UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Programação e Desenvolvimento de Software 2

Resolvendo Problemas com TADs

Vítor Machado Guilherme Barros
vitorbarros@dcc.ufmg.br

DCC
DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO

Resolvendo Problemas com TADs

- ***Containers***
  - Vector
  - List
  - Map
  - Set
- **Estruturas auxiliares**
  - Functors
  - Iterators

Resolvendo Problemas com TADs (Vector)

- **Vector**
  - Estrutura amplamente utilizada para armazenar dados de forma contígua na memória.
  - Aritmética de ponteiros pode ser utilizada para iterar sob a estrutura. **Por quê?**
  - Objetos removidos da estrutura são, também, automaticamente apagados da memória.
  - Adição de novos elementos é mais eficiente quando feita ao fim da estrutura.
  - **Capacity** *vs.* **Size**: Tamanho alocado para a estrutura *vs.* número de elementos armazenados na estrutura.

Referência: http://www.cplusplus.com/reference/vector/vector/

Resolvendo Problemas com TADs (List)

- **List**
  - Indica-se o uso quando a combinação de várias listas é necessária, ou quando a adição e remoção de elementos no meio da estrutura é frequente.
  - **Estabilidade**. Iteradores e referências são inválidas se e somente se apontarem para um objeto que foi apagado.
  - São implementadas como listas duplamente encadeadas (*list*) ou listas encadeadas (*forward_list*).
  - Adição de novos elementos tem mesma eficiência em **qualquer posição** da lista. **Por quê?**

Referência: http://www.cplusplus.com/reference/list/list/

Resolvendo Problemas com TADs

- **Problema I:**
  - Imagine que um carro autônomo deve navegar por uma rodovia e realizar paradas em locais específicos previamente determinados. Esses locais são ordenados de acordo com um identificador e todos os locais são informados previamente, sem adição ou remoção de novas paradas durante a viagem.

Resolvendo Problemas com TADs

- **TAD Parada:**
  - Dado o **TAD Parada**, e considerando que as informações sobre as paradas são passadas por um arquivo de texto na entrada, qual a melhor estrutura STL para armazenar essas paradas?

```
struct Parada{
  int id;
  float latitude;
  float longitude;
  /*
  * Outros atributos...
  */
};
```

Resolvendo Problemas com TADs

- **Solução:**

```
int main(){
    int index = 0;
    float latitude, longitude;
    std::vector<Parada> paradas;
    std::string line;
    std::ifstream arq;
    arq.open("coordenadas.txt");
    if(!arq){
        exit(1);
    }
    while(arq >> latitude >> longitude){
        Parada p;
        p.id = index;
        p.latitude = latitude;
        p.longitude = longitude;
        paradas.push_back(p);
        index++;
    }
    arq.close();
    /*
    * Outras operações...
    */
    return 0;
}
```

Vetor de **objetos** do tipo Parada.

Objeto do tipo Parada.

Adição de objetos ao vetor.

Mas e se minhas Paradas fossem **ponteiros**?

Resolvendo Problemas com TADs

```
Parada::Parada(int id, float latitude, float longitude){
    this->id = id;
    this->latitude = latitude;
    this->longitude = longitude;
}

int main(){
    int index = 0;
    float latitude, longitude;
    std::vector<Parada*> paradas;
    std::string line;
    std::ifstream arq;
    arq.open("coordenadas.txt");

    if(!arq){
        exit(1);
    }

    while(arq >> latitude >> longitude){
        Parada *p = new Parada(index, latitude, longitude);
        paradas.push_back(p);
        index++;
    }
    arq.close();
    /*
    * Outras operações...
    */
    return 0;
}
```

Construtor para Parada

Vetor de **ponteiros** para Parada.

Ponteiros devem ser inicializados!

Mas é só isso?

Resolvendo Problemas com TADs

```
==1417== LEAK SUMMARY:
==1417==    definitely lost: 13,648 bytes in 853 blocks
==1417==    indirectly lost: 0 bytes in 0 blocks
==1417==      possibly lost: 72 bytes in 3 blocks
==1417==    still reachable: 524 bytes in 14 blocks
==1417==         suppressed: 18,066 bytes in 156 blocks
==1417== Reachable blocks (those to which a pointer was found) are not shown.
==1417== To see them, rerun with: --leak-check=full --show-leak-kinds=all
```

