Bruno Crespo Ferreira(202120489), Gustavo Soares Silva(202120103), Vinícius de Oliveira Fabiano(202120494).
Paula Christina Figueira Cardoso.
Introdução aos Algoritmos
17 de abril de 2022

Relatório Trabalho Prático

Introdução

Como objetivo deste projeto prático tivemos a elaboração de um programa onde o usuário possa cadastrar, ordenar e fazer a manutenção dos dados sobre o tema escolhido: Jogos. No programa, nos baseamos nas matérias passadas em Introdução aos Algoritmos e também em nossas pesquisas sobre os assuntos necessários para a elaboração do programa.

Desenvolvimento

Para a utilização do programa a grande base foi a utilização de um registro, criando um vetor dele para o armazenamento dos jogos e outras funçoẽs. Nesse registro, temos a variável “id” que é o identificador do jogo sendo uma variável crescente de um por um e ela é utilizada na amostragem da lista de jogos presentes no programa, sendo uma variável que não muda durante a utilização do mesmo. Ainda temos a existência de uma variável “nome” que armazena o nome do jogo; uma variável “ano_lancamento” que armazena o ano de lançamento do jogo; uma variável “plataforma” que armazena a plataforma ou plataformas que há a existência deste jogo; uma variável “descricao” que armazena uma descrição feita pelo usuário sobre o jogo; e uma variável "valida" do tipo booleana sendo considerada um flag, onde se esse variável for verdadeira(true) o jogo existe e se for falsa(false) ele não existe no programa. Inicialmente pode parecer que o jogo com “id=1” exista mas isso não afeta o programa, tal jogo é sobrescrito.

```
 4  struct Jogo{
 5
 6      int id = 1;
 7
 8      string nome;
 9      int ano lancamento;
10      string plataforma;
11      string descricao;
12
13      bool valido = true;
14  };
```

Para a inserção de um novo jogo foi criado um vetor do registro “Jogos” alocado dinamicamente e foi feito uma função desse vetor do registro com o nome “inserir”, passando os valores do vetor do registro “Jogos”, o tamanho do vetor(passado por parâmetros), a posição que o usuário irá inserir no vetor(passado por parâmetros) e o campo atual. Nessa função, primeiramente foi criado um condicional que tem como função verificar se existe algum item que foi excluído e está localizado no final do vetor (pela ordenação), se existir o próximo item é inserido nessa posição. Para a inserção do nome do jogo foi utilizado o “getline” em conjunto com “cin.ignore()” para ignorar os espaços e um condicional usando os números da tabela ascii para mudar a primeira letra do nome para maiúscula caso necessário. Na inserção do ano de lançamento foi feita uma função que caso o usuário digite letras ao invés de números, pede uma nova inserção da forma correta. Na inserção da plataforma utilizada pelo jogo também foi utilizado o “getline” juntamente com o “cin.ignore()” e mais um condicional para mudar a primeira letra do nome para maiúscula caso necessário. Na inserção da descrição do jogo também foi utilizado o ”getline”. A cada uso dessa função inserção de um novo jogo é feito um incremento de mais 1 na variável posição (pos) para a inserção de mais um novo jogo. O vetor jogos é ordenado após a inserção do novo jogo à ele, tal ordenação depende do campo atual selecionado, do próprio vetor e a posição inicial e final dele, fazendo a chamada da função “quicksort” com “jogos, tamanho=0, posição-1 e o campo atual”. Retornando o novo vetor jogos com o novo jogo inserido e ordenado corretamente.

```
234 Jogo* inserir(Jogo *jogos, int &tam, int &pos, string campoAtual){
235
236     if(pos != 0 and !jogos[pos-1].valido){
237         pos--;
238         jogos[pos].valido = true;
239     }
240     else{
241         if(pos == tam)
242             jogos = aumentaVetor(jogos, tam);
243         jogos[pos].id += pos;
244     }
245
246     cout << "Insira o nome do jogo: ";
247     cin.ignore();
248     getline(cin, jogos[pos].nome);
249
250     if(jogos[pos].nome[0] >= 97 and jogos[pos].nome[0] <= 122)
251         jogos[pos].nome[0] -= 32;
252
253     cout << "Insira o ano de lancamento do jogo: ";
254     while(!(cin >> jogos[pos].ano_lancamento)){
255         cin.clear();
256         cin.ignore(100, '\n');
257         cout << "Insira um ano valido: ";
258     }
```

```
260     cout << "Insira a plataforma do jogo: ";
261     cin.ignore();
262     getline(cin, jogos[pos].plataforma);
263     if(jogos[pos].plataforma[0] >= 97 and jogos[pos].plataforma[0] <= 122)
264         jogos[pos].plataforma[0] -= 32;
265
266     cout << "Insira a descricao sobre o jogo: ";
267     getline(cin, jogos[pos].descricao);
268
269     pos++;
270
271     quicksort(jogos, 0, pos-1, campoAtual);
272
273
274     return jogos;
275 }
```

