

**UNIVERSIDADE FEDERAL DE VIÇOSA
CAMPUS FLORESTAL**

SAMUEL PEDRO CAMPOS SENA - 3494

SAULO MIRANDA SILVA - 3475

**PROGRAMAÇÃO ORIENTADA A OBJETOS
TRABALHO PRÁTICO**

FLORESTAL

2019

SUMÁRIO

1 INTRODUÇÃO -----	3
2 DESENVOLVIMENTO -----	4
3 CONCLUSÃO -----	8

INTRODUÇÃO

O objetivo do trabalho apresentado é elaborar um sistema de gestão acadêmica no qual é possível adicionar professores, alunos, notas e criar turmas. Outros requisitos do sistema são modularidade e funcionamento contínuo mesmo se houver entrada diferente da esperada (inválida),

Para compilar utilizamos makefile, este trabalho foi desenvolvido e testado em distribuições GNU/Linux, não garantimos que em outros sistemas operacionais obterá o mesmo resultado de execução.

Para compilar, abra uma janela do terminal devidamente navegado até a pasta em que se localiza o arquivo makefile e execute para compilar:

```
$ make
```

E para executar o algoritmo:

```
$ make run
```

Caso o usuário deseje compilar o programa manualmente, basta executar o seguinte comando em uma janela do terminal devidamente navegado até a pasta onde se encontra o arquivo “main.cpp”:

```
$ g++ main.cpp -o EXEC Sources/Professor.cpp Sources/Aluno.cpp  
Sources/Turma.cpp Sources/Notas.cpp Sources/Escola.cpp
```

Em seguida para executar o programa:

```
$ ./EXEC
```

A utilização do programa é uma tarefa bastante intuitiva, sem grandes dificuldades e avisos úteis que informam o usuário o que acontece durante toda a execução.

DESENVOLVIMENTO

Classe Aluno:

Essa classe contém os atributos do aluno(Nome, Matrícula, Nome do pai, Nome da mãe, Data de nascimento e Endereço), além dos métodos para alterar e acessar tais atributos.

```
class Aluno{
private:
    string Nome;
    int N_Matricula;
    string NomePai;
    string NomeMae;
    string DataNasc;
    string Endereco;
public:
    Aluno(string Nome); //Matriculas serao incrementadas//
    //virtual ~Aluno();
    void Set_DataNasc(string DataNasc);
    void Set_NumMatricula(int NumMatricula);
    void Set_NomePais(string nomepai,string nomemae);
    void Set_Endereco(string endereco);
    void ModificaNome(string Nome);
    int GetMatricula();
    string Get_NomeMae();
    string Get_NomePai();
    string Get_Nome();
    string Get_Endereco();
    string Get_DataNasc();
};
```

Classe Notas:

Possui um vetor com as notas do aluno.

```
class Notas{
private:
    std::vector<double> NotasAlunos;
public:
    Notas();
    void AdicionaNota(double Nota);
    void ModificaNota(double NovaNota,int indexNota);
    void ListaNotas();
    void ImprimeMediaNotas();
};
```

A media das notas do aluno é calculado através da soma de todas as notas cadastrada (do mesmo aluno), dividido pela quantidade de notas. Caso esse resultado seja maior ou igual a 60, é impresso na tela que o aluno encontra-se aprovado, caso contrario é impresso que encontra-se reprovado.

Classe Professor:

Contém as informações do professor(Nome, Endereço, Área e Salário por Hora) e métodos para acessá-las e alterá-las.

```
class Professor{
private:
    string Nome;
    string Endereco;
    string Area;
    double SalarioPHora;
public:
    Professor(string Nome);
    void Set_Area(string area);
    void Set_Endereco(string endereco);
    bool Set_Salario(double Salario);
    void ModificaNome(string Nome);
    string Get_Nome();
    string Get_Endereco();
    string Get_Area();
    double Get_Salario();
};
```

Classe Turma:

A Turma possui os seguintes atributos:

Professor responsável(ponteiro para professor);

Vetor de ponteiros para objetos Alunos;

Vetor de Notas de Alunos(Objetos Notas);

Ano;

Código;

Nessa classe ficam salvas as notas dos alunos, separados dos objetos alunos, para manter a coesão dos módulos. O índice do aluno no vetor de ponteiros do tipo Aluno é o mesmo que o índice da nota desse aluno no vetor de notas, dessa forma a nota no índice “i” pertence ao aluno no índice “i”.

```
class Turma{
private:
    Professor* ProfessorResponsavel;
    std::vector<Aluno*> Alunos;
    std::vector<Notas> NotasAlunos;
    int Ano;
    int Codigo;
    friend class Escola;
public:
    Turma(Professor* profResp, int Codigo, int Ano);
    void AdicionaAluno(Aluno* aluno);
    void AdicionaNotas();
    bool ImprimeAlunos();
    void ImprimeProfessor();
    int GetAno();
    int GetCodigo();
    string GetProfessor();
    void Imprime_Boletim(int indexAluno);
    bool AlunoPresente(int Matricula);
};
```

Aqui fizemos tratamento para redundância, então não é possível entrar com o mesmo aluno na mesma turma mais de uma vez.

Classe Escola:

A Escola foi definida como a Visão do nosso programa, contendo as saídas e entradas necessárias na interação do usuário com a máquina.

Ela possui os seguintes atributos:

CódigoTurma;
ContMatricula;
Vetor de Professores;
Vetor de Alunos;
Vetor de Turmas;

```
class Escola{
private:
    int ContMatricula;
    int CodigoTurma;
    std::vector<Professor> Professores;
    std::vector<Aluno> Alunos;
    std::vector<Turma> Turmas;
public:
    Escola();
    void AdicionaProfessor();
    void AdicionaAluno();
    void CriaTurma();
    void AdicionaAlunoEmTurma();
    void AdicionaNota();
    void ImprimeBoletim();
    void ImprimeProfessores();
    void ImprimeAlunos();
    void ImprimeTurmas();
    void ImprimeAlunosDeTurma();
    void ModificaProfessor();
    void ModificaAluno();
};

int Menu_Principal();
int Menu_Aluno();
int Menu_Professor();
int Menu_Turma();
int Menu_Finalizar();
```

Os atributos `CódigoTurma` e `ContMatricula` são inicializados com os valores 100 e 3000, respectivamente, e são utilizados para a devida numeração dos códigos de turma e valores de matrículas dos alunos. A cada nova turma ou novo aluno cadastrado, o valor de `CódigoTurma` ou `ContMatricula` é atribuído ao novo cadastro e seu valor é incrementado (para um próximo cadastro receber o novo valor).

Aqui realizamos uma espécie de tratamento para quando a entrada for um tipo numérico e por erro do usuário, ele entrar com algum carácter alfabético, o programa não apresentará erro e continuará a execução.

Conclusão

O respectivo trabalho foi de suma importância para o aprendizado do uso e domínio da linguagem C++ , devido aos desafios da tarefa além de possibilitar que uma melhor compreensão fosse possível, tornou necessário aprender bastante sobre a linguagem e suas características.

Agradecimentos ao professor Fabrício pela oportunidade e ensinamentos que foram necessários para a execução do trabalho.