Resolvendo Problemas com TADs

```
Parada::Parada(int id, float latitude, float longitude){
    this->id = id;
    this->latitude = latitude;
    this->longitude = longitude;
}

int main(){
    int index = 0;
    float latitude, longitude;
    std::vector<Parada*> paradas;
    std::string line;
    std::ifstream arq;
    arq.open("coordenadas.txt");

    if(!arq){
        exit(1);
    }

    while(arq >> latitude >> longitude){
        Parada *p = new Parada(index, latitude, longitude);
        paradas.push_back(p);
        index++;
    }
    arq.close();
    /*
    * Outras operações...
    */
    for (Parada *p : paradas)
        delete p;
    return 0;
}
```

```
==1446== LEAK SUMMARY:
==1446==    definitely lost: 0 bytes in 0 blocks
==1446==    indirectly lost: 0 bytes in 0 blocks
==1446==      possibly lost: 72 bytes in 3 blocks
==1446==    still reachable: 524 bytes in 14 blocks
==1446==         suppressed: 18,066 bytes in 156 blocks
==1446== Reachable blocks (those to which a pointer was found) are not shown.
==1446== To see them, rerun with: --leak-check=full --show-leak-kinds=all
```

Resolvendo Problemas com TADs

- Ponteiros em containers: usar ou não usar?

Use Ponteiros (std::vector<...*> v)	Não Use Ponteiros (std::vector<...> v)
Você se importa com os ponteiros, e não esquecerá de os deletar ao fim do uso.	Você consegue criar todos os objetos da sua estrutura sem a necessidade de realocar memória muitas vezes.
Sua estrutura tem custo computacional alto para ser criada.	Para criar um objeto da sua estrutura você não realiza operações custosas.
Você deseja usar **polimorfismo** (cenas dos próximos capítulos…).	Não existem ponteiros para outras estruturas dentro da sua estrutura.

Referência: https://stackoverflow.com/questions/6624819/c-vector-of-objects-vs-vector-of-pointers-to-objects

Resolvendo Problemas com TADs

- **Problema II:**
  - É seu aniversário e você convida muitos amigos para uma festa. Na tentativa de agradar a todos com a música, você pede que eles enviem suas *playlists* para que você possa criar uma única com todas as músicas. Para facilitar que seus amigos encontrem as músicas na *playlist* final, você decide a manter ordenada alfabeticamente pelo nome do artista.

Resolvendo Problemas com TADs

- **TAD Música:**
  - Dado o **TAD Música** a seguir, qual a melhor estrutura para gerenciar as *playlists*?

```
struct Musica{
   std::string nome;
   std::string artista;
   std::string album;
   /*
   * Outros atributos...
   */
};
```

Resolvendo Problemas com TADs

```
int main(){
    std::ifstream joao_playlist("joao_list.csv");
    std::ifstream jose_playlist("jose_list.csv");
    std::ifstream maria_playlist("maria_list.csv");
    std::string line;
    std::list<Musica> playlist_joao, playlist_jose, playlist_maria, playlist_final;
    if(!joao_playlist || !jose_playlist || !maria_playlist){
        std::cout<<"Failed"<<std::endl;
        exit(1);
    }
    while (std::getline(joao_playlist, line)){
        parseInput(line, &playlist_joao);
    }
    while (std::getline(jose_playlist, line)){
        parseInput(line, &playlist_jose);
    }
    while (std::getline(maria_playlist, line)){
        parseInput(line, &playlist_maria);
    }
    joao_playlist.close();
    jose_playlist.close();
    maria_playlist.close();
    playlist_final.merge(playlist_joao, mergeListsSortedByArtist);
    playlist_final.merge(playlist_jose, mergeListsSortedByArtist);
    playlist_final.merge(playlist_maria, mergeListsSortedByArtist);
    /*
    * Outras operações...
    */
    return 0;
}
```

Listas de objetos do tipo **Musica**.

Função extra para processar as linhas do arquivo.

O método ***merge*** **exige** que uma função de comparação seja usada para executar a operação.

Resolvendo Problemas com TADs

Função de comparação para o método **merge**.

```
bool mergeListsSortedByArtist(const Musica& first, const Musica& second){
    int i=0;
    while ((i<first.artista.length()) && (i<second.artista.length()) ){
        if (tolower(first.artista[i])<tolower(second.artista[i]))
            return true;
        else if (tolower(first.artista[i])>tolower(second.artista[i]))
            return false;
        ++i;
    }
    return (first.artista.length() < second.artista.length());
}
```

Usamos a função "tolower()" para tratar cada letra como minúscula.

