Programação de Computadores

ARQUIVOS BINÁRIOS

Judson Santos Santiago

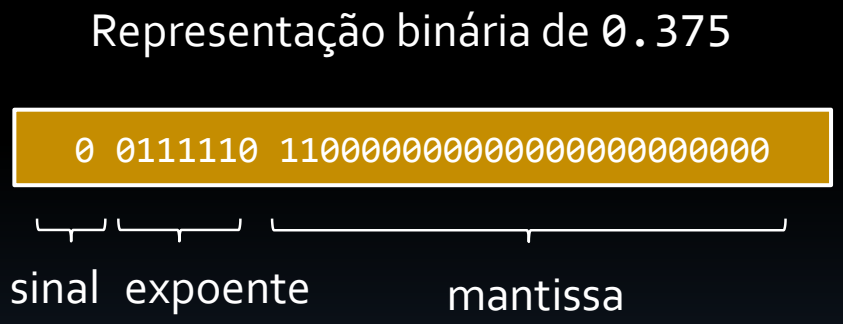
Introdução

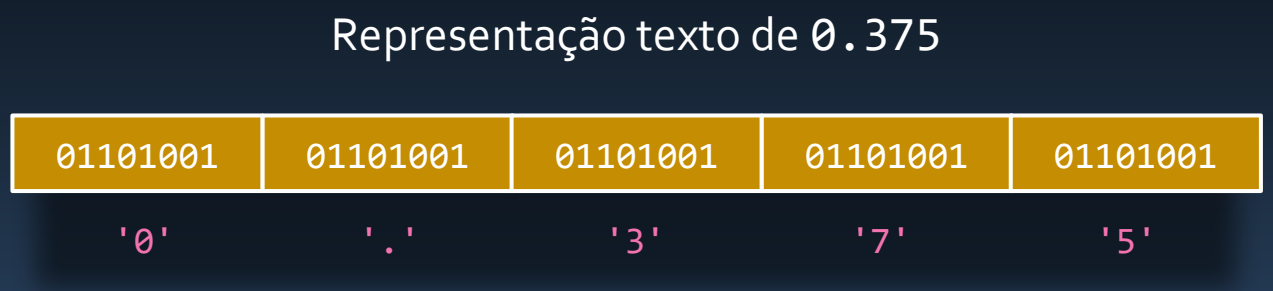
- Um arquivo é um conjunto de bits gravados em algum dispositivo de armazenamento permanente

- Para o programador, os arquivos se dividem em:
  - Arquivos texto
    Um conjunto de n bits representa um caractere
  - Arquivos binários
    Um conjunto de n bits representa uma informação na sua forma nativa (inteira, ponto-flutuante, caractere, etc.)

Introdução

- Diferença entre arquivos texto e binário:

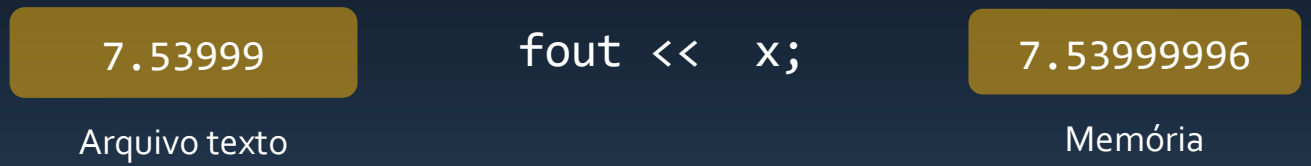




Introdução

- Arquivos texto são práticos
  - Facilmente exibidos por qualquer editor de texto

- Porém muitas aplicações necessitam:
  - Ocultar a informação armazenada
  - Manter a precisão da informação



Modos de Abertura

- Ao abrir um arquivo pode-se passar o modo de abertura

```
fin.open("boliche.txt", ios_base::in);
```

- O modo de abertura permite definir como ele será usado:
  - Escrita de dados
  - Leitura de dados
  - Adição de dados
  - Modo texto ou binário

Modos de Abertura

- A tabela abaixo lista as constantes que podem ser usadas para o modo de abertura:

Constante	Significado
ios_base::in	Abre arquivo para leitura
ios_base::out	Abre arquivo para escrita
ios_base::ate	Escreve no final do arquivo
ios_base::app	Adiciona ao final do arquivo
ios_base::trunc	Limpa arquivo, se ele existir
ios_base::binary	Cria arquivo binário

`ifstream` aceita apenas `ios_base::in` e `ios_base::binary`

`ofstream` aceita todas mas `ios_base::in` apenas faz com que o arquivo não seja limpo.

