Streams e Arquivos em C++

Roland Teodorowitsch

Programação Orientada a Objetos - ECo - Curso de Engenharia de Computação - PUCRS

30 de agosto de 2023

Streams

Streams em C++

- C++ permite acesso a dispositivos de entrada e saída
- *Streams* são abstrações que correspondem a fluxos de entrada e saída
- `cout` e `cin` são fluxos de entrada e saída para o terminal
  - `cout` é usado para escrever
  - `cin` é usado para ler
- A mesma abstração poderia ser usada para compor *strings*, como se estivéssemos escrevendo na tela

stringstream

- Exemplo:

```
#include <iostream>
#include <sstream>
using namespace std;
int main () {
  string aux = "string de teste ";
  stringstream ss;
  ss << "120 42 377 6 5 20\n";
  ss << aux;
  ss << 10;
  ss << "\n\tteste\n";
  ss << "final de string";
  cout << "resultado: " << ss.str();
  cout << endl;
  return 0;
}
```

- Resultado:

```
resultado: 120 42 377 6 5 20
string de teste 10
        teste
final de string
```

Usando stringstream para criar método str() para objeto

```
#include <iostream>
#include <sstream>

using namespace std;

class Pessoa {
private:
  string nome;
  int idade;
public:
  Pessoa(string n = "", int i = 0) { nome = n;  idade = i; }

  string str() {
    stringstream ss;
    ss << nome << " (" << idade << " anos)";
    return ss.str();
  }

};

int main() {
  Pessoa p("Joao", 21);
  cout << p.str() << endl;
  return 0;
}
```

Arquivos

Streams em C++

- *Streams*: arquivos texto, arquivos binários, *sockets*, etc.
- Acesso a arquivos:
  - Criação e gravação de dados em arquivos
  - Leitura dos dados de arquivos
- Exemplos:
  - Gravação de um conjunto de números aleatórios em um **arquivo texto** em C++ (`escrita.cpp`)
  - Leitura desse arquivo (`leitura.cpp`)

Escrevendo em um Arquivo Texto

- Crie um arquivo chamado `escrita.cpp` e acrescente a ele os trechos de código a seguir...
- O principal arquivo de cabeçalho para trabalhar com arquivos em C++ é `fstream`
- O início do cógigo-fonte será, portanto:

```
#include <fstream>   // para usar file streams (ifstream, ofstream)
#include <iostream>  // para usar cin, cout
#include <string>    // para usar string
#include <iomanip>   // para usar manip. (setw, right, ...)
#include <cstdlib>   // para usar srand(), rand() e exit()
#include <ctime>     // para usar time()

#define TAM 10000

using namespace std;
```

Escrevendo em um Arquivo Texto

- O primeiro passo dentro de `main()` é criar uma instância de `ofstream`, que será o fluxo de saída associado ao arquivo:

```
int main() {
  ofstream arqsaida; // Cria output file stream (ofstream)
```

- O próximo passo é abrir o arquivo para escrita:

```
  arqsaida.open("teste.txt", ios::out ); // Cria e abre arquivo
  if (!arqsaida.is_open()) return 1; // Se houver erro, sai do programa
```

Neste trecho de código:

- Usa-se o modo `ios::out`, que cria o arquivo e abre ele para escrita
- Cuidado: se esse modo for usado em arquivos já existentes, eles serão **apagados**!
- O método `is_open()` retorna `false` se o arquivo não está aberto, então usa-se para testar se foi possível realizar a operação

Escrevendo em um Arquivo Texto

- Como em linguagem C, a função `srand()` é utilizada para inicializar o gerador de números aleatórios (semente)

```
srand(time(NULL)); // Semente utilizando o tempo atual do sistema
```

- Início da gravação dos dados (cabeçalho do arquivo):

```
cout << "Gerando dados..." << endl;
arqsaida << "Cabecalho do arquivo" << endl; // Grava o cabecalho
```

- Gravação dos TAM registros numéricos:

```
for (int i = 0; i < TAM; i++) {
    int num = rand() % TAM; // Gera numero aleatorio
    arqsaida << i << setw(10) << num << endl; // Grava no arquivo
    if (arqsaida.fail()) { cerr << "Erro fatal!" << endl;  exit(1); }
}
```