Utilizamos essa função para comparar duas *strings* e executar a operação de **merge**. É importante notar que o atributo comparado é o atributo **artista**, ou seja, a ordenação será feita em ordem lexicográfica do nome do artista.

E caso fosse pedida a ordenação por nome da música?

Resolvendo Problemas com TADs

- **Vector vs. List**
  - Embora sejam estruturas muito parecidas, cada uma tem suas vantagens.
  - Vantagens normalmente associadas ao custo computacional. **NÃO É ASSUNTO PARA ESSA DISCIPLINA**!
  - Então quando eu devo usar ***List*** e quando usar ***Vector*** ?

What we should see it that we need a sequence of elements. And the default sequence of elements in C++ is the vector. Now, because that is compact and efficient. Implementation, mapping to hardware, matters. Now, if you want to optimize for insertion and deletion - you say, 'well, I don't want the default version of a sequence. I want the specialized one, which is a list'. - **Bjarne Stroustrup**.

Ver: https://channel9.msdn.com/Events/GoingNative/GoingNative-2012/Keynote-Bjarne-Stroustrup-Cpp11-Style

Resolvendo Problemas com TADs (Map)

- **Map**
  - Estrutura amplamente utilizada para armazenar dados de forma associativa.
  - Associação feita da forma Chave → Valor.
  - Chaves **são únicas**!
  - Na distribuição C++11, o tipo **Map** é ordenado pela sua chave por *default.*
  - Acesso a qualquer elemento da estrutura facilitado. **Como**?

Referência: http://www.cplusplus.com/reference/map/map/

Resolvendo Problemas com TADs (Set)

- **Set**
  - Estrutura amplamente utilizada para armazenar dados de forma única, sem repetições.
  - Dados são armazenados obedecendo a estrutura de árvore binária de pesquisa.
  - Elementos são inseridos de maneira a manter a estrutura ordenada.

Referência: http://www.cplusplus.com/reference/set/set/

Resolvendo Problemas com TADs

- **Problema III:**
  - Você está desenvolvendo um sistema para alocar salas para disciplinas do curso de Computação do DCC. O seu programa deve informar em qual(is) sala(s) estão ofertando cada uma das disciplinas em **ordem alfabética do nome da disciplina**.

Resolvendo Problemas com TADs

- **TAD Disciplina:**
  - Dado o **TAD Disciplina** a seguir, qual a melhor estrutura para apresentar quais disciplinas estão em cada uma das salas?

```
struct Disciplina{
    std::string nome;
    std::string codigo;
    std::string local;
    /*
    * Outros atributos...
    */
}
```

Resolvendo Problemas com TADs

```
void createMap(std::string line, std::map<std::string, std::set<std::string>> *alocacao){
    std::stringstream parser(line);
    std::vector<std::string> parsedString;
    while(parser.good()){
        std::string substring;
        getline(parser, substring, ',');
        parsedString.push_back(substring);
    }
    if(alocacao->find(parsedString[0]) == alocacao->end()){
        std::set<std::string> s;
        s.insert(parsedString[2]);
        alocacao->insert({parsedString[0], s});
    }
    else{
        alocacao->at(parsedString[0]).insert(parsedString[2]);
    }
}

int main(){
    std::string line;
    std::map<std::string, std::set<std::string>> alocacao;
    std::ifstream disciplinas("disciplinas.csv");
    if(!disciplinas){
        std::cout<<"Failed"<<std::endl;
        exit(1);
    }
    while (std::getline(disciplinas, line)){
        createMap(line, &alocacao);
    }
    /*
    * Outras operações...
    */
    return 0;
}
```

Inserção de um novo elemento no *map*.

Modificação de um valor em uma chave já existente no *map*. Por que **não foi usado** o operador "[ ]" (alocacao[parsedString[0]]...) ?

Nossa estrutura usa ***strings*** como chave e um ***set*** (de ***strings***) como valor.

E para imprimir os valores do meu *map*?

Resolvendo Problemas com TADs

- **Problema IV:**
  - Você é o novo responsável pela alocação de recursos referentes aos cursos de exatas da UFMG. Para uma alocação de recursos mais favorável aos horários dos professores e alunos, você deseja saber quais disciplinas são comuns à grade de dois desses cursos.