Caso for necessário aumentar o tamanho do vetor Jogos, que começa com 5 posições, é chamado a função “aumentaVetor”, que passa o vetor do registro e o tamanho(passado por parâmetro). Dentro dessa função, é criado uma variável “tam2” que representa o novo tamanho do vetor, que tem um aumento de 20% e com isso é criado um novo vetor dinamicamente com o novo tamanho; logo após isso é criado um condicional para passar os jogos do vetor antigo para o novo já aumentado e o tamanho antigo recebe o valor de seu novo tamanho, como ele está sendo passado por referência, essa alteração é válida para todo o código. Logo, também é feita a desalocação do vetor de jogos antigo e é retornado o vetor “Jogos” realocado.

```
217  Jogo* aumentaVetor(Jogo *jogos, int &tam){
218
219      int tam2 = tam + (int)tam*0.2;
220
221
222      Jogo* novo = new Jogo[tam2];
223
224      for(int i = 0;i < tam;i++)
225          novo[i] = jogos[i];
226
227      tam = tam2;
228      delete[] jogos;
229
230      return novo;
231  }
```

Caso o usuário desejar excluir um jogo do programa é chamado a função booleana(pode ser falso ou verdadeiro) “excluir” onde foi passado o vetor de registro”Jogos”, a posição e a posição atual. Dentro dessa função é criada uma variável chamada “Id” que recebe o valor da função “procuraId” e um condicional onde se a variável “Id” for igual a -1, ele retorna a função booleana como falso. Após isso, é criada uma variável chamada “escolha” onde o usuário confirma se quer mesmo ou não excluir o jogo do programa, se a resposta for sim é avisado ao usuário que o jogo foi excluído e a flag desse jogo é colocada como falsa e é feita a chamada da função “quicksort” com “jogos, tamanho=0, posição-1 e o campo atual” e retornando o valor verdadeiro para a função booleana; se a resposta for não é avisado ao usuário que o jogo não foi excluído e é retornado o valor falso para a função booleana.

```
193  bool excluir(Jogo *jogos, int pos, string atual){
194
195      int id = procuraId(jogos, pos);
196
197      if(id == -1) return false;
198
199      string escolha;
200      cout << "Voce tem certeza que quer deletar " << jogos[id].nome << "? (s / n) ";
201      cin >> escolha;
202
203      if(escolha == "s"){
204
205          jogos[id].valido = false;
206          cout << "O jogo foi deletado" << endl;
207          quicksort(jogos, 0, pos-1, atual);
208          return true;
209      }
210      else{
211          cout << "O jogo nao foi deletado" << endl;
212          return false;
213      }
214  }
```