Modos de Abertura

- Se o modo de abertura é omitido são usados valores padrões:
  - Ifstream

```
fin.open("boliche.txt", ios_base::in);
```

  - Ofstream

```
fout.open("pesca.txt", ios_base::out | ios_base::trunc);
```

```
00000001: 1 (out)
00000010: 2 (trunc)
----------------------
00000011: 3 (out|trunc)
```

- Para escrever dados sem apagar o arquivo:

```
fout.open("append.txt", ios_base::out | ios_base::app);
```

Arquivos Binários

- A manipulação de arquivos binários é feita com registros
  - O registro se torna o molde usado para gravar e ler informações

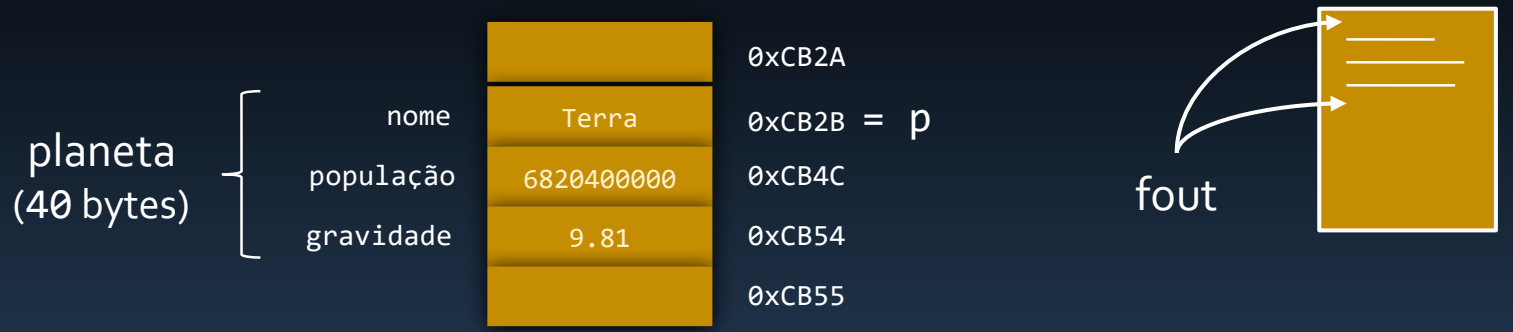
```
struct planeta
{
    char nome[24];      // nome do planeta
    double populacao;   // número de habitantes
    double gravidade;   // aceleração da gravidade
};
```

Arquivos Binários

- Para salvar o registro em um arquivo binário

```
planeta p = {"Terra", 6820400000, 9.81};

ofstream fout;
fout.open("planetas.dat", ios_base::out | ios_base::binary);
fout.write( &p, sizeof(planeta));
```



Arquivos Binários

- Para recuperar a informação é preciso usar o mesmo registro

```
struct planeta
{
    char nome[24];      // nome do planeta
    double populacao;   // número de habitantes
    double gravidade;   // aceleração da gravidade
};

planeta n;
```

- Para ler o registro de um arquivo binário:

```
ifstream fin;
fin.open("planetas.dat", ios_base::in | ios_base::binary);
fin.read((char*) &n, sizeof(planeta));
```

Arquivos Binários

```cpp
// entrada e saída em arquivos binários
#include <iostream>
#include <fstream>
using namespace std;

struct planeta
{
    char nome[24];     // nome do planeta
    double populacao;  // número de habitantes
    double gravidade;  // aceleração da gravidade
};

int main()
{
     planeta p;

     ifstream fin;     // cria objeto para leitura de arquivo
     fin.open("planetas.dat", ios_base::in | ios_base::binary);

     // continua
```

// Planetas.cpp

Arquivos Binários

```
if (fin.is_open())     // se o arquivo foi aberto sem erros
{
   cout << "Aqui está o conteúdo do arquivo:" << endl;
   while (fin.read((char *) &p, sizeof(planeta)))
   {
       cout << p.nome << " "
            << p.populacao << " "
            << p.gravidade << endl;
   }
   fin.close();
}
// acrescenta mais dados
ofstream fout;
fout.open("planetas.dat", ios_base::out | ios_base::app | ios_base::binary);

if (!fout.is_open())
{
    cout << "Arquivo não pode ser aberto!" << endl;
    system("pause");
    return EXIT_FAILURE;
}
```

// Planetas.cpp

Arquivos Binários

```
        cout << "\nNome do planeta: ";
        cin >> p.nome;
        cout << "População: ";
        cin >> p.populacao;
        cout << "Gravidade: ";
        cin >> p.gravidade;

        fout.write((char *) &p, sizeof(planeta));
        fout.close();

        fin.open("planetas.dat", ios_base::in | ios_base::binary);
        if (fin.is_open())
        {
            cout << "\nAqui está o conteúdo do arquivo:" << endl;
            while (fin.read((char *) &p, sizeof(planeta))) {
                  cout << p.nome << " " << p.populacao << " "
                       << p.gravidade << endl;
            }
            fin.close();
        }
}
```

// Planetas.cpp

Arquivos Binários

- Saída do programa:

