Pontifícia Universidade Católica de Minas Gerais / Instituto Politécnico / Instituto de Informática
Curso de Engenharia Mecânica – ênfase em Mecatrônica
Disciplina: Técnicas de Programação para Engenharia
Introdução a arquivos utilizando C++
Os seguintes exemplos foram retirados do livro Treinamento em Linguagem C++
Modulo 2 de Victorine VIVIANE Mizrahi

1. Criar um arquivo texto e gravar uma string.

Digite o seguinte programa

```
// OFLSTR.CPP

// Grava strings em arquivo

# include <fstream.h> // para funções de arquivo

void main()

{ ofstream fout (“teste.txt”); // cria arquivo para gravação em modo texto

fout << “Um grande antídoto contra o egoísmo \n”;

fout << “é a generosidade ... Dê, mesmo que \n “;

fout << “isso requeira de você um esforço \n”;

fout << “consciente. Pelo fato de partilhar \n”;

fout << “tudo o que possui, seu egoísmo se \n “;

fout << “abrandará.\n”;

}
```

Neste programa definimos um objeto chamado fout da classe ofstream.

Inicializamos esse objeto com o nome do arquivo TESTE.TXT. Esta inicialização associa o objeto fout ao arquivo em disco TESTE.TXT para gravação.

Para verificar se o arquivo foi gravado corretamente, você pode utilizar o comando TYPE do DOS.

2. Abrir um arquivo texto e ler seu conteúdo. O programa a seguir lê o arquivo TESTE.TXT criado pelo programa anterior. Digite o seguinte programa

```
// IFLSTR.CPP

// Lê strings de arquivo

# include <fstream.h> // para funções de arquivo

void main()

{ const int MAX=80;

char buff [MAX];

ifstream fin (“teste.txt”); // abre arquivo para leitura em modo texto

while (fin) // enquanto não terminou o arquivo

{ fin.getline(buff,MAX); // lê uma linha de texto

cout << buff << ‘\n’;

}

}
```

Para ler o arquivo, criamos um objeto da classe ifstream de nome fin e associamos este objeto ao arquivo TESTE.TXT.

Lemos o arquivo uma linha por vez, usando a função getline(). A função getline() aceita um terceiro argumento que especifica o caractere de fim de leitura. Se não for especificado este argumento, o valor default é o caractere ‘\n’.

Neste exemplo o operador >> não foi utilizado. Verifique o resultado se fosse utilizado. Substitua a linha de leitura por

```
fin >> buff;
```

Os objetos da classe ifstream têm um valor que pode ser testado para verificação do fim de arquivo. O objeto fin terá o valor zero se o sistema operacional enviar ao programa o sinal de fim-de-arquivo e um valor não zero caso contrário.

Com isso nosso programa poderia ser assim escrito:

```
// IFLSTR.CPP

// Lê strings dearquivo

# include <fstream.h> // para funções de arquivo

void main()

{ const int MAX=80;

char buff [MAX};

ifstream fin (“teste.txt”); // abre arquivo para leitura em modo texto

while (fin.getline(buff,MAX)) // enquanto não eof

cout << buff << ‘\n’;

}
```

3. Lendo e Gravando um caractere por vez no arquivo

Para trabalhar com um único caractere por vez, usamos as funções put() e get(), membros das classes ostream e istream respectivamente. O nosso próximo exemplo lê um caractere por vez do teclado e grava-o num arquivo. A leitura do teclado termina quando se pressiona CTRL-Z.

```
// OFILCH.CPP

// Grava um caractere por vez num arquivo

# include <fstream.h> // Para as funções de arquivos

void main()

{ ofstream fout (“teste1.txt”);

char ch;

// enquanto não pressionado CTRL-Z

while (cin.get(ch) // Lê um caractere do teclado

fout.put (ch);

}
```

Neste programa, o objeto ofstream é criado da mesma forma que fizemos no exemplo 1. O laço while verifica se o caractere que indica fim-de-arquivo foi digitado no teclado. Este caractere tem código ‘\x1A\ e é inserido pressionando-se a tecla CTRL junto com a tecla Z

Para ler o arquivo Teste1.txt poderíamos utilizar:

```
// OFILCH.CPP

// Lê um caractere por vez de um arquivo

# include <fstream.h> // Para as funções de arquivos

void main()

{ ifstream fin (“teste1.txt”);

char ch;

while (fin.get(ch) // Lê um caractere do arquivo até fim de arquivo

cout << ch;

}
```

Função OPEN

Quando usamos um objeto da classe ofstream ou um objeto da classe ifstream,, é necessário associá-lo a um arquivo. Esta associação pode ser feita pelo construtor da classe ou usando a função open(), membro da classe fstream, numa instrução após a criação do objeto.
Tanto o construtor como a função open() aceitam a inclusão de argumentos indicando o modo de abertura do arquivo. Esses são:

Modos	Descrição
ios::in	Abre para leitura (default de ifstream)
ios::out	Abre para gravação (defaut de ofstream)
ios::ate	abre e posiciona no final do arquivo e trabalha com leitura e escrita
ios:app	Grava a partir do fim do arquivo
ios::trunc	Abre a apaga todo o conteúdo do arquivo
ios::nocreate	Erro de abertura se o arquivo não existir
ios::noreplace	Erro de abertura se o arquivo existir.
ios::binary	Abre em binário (default é texto)

4. Gravando Objetos

Para gravar ou ler objetos em disco, não faz sentido gravar ou ler um caractere ou uma linha por vez. Se o arquivo conterá objetos, gostaríamos de gravar ou ler um objeto por vez. As funções apropriadas para o processo de gravação e leitura de objetos são as funções write() e read(). Para mostrar o seu uso, primeiramente criaremos um programa para gravar registros de livros de uma biblioteca. Eis a listagem:

```
// WFILOBJ.CPP

// Grava objetos em disco
# include <fstream.h> // Para as funções de arquivos
# include <stdio.h> // para gets()
# include <conio.h> // Para getche()

class Livro

{ private:

char titulo [50];

char autor [50];

int numreg;

double preco;

public:

void novonome();

};
void Livro::novonome()

{ cout << "\n\tDigite título: ";

gets (titulo);

cout <<"\tDigite autor: ";

gets (autor);

cout << "\tDigite o número do registro: ";

cin >> numreg;

cout << "\tDigite o preco: ";

cin >> preco;

}
```

```
void main()

{ ofstream fout (“lista.dat”);

Livro li; //cria objeto Livro

do

{ li.novonome();

fout.write((char *)&li,sizeof(Livro));

cout << “\nMais um livro (s/n)? “;

} while(getche()!=’n’);

}
```

A função novonome() é chamada para solicitar ao usuário a entrada das informações dos registros. Após a inserção de cada registro, o seu conteúdo é gravado no arquivo lista.dat por meio da função write() e é perguntado ao usuário se ele deseja inserir mais um registro.

A função write() é membro da classe ostream e recebe dois argumentos: o primeiro é o endereço do objeto a ser gravado, e o segundo, o tamanho do objeto em bytes. O endereço do objeto deve ser convertido para um ponteiro char.

5. Lendo objetos

Para ler os objetos gravados pelo programa anterior, usaremos a função read().

```
// RFILOBJ.CPP

// Lê objetos em disco

# include <fstream.h> // Para as funções de arquivos

# include <stdio.h> // para gets()

class Livro

{ private:

char titulo [50];

char autor [50];

int numreg;

double preco;

public:

void print();

};
```

```
void Livro::print()

{ cout << "\n\tTítulo: " << titulo;

cout << "\n\tAutor: " << autor;

cout << "\n\tNo.Reg: " << numreg;

cout << "\n\tPreço: " << preco;

}

void main()

{ ifstream fin ("lista.dat");

Livro li;

while(fin.read((char *)&li,sizeof(Livro)))

li.print();

}
```

A função read() é membro da classe istream e recebe dois argumentos: o primeiro é o endereço do objeto para onde irão os dados lidos, e o segundo, o tamanho do objeto em bytes. O endereço do objeto deve ser covertido para um ponteiro char.