Caso o usuário desejar alterar alguma informação sobre os jogos já cadastrados é chamado a função booleana “alterar” onde foi passado o vetor do registro e a posição. Dentro dessa função, é criada uma variável id que recebe o valor da função “procuraId” e um condicional onde se a variável “Id” for igual a -1, ele retorna a função booleana como falso. Após isso é criada uma variável booleana “controle” que é igual a verdadeiro; o programa pergunta qual campo do jogo o usuário quer alterar, se o usuário digitar números ao invés de letras o programa pergunta novamente o campo, caso a resposta seja 1 o programa pede que o usuário escreva o novo nome do jogo usando novamente o “getline” e o cin.ignore” e caso for necessário mudar a primeira letra de minúscula para maiúscula ele o faz, retornando a confirmação ao usuário que o nome foi trocado e fazendo a variável controle se tornar falsa parando a repetição controle e faz o uso do “break” para a repetição da condicional dos números; caso a resposta seja 2 o programa pede que o usuário escreva o ano de lançamento do jogo, se o usuário digitar letras ao invés de números o programa pergunta novamente o campo, se for uma resposta válida o programa retorna a confirmação da troca ao usuário e a variável controle se torna falsa parando a repetição controle e faz o uso do “break” para a repetição da condicional dos números; caso a resposta seja 3 o programa pede que o usuário escreva a plataforma usada pelo jogo, trocando a primeira letra para maiúsculo caso o necessário, após isso o programa retorna a confirmação da troca ao usuário e a variável controle se torna falsa parando a repetição controle e faz o uso do “break” para a repetição da condicional dos números; caso a resposta seja 4 o programa pede que o usuário escreva a nova descrição do jogo, após isso o programa retorna a confirmação da troca ao usuário e a variável controle se torna falsa parando a repetição controle e faz o uso do “break” para a repetição da condicional dos números; caso o usuário escreve algum número invalido o programa pede que o usuário digite um número válido; quando alguma das condições for satisfeita o programa retorna a função booleana “alterar” como verdadeira. No final da função temos um loop para que se o usuário desejar possa fazer mais alguma alteração no jogo já selecionado.

```
128 bool alterar(Jogo *jogos, int pos){
129     int id = procuraId(jogos, pos);
130     if(id == -1) return false;
131     bool controle = true;
132     cout << "Voce escolheu o jogo " << jogos[id].nome << endl;
133     cout << "Escolha um campo para alterar:" << endl;
134     cout << "1 - Nome" << endl;
135     cout << "2 - Ano de Lancamento" << endl;
136     cout << "3 - Plataforma" << endl;
137     cout << "4 - Descricao" << endl;
138     do{
139         int opcao;
140         while(!(cin >> opcao)){
141             cin.clear();
142             cin.ignore(100, '\n');
143             cout << "Digite uma opcao valida: ";
144         }
145         switch(opcao){
146             case 1:
147                 cout << "Insira o nome do jogo: ";
148                 cin.ignore();
149                 getline(cin, jogos[id].nome);
150                 if(jogos[id].nome[0] >= 97 and jogos[id].nome[0] <= 122)
151                     jogos[id].nome[0] -= 32;
152                 cout << "Alteracao feita! Nome atual: " << jogos[id].nome << endl;
153                 controle = false;
154                 break;
```

```
155             case 2:
156                 cout << "Insira o ano de lancamento do jogo: ";
157                 while(!(cin >> jogos[id].ano lancamento)){
158                     cin.clear();
159                     cin.ignore(100, '\n');
160                     cout << "Insira um ano valido: ";
161                 }
162                 cout << "Alteracao feita! Ano de lancamento atual: " << jogos[id].ano lancamento << endl;
163                 controle = false;
164                 break;
165             case 3:
166                 cout << "Insira a plataforma do jogo: ";
167                 cin.ignore();
168                 getline(cin, jogos[id].plataforma);
169                 if(jogos[id].plataforma[0] >= 97 and jogos[id].plataforma[0] <= 122)
170                     jogos[id].plataforma[0] -= 32;
171                 cout << "Alteracao feita! Plataforma atual: " << jogos[id].plataforma << endl;
172                 controle = false;
173                 break;
174             case 4:
175                 cout << "Insira a descricao sobre o jogo: ";
176                 cin.ignore();
177                 getline(cin, jogos[id].descricao);
178                 cout << "Alteracao feita! Descricao atual: " << jogos[id].descricao << endl;
179                 controle = false;
180                 break;
```

```
195             default:
196                 cout << "Opcao invalida. Escolha novamente: ";
197                 break;
198         }
199         if(opcao > 0 and opcao < 5){
200             cout << "Gostaria de fazer mais alguma alteracao no jogo " << jogos[id].nome << "? (s/n)";
201             string novaAlteracao;
202             cin >> novaAlteracao;
203             if(novaAlteracao == "s")
204                 controle = true;
205         }
206     }while(controle);
207
208     return true;
209 }
```

A função “procuraId” que recebe o vetor do registro “Jogos” e a posição é usada nas funções “alterar” e “excluir” para ter a confirmação de que no identificador exista um jogo e

que ele seja passado para as outras duas funções. Dentro da função o programa se certifica que o usuário escreva um número inteiro e que não seja alguma palavra ou letra, retornando a posição do vetor que ele se encontra se for um número válido e, se não for, retorna ao usuário que o identificador não está presente e retorna as demais funções o valor -1.

```
112  int procuraId(Jogo *jogos, int pos){
113      int idProcurado;
114      cout << "Por favor insira o id do jogo: ";
115      while(!(cin >> idProcurado)){
116          cin.clear();
117          cin.ignore(100, '\n');
118          cout << "Digite um inteiro: ";
119      }
120      for(int i=0; i<pos; i++)
121          if(idProcurado == jogos[i].id and jogos[i].valido)
122              return i;
123
124      cout << "O id digitado nao esta presente nos jogos." << endl;
125      return -1;
126  }
```

Caso o usuário queira mudar o campo que o vetor “Jogos” será ordenado é chamado a função “alterarOrdenacaoo” que recebe o vetor do registro “Jogos”, a posição e a ordenação atual. Dentro do programa é criada uma variável booleana “controle” que é igual a verdadeiro. Após isso o programa pede que o usuário escolha o campo que ele deseja ordenar de acordo com o campo escolhido, se o usuário escrever uma opção inválida ele retorna pedindo para que o usuário escreva uma opção válida. Caso a resposta seja 1, a ordenação atual se torna o “identificador” e a variável “controle” se torna falsa parando a repetição controle e faz o uso do “break” para a repetição da condicional dos números; caso a resposta seja 2, a ordenação atual se torna o “nome” e a variável “controle” se torna falsa parando a repetição controle e faz o uso do “break” para a repetição da condicional dos números; caso a resposta seja 3, a ordenação atual se torna o “ano de lançamento” e a variável “controle” se torna falsa parando a repetição controle e faz o uso do “break” para a repetição da condicional dos números; caso a resposta seja 4, a ordenação atual se torna a “plataforma”e a variável “controle” se torna falsa parando a repetição controle e faz o uso do “break” para a repetição da condicional dos números; caso o usuário não escreve um número válido o programa avisa do erro e pede que o usuário escreva um número válido. No final da função é utilizado a chamada da função

“quicksort” com “jogos, tamanho=0, posição-1 e o campo atual” e caso uma das condições seja satisfeita, retorna a ordenação atual escolhida pelo usuário.

```
70  string alterarOrdenacao(Jogo *jogos, int pos, string atual){
71      bool controle = true;
72      cout << "Escolha um campo para ordenar a lista:" << endl;
73      cout << "1 - Identificador" << endl;
74      cout << "2 - Nome" << endl;
75      cout << "3 - Ano de Lancamento" << endl;
76      cout << "4 - Plataforma" << endl;
77      do{
78          int opcao;
79          while(!(cin >> opcao)){
80              cin.clear();
81              cin.ignore(100, '\n');
82              cout << "Digite uma opcao valida: ";
83          }
84          switch(opcao){
85              case 1:
86                  atual = "identificador";
87                  controle = false;
88                  break;
89              case 2:
90                  atual = "nome";
91                  controle = false;
92                  break;
93              case 3:
94                  atual = "ano de lancamento";
95                  controle = false;
96                  break;
97              case 4:
98                  atual = "plataforma";
99                  controle = false;
100                 break;
101             default:
102                 cout << "Opcao invalida. Escolha novamente" << endl;
103                 break;
104         }
105     }while(controle);
106
107     quicksort(jogos, 0, pos-1, atual);
108
109     return atual;
110 }
```

A função “quicksort” que recebe o vetor do registo, o início do vetor, o final do vetor e o campo de ordenação atual é usada nas funções “inserir”, “excluir”, “alterar” e “alterarOrdenação” para ter uma melhor ordenação. Dentro da função há um condicional onde se o início do vetor for maior do que o final dele, a função usa um retorno somente para encerrar a recursão. Após isso temos a criação da variável “pivo”, necessária para a ordenação

desse tipo de função, que recebe o valor dado pela função “particao”; logo temos a chamada da própria função “quicksort” recebendo o vetor do registro, o início do vetor, o pivô-1 e o campo atual de ordenação e também temos a chamada mais um vez da própria função ”quicksort” porém recebendo o vetor do registro, o pivô+1, o final do vetor e o campo de ordenação atual.

```
60  void quicksort(Jogo *jogos, int ini, int fim, string campo){
61      if(ini >= fim) return;
62
63      int pivo = particao(jogos, ini, fim, campo);
64
65      quicksort(jogos, ini, pivo-1, campo);
66      quicksort(jogos, pivo+1, fim, campo);
67  }
```

A função “particao” faz parte da função da ordenação “quicksort” para repartir o vetor onde recebe o vetor do registro, o início do vetor, o final do vetor, e o campo de ordenação atual. Dentro da função é criada uma variável “i” que recebe o início do vetor-1. Após isso, há um condicional onde enquanto o inicio for menor que o final e se a função “ordenarPorCampo” for verdadeira há uma troca do vetor com posição igual a variável “i” e com uma variável “j” criada com o valor do início; caso a função “ordenarPorCampo” for falsa há uma troca do vetor com a posição igual a variável “i” com o fim; no final da condicional é retornado a variável “i”.

```
46  int particao(Jogo *jogos, int ini, int fim, string campo){
47      int i = ini-1;
48
49      for(int j=ini; j<fim; j++)
50          if(ordenarPorCampo(jogos[j], jogos[fim], campo)){
51              i++;
52              swap(jogos[i], jogos[j]);
53          }
54      i++;
55      swap(jogos[i], jogos[fim]);
56
57      return i;
58  }
```

A função “ordenarPorCampo” que recebe o vetor do registro “jogo1”, o vetor do registro “jogo2”, e o campo de ordenação. Essa função é utilizada para retornar como o vetor

será ordenado, dependendo do campo selecionado. Dentro da função temos condicionais que conforme o campo selecionado pelo usuário retornam o valor para o campo selecionado.

```
34  bool ordenarPorCampo(Jogo jogo1, Jogo jogo2, string campo){
35
36      if(campo == "identificador")
37          return ((jogo1.id < jogo2.id and jogo1.valido) or (!jogo2.valido));
38      else if(campo == "nome")
39          return ((jogo1.nome < jogo2.nome and jogo1.valido) or (!jogo2.valido));
40      else if(campo == "ano de lancamento")
41          return ((jogo1.ano lancamento < jogo2.ano lancamento and jogo1.valido) or (!jogo2.valido));
42      else
43          return ((jogo1.plataforma < jogo2.plataforma and jogo1.valido) or (!jogo2.valido));
44  }
```