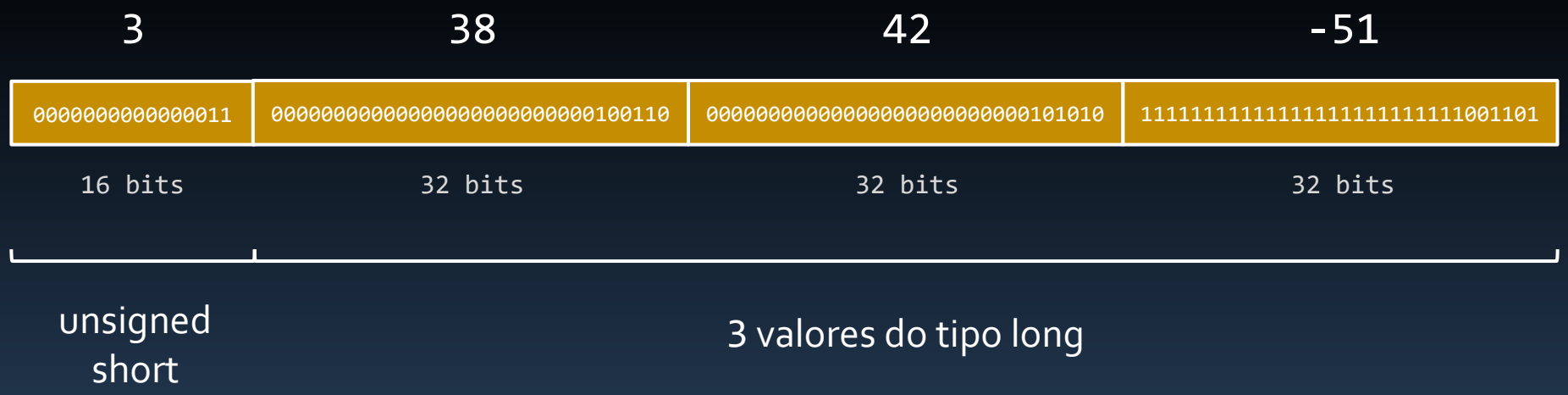
```
Aqui está o conteúdo do arquivo:
Terra 6820400000 9.81
Marte 3 9.81
Venus 0 6.53
Mercurio 0 8.88

Nome do planeta: Saturno
População: 0
Gravidade: 3.10

Aqui está o conteúdo do arquivo:
Terra 6820400000 9.81
Marte 3 9.81
Venus 0 6.53
Mercurio 0 8.88
Saturno 0 3.10
```

Usando Cabeçalhos

- Os registros facilitam a escrita/leitura em arquivos binários
  - Mas o uso deles não é obrigatório
  - Para ler basta conhecer a ordem e o tipo dos dados escritos



Usando Cabeçalhos

```cpp
#include <fstream>
using namespace std;
int main()
{
    ofstream fout;
    fout.open("valores.dat", ios_base::out | ios_base::binary);

    const unsigned short TamVet = 3;
    long vet[TamVet] = { 38, 42, -51 };

    // escreve quantidade de elementos
    fout.write((char *) &TamVet, sizeof(unsigned short));

    // escreve elementos do vetor
    for (unsigned short i = 0; i < TamVet; ++i)
        fout.write((char *) &vet[i], sizeof(long));
    fout.close();
}
```

// Escrita.cpp

Usando Cabeçalhos

- Saída do programa:

```
Pressione qualquer tecla para continuar...
```

  - A gravação do vetor poderia ser feita em uma única instrução:

```
const unsigned short TamVet = 3;
long vet[TamVet] = { 38, 42, -51 };

...

fout.write((char*) vet, sizeof(long) * TamVet);
```

Usando Cabeçalhos

```cpp
#include <fstream>
using namespace std;
int main()
{
    ifstream fin;
    fin.open("valores.dat", ios_base::in | ios_base::binary);

    // lê a quantidade de elementos e cria vetor dinâmico
    unsigned short tam;
    fin.read((char *) &tam, sizeof(unsigned short));
    long * vet = new long[tam];

    // lê elementos para o vetor
    for (unsigned short i = 0; i < tam; ++i)
        fin.read((char *) &vet[i], sizeof(long));

    fin.close();
    delete [] vet;
}
```

// Leitura.cpp

Usando Cabeçalhos

- Saída do programa:

```
Pressione qualquer tecla para continuar...
```

  - A leitura do vetor poderia ser feita em uma única instrução:

```
unsigned short tam;
fin.read((char *) &tam, sizeof(unsigned short));
long * vet = new long[tam];

...

fout.read((char*) vet, sizeof(long) * tam);
```

Resumo

- Dados podem ser armazenados em:
  - Arquivos texto
    - Podem ser lidos por qualquer editor de texto
    - Marca de fim de linha (CR/LF) é diferente para cada S.O.
  - Arquivos binários
    - São mais precisos para armazenar números ponto-flutuante
    - As operações de leitura e escrita são mais rápidas
    - Geralmente ocupam mais espaço