Resolvendo Problemas com TADs

- **TAD Disciplina:**
  - Dado o **TAD Disciplina** a seguir, qual a melhor estrutura para apresentar quais disciplinas são comuns aos cursos?

```
struct Disciplina{
    std::string nome;
    std::string codigo;
    std::string local;
    /*
    * Outros atributos...
    */
}
```

Resolvendo Problemas com TADs

Quando utilizamos um set de **objetos**, precisamos de um **Functor**.

```
int main(){
    std::string line;
    std::set<Disciplina, disciplinaCmp> Curso_1, Curso_2, intersect;
    std::ifstream disciplinasCurso_1("Curso_1.csv");
    std::ifstream disciplinasCurso_2("Curso_2.csv");
    if(!disciplinasCurso_1 || !disciplinasCurso_2){
        std::cout<<"Failed"<<std::endl;
        exit(1);
    }
    while (std::getline(disciplinasCurso_1, line)){
        createSets(line, &Curso_1);
    }
    while (std::getline(disciplinasCurso_2, line)){
        createSets(line, &Curso_2);
    }
    set_intersection(Curso_1.begin(),Curso_1.end(),Curso_2.begin(),
        Curso_2.end(),std::inserter(intersect,intersect.begin()),comparator);
    /*
    * Outras operações...
    */
    return 0;
}
```

Função extra para processar as linhas do arquivo e as colocar em um Set.

Calcula a interseção entre os *sets* de **Curso_1** e **Curso_2** e armazena o resultado no *set* **intersect**. com base no comparador ***comparator***.

Resolvendo Problemas com TADs

```
struct disciplinaCmp{
    bool operator() (const Disciplina& first, const Disciplina& second) const{
        return first.nome != second.nome;
    }
};
```

```
bool comparator(const Disciplina& first, const Disciplina& second){
    return first.nome == second.nome;
}
```

Functor

Função de comparação

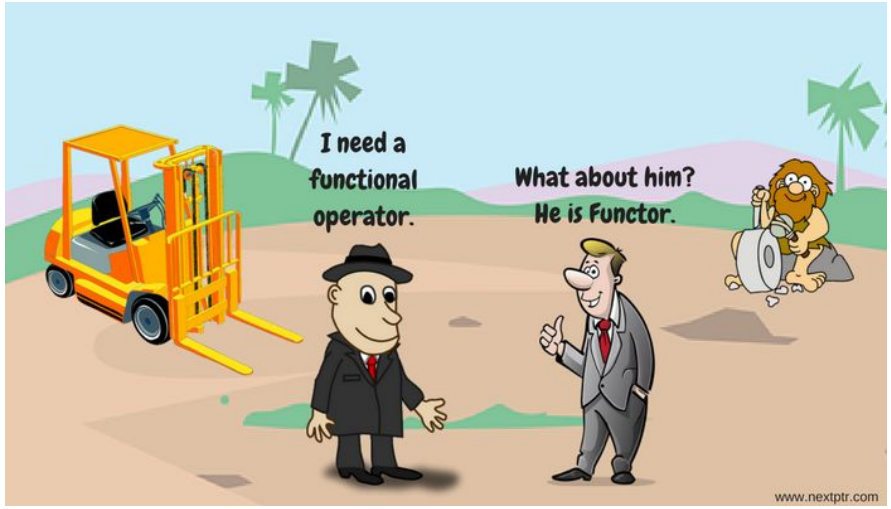
Por que eles são logicamente diferentes?

Um pouco além de TADs (Functor)

- **Functor**
  - **Problema**: Dado um conjunto de elementos de um TAD, como ordenar esse TAD? E se existir mais de um elemento numérico? E se existir mais de um elemento lexicográfico?
  - Uma solução: utilizar **Functors**!

Um pouco além de TADs (Functor)

- **Functor**
  - **Functors:** são **objetos** que funcionam como funções.
  - Por serem **objetos**, Functors podem persistir um estado e terem atributos próprios, o que em algumas vezes pode os tornar mais eficientes que funções.



Um pouco além de TADs (Functor)

Functor para comparação

- Imagine que você deseja exibir os alunos de uma disciplina com sua nota em ordem decrescente após registrar todos eles em uma estrutura.

  I) Dado o **TAD Aluno**, qual a melhor estrutura a ser utilizada nesse problema?

  II) Considere que você não pode fazer uso de nenhuma operação, apenas inserção de alunos na estrutura escolhida.

  Como imprimir a ordem decrescente pedida?

Um pouco além de TADs (Functor)

Functor para comparação

I) Como vimos anteriormente, *set* é uma estrutura interessante para armazenar elementos unitários. Cada aluno pode ser visto como um único elemento!

