UNIVALI POLITECNICA Kobrasol São José - Ciência da Computação Algoritmos e Programação 2p

ARQUIVOS

Um arquivo consiste em uma sequência linear de dados persistentes.
A utilização de arquivos tem dois sentidos: arquivo como fonte de dados (input) ou como destino de dados (output).

Modo texto (acesso sequencial)
Constituído por caracteres perceptíveis por nós (letras, números, caracteres vulgares e separadores espaço em branco, tab...), agrupados em linhas, terminadas pelo caracter NewLine ('\n'). No final do arquivo tem-se EOF (end of file, equivale ao valor –1).
No DOS o NewLine é representado por dois caracteres CR LF (ASCII 13 e 10). O final do arquivo é marcado por CTRL Z.
Há conversão de tipos para caracteres, ou seja, um número inteiro ocupará 1 byte em disco para cada dígito.

Modo binário (acesso direto)
Compostos por qualquer caracter da Tabela ASCII, podendo conter caracteres de controle, especiais ou mesmo sem representação visível (como o caso do '\0' cujo ASCII é 0).
Não há conversão, p.ex. um número inteiro ocupará a mesma quantidade de bytes em disco que ocupa em memória.

ARQUIVOS EM C++

Os serviços de entrada e saída de arquivos em C++ são implementados através da biblioteca ***fstream***, derivada de *iostream,* que conecta um arquivo a um programa, tanto para entrada quanto para saída.

Como especialidades desta biblioteca tem-se:
- ***ifstream***, que conecta um arquivo a um programa somente para entrada.
- ***ofstream***, que conecta um arquivo a um programa somente para saída.

Os operadores de fluxo de entrada >> e de saída << também estão implementados para efetuar a entrada e saída em arquivos, bem como as funções específicas getline, get (o arquivo é usado no lugar do cout/cin).
Ex.:
```
arqSaida << "Texto"; // grava uma string no arqSaida
getline(arqEntrada,linha); // le uma string do arqEntrada
```

Opções para declaração de arquivos
```
ifstream fileIn; // Cria arquivo texto para leitura
ifstream fileIn("teste.txt"); //Cria arq leitura via construtor

ofstream fileOut; // Cria arquivo texto para gravação
ofstream fileOut("teste.txt"); //Cria arq grav. via construtor
```

UNIVALI POLITECNICA Kobrasol São José - Ciência da Computação Algoritmos e Programação 2p

Opções para abertura de arquivos

Quando não se vincula um arquivo à variável do programa imediatamente na criação, deve-se usar a função membro **open(***nomeLogico, ModoAbertura***)** para realizar abertura do arquivo.

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

Estes modos de abertura devem ser usados tanto na declaração de um arquivo (com parâmetros, via construtor) quanto na função open.

Todo o arquivo tem um controle de posicionamento interno de leitura/gravação. A cada comando de entrada/saída a posição física corrente é atualizada (a primeira posição física do arquivo é 0).

Exs.:

```
ofstream arqdiario( "movHoje.txt", ios::trunc ); // reiniciar
ifstream arqRegistros( "livros.dat", ios::binary ); // binario
fstream io( "teste.txt", ios_base::in | ios_base::app );
```

Fechamento de arquivos

Para desconectar um arquivo de um programa, deve-se usar a função membro **close().** Outra função que faz o fechamento do arquivo de forma emergencial é **exit(***parâmetro***).** Quando do fechamento do arquivo, os caracteres que permanecem no "buffer" são gravados.

Ex.: programa para gravar uma frase em arquivo texto, após recupera este conteúdo e apresenta na tela.