Escrevendo em um Arquivo Texto

- Para finalizar, feche o arquivo:

```
  cout << "Fechando o arquivo..." << endl;
  arqsaida.close();
  return 0;
}
```

- Compile o programa `escrita.cpp`, depois execute-o e verifique se o arquivo `teste.txt` foi criado no diretório corrente

Números Aleatórios em C++

- C++11 traz uma nova forma para gerar números randômicos
- Sintaxe:

```
// Arquivo de cabecalho da biblioteca
#include <random>

// Obtem um numero randomico do HW:
std::random_device rd;
// Gerador de semente:
std::mt19937 eng(rd());
// Define a distribuicao, neste caso a partir de 1 ate 10:
std::uniform_int_distribution<> distr(1, 10);
// Gera o numero randomico:
int num = distr(eng);
```

Lendo de um Arquivo Texto

- Crie um arquivo chamado `leitura.cpp` e acrescente a ele os trechos de código a seguir...
- Os arquivos de cabeçalho que devem ser incluídos são os mesmos do programa que faz a escrita:

```
#include <fstream>   // para usar file streams (ifstream, ofstream)
#include <iostream>  // para usar cin, cout
#include <string>    // para usar string
#include <iomanip>   // para usar manip. (setw, right, ...)
#include <cstdlib>   // para usar srand(), rand() e exit()
#include <ctime>     // para usar time()

using namespace std;
```

Lendo de um Arquivo Texto

- O primeiro passo dentro de `main()` é criar uma instância de `ifstream`, que será o fluxo de entrada associado do arquivo:

```
int main() {
  ifstream arq; // Cria input file stream (ifstream)
```

- O próximo passo é abrir o arquivo para leitura:

```
  arq.open( "teste.txt" , ios::in ); // Abre arquivo
  if (!arq.is_open()) return 1; // Se houver erro, sai do programa
```

Neste trecho de código, usa-se o modo `ios::in`, que abre o arquivo para leitura

Lendo de um Arquivo Texto

- Ler o cabeçalho do arquivo:

```
string cabecalho;
getline(arq,cabecalho); // Le cabecalho
cout << cabecalho << endl; // Exibe cabecalho na tela
```

Lendo de um Arquivo Texto

- Ler cada um dos valores que estão gravados no arquivo, mas, em vez de ler linhas, os valores serão lidos como inteiros:

```
do { // Le n registros numericos
   int num, valor;
   arq >> num >> valor;
   if (!arq.fail()) cout << num << " " << valor << endl;
}  while (arq.good());
```

Sobre a leitura:

- Depois do cabeçalho, as próximas linhas são compostas de dois números: um contador e o valor armazenado
- Basta então fazer uma repetição, que terminará quando não houver mais dados no arquivo
- O método `good()` retorna `false` quando algo diferente acontecer (erro ou final do arquivo)
- O dado lido só é exibido na tela, se o método `fail()` retornar `false`

Lendo de um Arquivo Texto

- O laço pode ter terminado por dois motivos: ou houve um erro, ou o arquivo terminou
  - No primeiro caso, o método `bad()` retorna `true`
  - No segundo caso, o método `eof()` retorna `true`
- Logo, uma situação de erro ocorre quando `bad()` retornou `true`, ou `eof()` retornou `false`:

```
if (arq.bad() || !arq.eof()) { // Aborta programa
    cerr << "Erro fatal!" << endl;  arq.close();  exit(1);
}
```

Lendo de um Arquivo Texto

- Para finalizar, feche o arquivo:

```
  cout << "Fechando o arquivo..." << endl;
  arq.close();
  return 0;
}
```

- Compile o programa `leitura.cpp`, depois execute-o e verifique se os dados do arquivo `teste.txt` estão sendo corretamente exibidos no terminal
- Mais informações em:
  - `http://www.cplusplus.com/reference/istream/iostream/`
  - `http://www.inf.pucrs.br/~pinho/PRGSWB/Streams/streams.html`
  - `http://www.inf.pucrs.br/~flash/lapro2ec/aulas/streams/aula_streams.html`

Lista de Exercícios

Lista de Exercícios

1. Copie os códigos vistos em aula em dois arquivos (respectivamente, `escrita.cpp` para o programa que gera os números e os salva em arquivo e `leitura.cpp` para o programa que lê os números do arquivo, exibindo-os no terminal). Compile-os e teste-os para verificar a criação e leitura do arquivo "teste.txt" no diretório corrente.

Lista de Exercícios

2. Escreva um programa em C++ que permita controlar as presenças dos alunos de uma turma. O programa deve:
   a. Ler os dados da turma em um arquivo com o seguinte formato: linha inicial contendo o número de alunos e o número de presenças ou faltas. Na sequência aparece uma linha para cada aluno com: nome do aluno (considere nomes com apenas uma palavra) seguido pelas presenças já registradas desse aluno ("P" para presença, "F" para falta). Crie uma classe `Aluno` (que possui nome e vetor parcialmente preenchido para as presenças) e trabalhe com um vetor parcialmente preenchido de objetos da classe `Aluno`. O conteúdo do arquivo poderia ser o seguinte:

```
3 5
Joaozinho P P P F P
Mariazinha F P F P P
Pedrinho P P P P P
```

   b. Realizar uma chamada na data corrente, imprimindo o nome do aluno e perguntando se o aluno está presente ("P") ou ausente ("F").
   c. Imprimir um relatório indicando para cada aluno: número de presenças, número de faltas e porcentagem de aulas que o aluno assistiu.
   d. Gerar um novo arquivo com o controle de presenças atualizado.

Créditos

Créditos

- Estas lâminas contêm trechos de materiais disponibilizados pelos professores Rafael Garibotti e Matheus Trevisan.

Soluções

Exercício 1: escrita.cpp

```cpp
#include <fstream>    // para usar file streams (ifstream, ofstream)
#include <iostream>   // para usar cin, cout
#include <string>     // para usar string
#include <iomanip>    // para usar manip. (setw, right, ...)
#include <cstdlib>    // para usar srand(), rand() e exit()
#include <ctime>      // para usar time()

#define TAM 10000

using namespace std;

int main() {
  ofstream arqsaida; // Cria output file stream (ofstream)
  arqsaida.open("teste.txt", ios::out ); // Cria e abre arquivo
  if (!arqsaida.is_open()) return 1; // Se houver erro, sai do programa
  srand(time(NULL)); // Semente utilizando o tempo atual do sistema
  cout << "Gerando dados..." << endl;
  arqsaida << "Cabecalho do arquivo" << endl; // Grava o cabecalho
  for (int i = 0; i < TAM; i++) {
      int num = rand() % TAM; // Gera numero aleatorio
      arqsaida << i << setw(10) << num << endl; // Grava no arquivo
      if (arqsaida.fail()) { cerr << "Erro fatal!" << endl;  exit(1); }
  }
  cout << "Fechando o arquivo..." << endl;
  arqsaida.close();
  return 0;
}
```

Exercício 1: leitura.cpp

```cpp
#include <fstream>    // para usar file streams (ifstream, ofstream)
#include <iostream>   // para usar cin, cout
#include <string>     // para usar string
#include <iomanip>    // para usar manip. (setw, right, ...)
#include <cstdlib>    // para usar srand(), rand() e exit()
#include <ctime>      // para usar time()

using namespace std;

int main() {
  ifstream arq; // Cria input file stream (ifstream)
  cout << "Abrindo arquivo texto..." << endl;
  arq.open( "teste.txt" , ios::in ); // Abre arquivo
  if (!arq.is_open()) return 1; // Se houver erro, sai do programa
  string cabecalho;
  getline(arq,cabecalho); // Le cabecalho
  cout << cabecalho << endl; // Exibe cabecalho na tela
  do { // Le n registros numericos
     int num, valor;
     arq >> num >> valor;
     if (!arq.fail()) cout << num << " " << valor << endl;
  }  while (arq.good());
  if (arq.bad() || !arq.eof()) { // Aborta programa
     cerr << "Erro fatal!" << endl;  arq.close();  exit(1);
  }
  cout << "Fechando o arquivo..." << endl;
  arq.close();
  return 0;
}
```