Para que os programas de leitura e gravação de objetos trabalhem corretamente, é necessário que a seção de dados da classe dos objetos seja a mesma tanto na leitura como na gravação.

Os itens de dados devem ser iguais, as funções-membro podem ser diferentes.

6. Gravando e Lendo objetos de um mesmo arquivo.

O nosso próximo programa trabalha num mesmo arquivo tanto para gravação como para leitura. O programa grava tantos objetos quantos o usuário desejar e depois lê e imprime o conteúdo total do arquivo. Eis a listagem:

```
// WRFILOBJ.CPP

// Grava e lê objetos em disco

# include <fstream.h> // Para as funções de arquivos

# include <stdio.h> // para gets()

# include <conio.h> // Para getche()

class Livro

{ private:

char titulo [50];

char autor [50];

int numreg;

double preco;

public:

void novonome();

void print();

};

void Livro::novonome()

{ cout << "\n\tDigite título: ";

gets (titulo);

cout <<"\tDigite autor: ";

gets (autor);

cout << "\tDigite o número do registro: ";

cin >> numreg;

cout << "\tDigite o preco: ";

cin >> preco;

}
```

```
void Livro::print()

{ cout << "\n\tTítulo: " << titulo;

cout << "\n\tAutor: " << autor;

cout << "\n\tNo.Reg: " << numreg;

cout << "\n\tPreço: " << preco;

}

void main()

{ fstream fio ; // cria objeto de leitura e gravação

Livro li; //cria objeto Livro

fio.open ("lista.dat", ios::binary | ios::ate | ios::out | ios::in);

do

{ li.novonome();

fio.write((char *)&li,sizeof(Livro));

cout << "\nMais um livro (s/n)? ";

} while(getche()!='n'); // Fim se 'n'

fio.seekg(0); // coloca o ponteiro no início do arquivo

cout << "\nLista de Livros do Arquivo";

cout << "\n==================";

while (fio.read((char *)&li, sizeof(Livro)))

li.print();

}
```

A primeira novidade neste programa é a classe usada para criar o objeto fio. A instrução fstream fio; cria um objeto para leitura e gravação. A segunda é o modo usado na função open() onde usamos vários modos para especificar os aspectos desejados para o arquivo ser aberto. O uso ate serve para preservar os objetos que já existem no arquivo e acrescentar novos objetos ao final dele. Se o arquivo não existir, será criado. Usamos o out e in pois queremos executar as duas operações: de gravação e de leitura.

Após a escrita de novos objetos utilizamos a instrução fio.seekg(0); para posicionar o arquivo em seu início.

A função seekg() permite movimentar a posição corrente de leitura do arquivo para uma posição escolhida, enquanto a função seekp() executa a mesma tarefa para a posição corrente de gravação. A função seekg() tem o seguinte protótipo:

istream& seekg(long pos, seek_dir posicao=ios::beg);
o primeiro argumento indica o deslocamento em bytes a partir da posição escolhida. Quando a função é chamada com um único argumento, a posição é assumida como sendo o início do arquivo (ios::beg).

O segundo argumento, quando usado, deve ser um dos modos seguintes:

ios::beg A partir do início do arquivo

ios::cur Apartir da posição corrente

ios::end A partir do fim do arquivo.

Exercício

Elabore um programa para que se possa executar as seguintes funções:

Incluir um novo livro

Alterar os dados de um livro armazenado. Neste caso o programa deverá pedir ao usuário o número do registro do livro armazenado, pesquisar para ver se o livro existe no arquivo. Caso não exista emitir uma mensagem de advertência. Caso exista deverá listar cada dado do livro e pedir a alteração.

Exemplo: suponha que o usuário deseje alterar os dados do livro número 2.

Entre com o número do livro:2

Titulo: Treinamento em C++ // : Treinamento em Linguagem C++

Autor: Viviane // : VIVIANE Mizrahi

Registro: 2 // 2

Preço: 44.00 // 45.00

Listar o conteúdo do arquivo.