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
     ofstream fout("teste.txt"); // Cria arquivo txt para gravar
     fout << "Texto gravado no arquivo por meio de programa C++";
     fout.close();  // fecha o arquivo
     ifstream fin("teste.txt"); // Abre o arq para leitura
     char ch;
     // Enquanto não for fim de arquivo:
     while(fin.get(ch))  // lê um caracter do arquivo
        cout << ch;      // mostra o caracter na tela
     fin.close();        // fecha o arquivo
     return 1;
}
```

UNIVALI POLITECNICA Kobrasol São José - Ciência da Computação Algoritmos e Programação 2p

CONDIÇÕES DE ERRO

Situações de erro ao tratar com arquivos podem ser analisadas através de status da classe ios, obtida pela função **rdstate()**. Os bits individuais do valor encontrado podem ser testados pelo operador AND bit-a-bit (&) e os seguintes valores enumerados:

- ios::goodbit Nenhum bit setado, sem erros
- ios::eofbit Encontrado o fim de arquivo
- ios::failbit Erro de leitura ou gravação
- ios::badbit Erro irrecuperável

A função **clear()** de protótipo `void clear(int status=0);` modifica o status. Se usada sem argumentos, todos os bits são limpos. Do contrário, os bits são setados de acordo com os valores enumerados escolhidos e combinados pelo operador OR (|).
Ex.: `clear (ios::eofbit | ios::failtbit);`

Outras funções que retornam o status de um bit individual: `good(), eof(), fail(), bad()`.

Ex. do uso destas funções de erro:

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
    ifstream fin("copia.xxx");
    if(!fin)
        cout << "\nNao posso abrir arquivo copia.xxx\n";
    else
        cout << "\nArquivo aberto com sucesso\n";

    cout << "\nrdstate() = " << fin.rdstate();
    cout << "\ngood() = " << fin.good();
    cout << "\neof() = " << fin.eof();
    cout << "\nfail() = " << fin.fail();
    cout << "\nbad() = " << fin.bad();
    return 1;
}
```

No caso deste programa, caso o arquivo não exista, a resposta gerada será: 4, 0, 0, 1, 0 (cfe. os cout's acima). Caso exista e esteja em perfeitas condições para uso, será 0, 1, 0, 0, 0.

LEITURA E GRAVAÇÃO DE STRUCTS

Variáveis estruturadas (structs) podem ser gravados/lidos em arquivos:

```
<ofsrteam>.write((const char*)(&var), sizeof(tipo));
   <ifstream>.read((char*)(&var), sizeof(tipo));
```

UNIVALI POLITECNICA Kobrasol São José - Ciência da Computação Algoritmos e Programação 2p

Ex. arquivos binários + structs: alguns autores indicam colocar no início do arquivo a quantidade de registros que estão armazenados no arquivo, facilitando sua manipulação.

```
#include <iostream>
#include <fstream>
using namespace std;
const int MAX_FUNCS = 3;
typedef struct {
      int codigo;
      char nome[50];  // tem q ser string no padrao C (vetor de char)
      float salario;
      char sexo;
} Funcionario;

void leFuncionarios(Funcionario f[], int quantidade){
      string nome;
      for (int i = 0; i < quantidade; i++){
            cout << "Leitura de dados de um funcionario\n\n";
            cout << "Codigo.: ";
            cin >> f[i].codigo;
            cin.ignore();
            cout << "Nome...: ";
            getline(cin,nome); strcpy(f[i].nome, nome.c_str());
            cout << "Salario: ";
            cin >> f[i].salario; cin.ignore();
            cout << "Sexo...: ";
            cin >> f[i].sexo; cin.ignore();
      }
}
void mostraFuncionarios(Funcionario f[], int quantidade){
      for (int i = 0; i < quantidade; i++) {
            cout << "Leitura de dados de um funcionario\n\n";
            cout << "Codigo.: " << f[i].codigo << endl;
            cout << "Nome...: " << f[i].nome << endl;
            cout << "Salario: " << f[i].salario << endl;
            cout << "Sexo...: " << f[i].sexo << endl;
      }
}
void gravaFuncionarios(ofstream &ofs, Funcionario f[], int quantidade){
      // grava a quantidade de funcionarios
      ofs.write((char*)&quantidade, sizeof(int));
      // grava cada um dos funcionários como um vetor de caracteres
      for (int i = 0; i < quantidade; i++)
            ofs.write((const char*)(&f[i]), sizeof(f[i]));
}
void recupFuncionarios(ifstream &ifs, Funcionario f[], int &quantidade){
      // lê o número de funcionários gravados no cabeçalho
      ifs.read((char*)&quantidade, sizeof(int));
      // lê os dados
      for (int i = 0; i < quantidade; i++)
            ifs.read((char*)(&f[i]), sizeof(f[i]));
}
int main(){
      string nomeArquivo;
      Funcionario funcionarios[MAX_FUNCS], lidos[MAX_FUNCS];
      ofstream ofs;
```

```
	leFuncionarios(funcionarios, MAX_FUNCS); // preenche vetor c/dados

	cout << "Nome do arquivo: ";
	getline(cin,nomeArquivo);
	ofs.open(nomeArquivo.c_str(), ios::out | ios::binary);

	if(!ofs){
		cout << "Arquivo: " << nomeArquivo << " nao pode ser aberto para
		escrita." << endl;
	}else{
		// grava informações do vetor no arquivo e fecha arquivo
		gravaFuncionarios(ofs, funcionarios, MAX_FUNCS);
		ofs.close();
	}

	cout << "\nRecuperando os dados do arquivo";
	ifstream ifs(nomeArquivo.c_str(), ios::binary);

	if(!ifs){
		cout << "Arquivo: " << nomeArquivo << " nao pode ser aberto para
		leitura." << endl;
	}else{
		int quantidade = 0;
		// lê as informações do arquivo e mostra
		recupFuncionarios(ifs, lidos, quantidade);
		mostraFuncionarios(lidos, quantidade);
		ifs.close();
	}
	return 0;
}
```