II) Por que não criar um Functor para ser usado ao inserir os elementos no *set* ?

```
struct Aluno{
    std::string nome;
    float nota;
    Aluno(std::string, float);
    /*
    * Outros atributos...
    */
};
```

Um pouco além de TADs (Functor)

```
struct aluno_comparator_f{
    bool operator()(const Aluno &a1, const Aluno &a2) const{
        return a1.nota > a2.nota;
    }
};
```

Note a passagem de parâmetros.
Para que serve a palavra **const**?

Criamos o Functor para comparação.

```
int main(){
    std::set<Aluno, aluno_comparator_f> alunos;
    Aluno al1 = Aluno("João", 73.2);
    Aluno al2 = Aluno("José", 65.9);
    Aluno al3 = Aluno("Maria", 88.3);
    alunos.insert(al1);
    alunos.insert(al2);
    alunos.insert(al3);
    for(Aluno aln : alunos){
        std::cout<<aln.nome<<" "<<aln.nota<<std::endl;
    }
    return 0;
}
```

Indicamos que o Functor criado será usado na estrutura. Lembra-se do Problema IV?

Suponha que você tem os arquivos: main.cpp, Aluno.cpp e Aluno.hpp.
Qual desses arquivos é o mais indicado para conter o **Functor**?

Um pouco além de TADs (Iterators)

- **Iterators**
  - Iterators são **objetos**.
  - Ao apontar para um elemento dentro de um conjunto desses elementos (por exemplo, um elemento dentro de um *container*), *iterators* podem iterar sobre esses elementos usando **pelo menos** dois operadores: **incremento** (++) e **derreferência** (*(...)).
  - Você pode implementar o seu próprio iterator!
  - **Qual seria a estrutura mais simples de um *iterator* ?**

Referência: http://www.cplusplus.com/reference/iterator/

Um pouco além de TADs (Iterators)

```
struct Number{
    int decimal;
    int binary;
    Number(int, int);
    /*
    * Outros atributos...
    */
};

int main(){
    Number one = Number(1,1);
    Number two = Number(2,10);
    Number three = Number(3,11);
    Number four = Number(4,100);
    Number five = Number(5,101);
    std::vector<Number>::iterator it;
    std::vector<Number> numbers = {one, two, three, four, five};
    for(it = numbers.begin(); it != numbers.end(); it++){
        std::cout<<it->decimal<<"\t"<<it->binary<<std::endl;
    }
    return 0;
}
```

O *iterator* é do mesmo tipo que a estrutura sob a qual irá iterar.

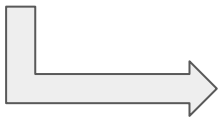
A operação de incremento **deve ser definida** para o *iterator*.

Um pouco além de TADs (Iterators)

```
struct Number{
    int decimal;
    int binary;
    Number(int, int);
    /*
    * Outros atributos...
    */
};

int main(){
    Number one = Number(1,1);
    Number two = Number(2,10);
    Number three = Number(3,11);
    Number four = Number(4,100);
    Number five = Number(5,101);
    std::vector<Number>::iterator it;
    std::vector<Number> numbers = {one, two, three, four, five};
    it = numbers.begin();
    std::cout<<it->decimal<<"\t"<<it->binary<<std::endl;
    it += 3;
    std::cout<<it->decimal<<"\t"<<it->binary<<std::endl;
    return 0;
}
```

Com o *iterator* na posição inicial da estrutura. O que a próxima linha imprimirá?



Adicionamos 3 ao *iterator*. O que a próxima linha imprimirá?

Um pouco além de TADs (Iterators)

Achou que eu me esqueci da impressão dos elementos do Problema III?

Um pouco além de TADs (Iterators)

```
void printMap(std::map<std::string, std::set<std::string>> *alocacao){
    std::map<std::string, std::set<std::string>>::iterator it;
    for(it = alocacao->begin(); it != alocacao->end(); it++){
        std::cout<<it->first<<" Salas: ";
        for (std::string sala : it->second){
            std::cout<<sala<<", ";
        }
        std::cout<<std::endl;
    }
}
```

Iterator para o nosso ***map***.

O *iterator* começa no início do *map* e o percorre até ser diferente do fim. Detalhe para

Nossos valores estão em uma estrutura com múltiplos valores. É necessário iterar sobre ela também! Por que **não usar** um iterator?

O que representa **it →first** ?

O que representa **it →second** ?

UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Vítor Machado Guilherme Barros
vitorbarros@dcc.ufmg.br

DCC
DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO