iomanip>
using namespace std;
typedef struct{
    string matr, nome, cargo, depto, dtAdm;
    float salario;
} Funcionario;

char leiaCaracterSN();
void leituraFuncionario(int, Funcionario []); // por posicionamento
void mostraFuncionario(Funcionario);
void relatorio(int, Funcionario []);
float calcMediaSalarial(int, Funcionario []);

int main(){
    Funcionario empresa[50];
    int i=0;
    char resp;
    cout<<"EMPRESA XXX"<<endl;
    do{
        leituraFuncionario(i, empresa);
        i++;
        resp = leiaCaracterSN();
    }while(resp=='S');
    relatorio(i, empresa);
    return 0;
}
char leiaCaracterSN(){  // fcao pra ler(e retornar) um char S ou N
    char c;
    do{
        cout<<"Outro? S/N";
        c = toupper(cin.get());
    }while(c!='S' and c!='N');
    cin.ignore();
    return c;
}
void leituraFuncionario(int i, Funcionario vet[]){
    cout<<"Dados do Funcionario - Nome:";
    getline(cin,vet[i].nome);
    cout<<"Matricula:";
    getline(cin,vet[i].matr);
    cout<<"Cargo:";
    getline(cin,vet[i].cargo);
    cout<<"Depto:";
    getline(cin,vet[i].depto);
    do{
        cout<<"Salario:";
        cin>>vet[i].salario;
    }while(vet[i].salario<=0);
    cin.ignore();
}
void mostraFunc(Funcionario reg){
```

```
    cout<<"Nome "<< reg.nome << ", "<< reg.cargo <<" - Salario "<< reg.salario
<< endl;
}
float calcMediaSalarial(int n, Funcionario vet[]){
    float media=0;
    for(int i=0; i<n; i++)
        media+=vet[i].salario;
    return media/n;
}
void relatorio(int n, Funcionario vet[]){
    float media = calcMediaSalarial(n,vet);
    cout<<fixed<<setprecision(2)<<endl;
    cout<<"Relatorio - salarios acima media salarial "<< media<<endl;
    for(int i=0; i<n; i++)
        if(vet[i].salario>media)
            mostraFunc(vet[i]);
}

// exercicio 5
#include <iostream>
using namespace std;
#define TMAX 100
typedef struct{
    int codigo;
    string nome;
} TProfissao;

int leiaNLimSup(int);
void leituraReg(TProfissao &);
void leituraVetReg(int, TProfissao []);
string pegaNomeProf(int, TProfissao [], int);

int main(){
    TProfissao profissao[TMAX];
    string nomeprof;
    int codDesejado, n = leiaNLimSup(TMAX);
    leituraVetReg(n, profissao);
    do{
        cout<<"Digite o codigo a pesquisar (-1 sair)"<<endl;
        cin>>codDesejado;
        if(codDesejado > -1){
            nomeprof = pegaNomeProf(n,profissao,codDesejado);
            cout<<"Nome da profissao: "<< nomeprof<< endl;
        }
    }while(codDesejado>=0);
    return 0;
}
int leiaNLimSup (int limSup){
    int n;
    do{
        cout << "Qtde (max" << limSup << ") : " ;
        cin>>n;
    }while(n<=0 or n>limSup);
    return n;
}
void leituraReg(TProfissao &prof){
    cout<<"Dados da profissao - Codigo:";
    cin>> prof.codigo;
    cin.ignore();
    cout<<"Nome:";
    getline(cin,prof.nome);
```

```
}
void leituraVetReg(int n, TProfissao vetProf[]){
    for (int i=0; i<n; i++)
        leituraReg(vetProf[i]);
}
string pegaNomeProf(int n, TProfissao profissao[], int cod){
    int i=0;
    for (int i=0; i<n; i++)
        if(profissao[i].codigo == cod)
            return profissao[i].nome;
    return "nao encontrado"; // caso nao exista, manda esta msg
}

// exercicio 6
#include <iostream>
using namespace std;
#define TMAX 50
typedef struct{
          string nome, dtNasc;
} Dependente;
typedef struct{
          string matr, nome;
          int ndep;
          Dependente depend[5]; //forte relacao de dependencia
} Socio;

char leiaCaracterSN();
int leiaNIntervalo(int, int);
void leituraSocio(Socio &);
void leituraVetDep(int, Dependente []);
void leituraDep(Dependente &);
void mostraSocio(Socio);
void mostraDep(Dependente);
void mostraVetDep(int, Dependente []);
void relatorio(int, Socio []);

int main(){
    Socio clube[TMAX];
    int i=0;
    char resp;
    do{
        leituraSocio(clube[i]);
        i++;
        resp = leiaCaracterSN();
    }while(resp=='S');
    relatorio(i, clube);
    return 0;
}
// Outra solução: criar 2 tipos estruturados independentes
// usando 2 vetores, um de socios e outro de dependentes

int leiaNIntervalo(int limInf, int limSup){
    int n;
    do{
        cout << "Qtde (max" << limSup << ") : " ;
        cin>>n;
    }while(n<limInf or n>limSup);
    return n;
}
char leiaCaracterSN(){ // fcao pra ler(e retornar) um char S ou N
    char c;
```

```
    do{
        cout<<"Outro? S/N";
        c = toupper(cin.get());
        cin.ignore();
    }while(c!='S' and c!='N');
    return c;
}
void leituraSocio(Socio &reg){
    cout<<"Dados do socio - Nome:";
    getline(cin,reg.nome);
    cout<<"Matricula:";
    getline(cin,reg.matr);
    cout<<"Dependentes ";
    reg.ndep = leiaNIntervalo(0,5);
    if(reg.ndep>0) { // se nao tiver depend nem entra
        cin.ignore();
        leituraVetDep(reg.ndep,reg.depend);
    }
}
void leituraVetDep(int n, Dependente vet[]){
    for(int i=0; i<n; i++)
        leituraDep(vet[i]);
}
void leituraDep(Dependente &reg){
    cout<<"Dados do dependente - Nome:";
    getline(cin,reg.nome);
    cout<<"Nascimento(dd/mm/aa):";
    getline(cin,reg.dtNasc);
}
void mostraDep(Dependente p){
    cout<<"Nome "<< p.nome << endl;
}
void mostraVetDep (int n, Dependente vet[]){
    for(int i=0; i<n; i++)
        mostraDep(vet[i]);
}
void mostraSocio(Socio p){
    cout<<"Nome "<< p.nome << endl;
    if(p.ndep>0){
        cout<<"Dependentes "<< endl;
            mostraVetDep(p.ndep, p.depend);
    }
}
void relatorio(int n, Socio vet[]){
   cout<<"Relatorio - socios com +3 dependentes "<< endl;
    for(int i=0; i<n; i++)
        if(vet[i].ndep>3)
          mostraSocio (vet[i]);
}
```