ACESSO ALEATÓRIO

Em arquivos pode-se acessar aleatoriamente um determinado dado utilizando as funções membro:

- `seekg(long n)` ou `seekp(long n)` – posiciona no byte n (a contar do início do arquivo), o que permite alterar o acesso ao arquivo para ações de leitura (get)/gravação(put).
- `seekg(offset, direcao)` ou `seekp(offset, direcao)` onde offset (deslocamento em bytes) é contado a partir de uma direção, que pode ser
  - ✓ `ios::beg`, início do arquivo
  - ✓ `ios::cur`, posição corrente do arquivo
  - ✓ `ios::end`, final do arquivo

Supondo a leitura de registros com tamanho sizeof(Tipo), pode-se posicionar o arquivo para leitura de cada um dos i registros através de:

```
for (int i = 0; i < registroCont; i++)
   readFile.seekg(i*sizeof(Tipo), ios::beg);
```

Avanço de um registro em relação à posição corrente:

```
arqleitura.seekg(sizeof(Tipo), ios::cur);
```

Pode-se também especificar um deslocamento negativo:

```
arqleitura.seekg(-sizeof(Tipo), ios::cur);
```

UNIVALI POLITECNICA Kobrasol São José - Ciência da Computação Algoritmos e Programação 2p

Em arquivos, pode-se também obter a posição corrente de um arquivo utilizando as funções membro:

- `tellg()` ou `tellp()`– retorna o valor do ponteiro get/put em relação ao início do arquivo de leitura/gravação.

Por exemplo:

```
ios::pos_type mark = writeFile.tellp(); // guarda posição atual
// ...
if(error)
   writeFile.seekp(mark); // retorna à posição armazenada
```

Se estas funções forem executadas quando o ponteiro get/put esteja no final do arquivo, tem-se o tamanho deste arquivo (em bytes).

Ex. arquivos binários + structs + acesso aleatório: sem quantidade de itens no início do arquivo.

```
#include <iostream>
#include <fstream>
#include <cstdlib> // para o exit
using namespace std;

typedef struct {
     float battery, bulbs, fuses;
} Tipo;

// PARA GERAR ARQUIVO, DE FORMA AUTOMATIZADA PARA USO NO ACESSO ALEATORIO
int main(){
     Tipo x;
     x.battery=30.99; x.bulbs=3.22; x.fuses=2.00;
     ofstream out_file("teste.dat");

     if (out_file.fail()){
          cout << "Impossivel abrir o arquivo" << endl; exit(1);
     }
     cout << out_file.tellp() << endl; // mostra tam em bytes do arq
     do{
          out_file.write((char*)(&x),sizeof(Tipo)); // grava item
          x.fuses-=0.25;
     }while(x.fuses>0.00);
     out_file.close();
     return 0;
}

//PARA ACESSO ALEATORIO DO ARQUIVO GERADO ANTERIORMENTE
int main(){
     Tipo x;
     int n;
     char resp;
     ifstream in_file("teste.dat");
     if (in_file.fail()){
          cout << "Impossivel abrir o arquivo, veja se existe" << endl;
          exit(1);
     }
```

```
      do {
            cout << "n 1 a 8: ";
            cin >> n;
            n--; // pq 1° reg estah posição 0

            in_file.seekg(sizeof(Tipo)*n,ios::beg); // posiciona registro n

            if(in_file.peek()!= EOF) {
                  in_file.read((char*)(&x),sizeof(Tipo));
                  cout <<"Batteries " << x.battery << "Bulbs " << x.bulbs
                  << "Fuses " << x.fuses<< endl;
                  cout << in_file.tellg()<< endl;
            } else
                  cout <<"Inexistente"<< endl;

            cin.ignore();
            cout << "Outro ??";
            resp = toupper(cin.get());
      } while ( resp=='S');
      in_file.close();
      return 0;
}
// se fosse posicionamento em arquivo ofstream => seekp()/tellp() !!!!
```