Exercício 2: Aluno.hpp

```
#ifndef _ALUNO_HPP
#define _ALUNO_HPP

#include <string>

#define MAX_PRESENCAS 100

using namespace std;

class Aluno {
private:
  string nome;
  int numPresencas;
  char presencas[MAX_PRESENCAS];
public:
  Aluno(string n = "");
  ~Aluno();
  string obtemNome();
  void defineNome(string n);
  int obtemNumPresencas();
  void limpaPresencas();
  bool obtemPresenca(int d, char &p);
  string str();
  bool definePresenca(int d, char p);
  bool registraPresenca(char p);
};
#endif
```

Exercício 2: `Aluno.cpp` (primeira parte)

```
#ifdef DEBUG
#include <iostream>
#endif
#include <sstream>
#include "Aluno.hpp"

Aluno::Aluno(string n) {
  nome = n;
  numPresencas = 0;
  #ifdef DEBUG
  cout << "+ Aluno (" << nome << ") criado..." << endl;
  #endif
}

Aluno::~Aluno() {
  #ifdef DEBUG
  cout << "- Aluno (" << nome << ") destruido..." << endl;
  #endif
}

string Aluno::obtemNome() { return nome; }
void Aluno::defineNome(string n) { nome = n; }
int Aluno::obtemNumPresencas() { return numPresencas; }
void Aluno::limpaPresencas() { numPresencas = 0; }
```

Exercício 2: `Aluno.cpp` (segunda parte)

```
bool Aluno::obtemPresenca(int d, char &p) {
  if (d < 0 || d >= numPresencas) return false;
  p = presencas[d];
  return true;
}

string Aluno::str() {
  stringstream ss;
  ss << nome;
  for (int i=0; i<numPresencas; ++i) ss << " " << presencas[i];
  return ss.str();
}

bool Aluno::definePresenca(int d, char p) {
  if (d < 0 || d >= numPresencas) return false;
  presencas[d] = p;
  return true;
}

bool Aluno::registraPresenca(char p) {
  if (numPresencas >= MAX_PRESENCAS) return false;
  presencas[ numPresencas++ ] = p;
  return true;
}
```

Exercício 2: chamada.cpp (primeira parte)

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include "Aluno.hpp"

#define MAX_ALUNOS 50

using namespace std;

int main() {
  char presenca;

  ifstream fin("chamada.txt", ios::in);  // LEITURA
  if (!fin.is_open()) { cerr << "Impossivel abrir arquivo 'chamada.txt'..." << endl;  return 1; }
  int numAlunos, numPresencas;
  fin >> numAlunos >> numPresencas;
  Aluno alunos[MAX_ALUNOS]; // 50 alunos serao criados (construtor chamado 50 vezes)
  for (int i=0; i<numAlunos; ++i) {
      string nome;
      fin >> nome;
      alunos[i].defineNome(nome);
      alunos[i].limpaPresencas();
      for (int j=0; j<numPresencas; ++j) {
          fin >> presenca;
          alunos[i].registraPresenca(presenca);
      }
  }
  fin.close();
```

Exercício 2: chamada.cpp (segunda parte)

```
  for (int i=0; i<numAlunos; ++i) {  // CHAMADA
      cout << alunos[i].obtemNome() << " (P/F)? ";
      cin >> presenca;
      alunos[i].registraPresenca(p);
  }
  ++numPresencas;

  for (int i=0; i<numAlunos; ++i) {  // RELATORIO
      int dias =  alunos[i].obtemNumPresencas(), p = 0, f = 0;
      for (int j=0; j<dias; ++j) {
          alunos[i].obtemPresenca(j,presenca);
          if ( presenca == 'P' ) ++p;
          else if (presenca = 'F' ) ++f;
      }
      double perc = 100.0 * p / dias;
      cout << alunos[i].obtemNome() << " / presencas=" << p << " / faltas=" << f;
      cout << " / perc=" << fixed << setprecision(2) << perc << endl;
  }

  ofstream fout("chamada2.txt", ios::out);  // ESCRITA
  if (!fout.is_open()) { cerr << "Impossivel abrir arquivo 'chamada2.txt'..." << endl; return 1; }
  fout << numAlunos << " " << numPresencas << endl;
  for (int i=0; i<numAlunos; ++i) fout << alunos[i].str() << endl;
  fout.close();

  return 0;
}
```