Caso o usuário queira listar os jogos presentes dentro do programa, é usada a função "listar" que recebe o vetor do registro e a posição. Dentro da função há uma repetição com uma condicional para o jogo ser válido, onde essa validez só ocorre se sua flag for verdadeira, logo não serão mostrados jogos excluídos do programa; caso sua flag for verdadeira o programa mostra os dados dos jogos presentes no programa.

```
18  void listar(Jogo *jogos, int pos){
19      cout << "------ SEUS JOGOS ------" << endl;
20
21      for(int i=0; i<pos; i++)
22          if(jogos[i].valido){
23              cout << "Id: " << jogos[i].id << endl;
24              cout << "Nome: " << jogos[i].nome << endl;
25              cout << "Ano de lancamento: " << jogos[i].ano lancamento << endl;
26              cout << "Plataformas: " << jogos[i].plataforma << endl;
27              cout << "Descricao: " << jogos[i].descricao << endl;
28              cout << "Posicao: " << i << endl;
29              cout << endl;
30          }
31  }
```

Na função principal, temos a variável "tam" que é o tamanho atual do vetor do registro e é inicializada com o valor 5; a variável "pos" que é a posição do jogo que o usuário irá inserir, a cada jogo inserido seu valor é somado em 1 e é inicializada com valor 0; a variável "nJogos" que é o número de jogos que o usuário tem atualmente e é inicializado com valor 0; a variável "campoOrdenacaoAtual" que é a ordenação atual do vetor jogos e é inicializada com "identificador". É criado também um vetor do registro "Jogo" que é alocada dinamicamente, uma variável "jogos" , uma variável booleana "controle" que é iniciada como verdadeira e uma variável também booleana "excluido", juntamente com o menu para a melhor utilização do usuário. É criada uma variável booleana "aux" inicializada com

verdadeiro e uma variável inteira “opcao” para o usuário escrever qual opção ele deseja utilizar, caso a resposta seja 1, a opção escolhida foi a de inserção de um novo jogo no programa, fazendo com que a variável “jogos” receba o valor dado pela função “inserir”, na variável “nJogos” é somado mais um e a variável “aux” se torna falsa parando a repetição controle e faz o uso do “break” para a repetição da condicional dos números; caso a resposta seja 2, a opção escolhida foi a de exclusão, fazendo com que a variável “excluido” receba o valor dado pela função “excluir” e se caso a variável “excluido” for verdadeira na variável ”nJogos” é subtraído 1, chegando ao final da condicional usando o “break” para para-la; caso a resposta seja 3, a opção escolhida foi a de listar os jogos, dentro dessa condicional temos outra estrutura condicional para se a variável “nJogos” seja 0, seja avisado ao usuário que não há jogos cadastrados, se não for igual a 0 é chamado a função ”listar” e temos mais uma estrutura condicional que se caso “nJogos” seja igual a 1 é retornada uma mensagem para o usuário e se não for é mostrado o valor de “nJogos”, sendo utilizado o “break” para parar; caso a resposta seja 4, a opção escolhida foi alterar algum dado de um jogo selecionado, dentro dessa condicional temos outra estrutura condicional onde se a variável “nJogos” for igual a 0 é retornada uma mensagem avisando ao usuário que não há jogos cadastrados, se não for igual a zero é feita outra condicional com a função booleana “alterar” com a variável “aux” se tornando falsa, sendo utilizado o “break” para parar; caso a resposta seja 5, a opção escolhida foi alterar a ordem da lista conforme a ordenação escolhida pelo usuário, nele a variável “campoOrdenacaoAtual” recebe o valor da função “alterarOrdenacao” e uma estrutura condicional para se a variável “nJogos” seja maior que 0, seja chamado a função “listar”, usando o “break” para parar a parar; caso a resposta seja 0, o programa agradece ao usuário por utilizá-lo, a variável “controle” se torna falsa e é feito o uso do “break” para a parada da repetição; caso o usuário não digite uma resposta válida o programa pede que ele digite novamente mas com um valor válido e é feito o uso do “break” para parar a repetição. No final da função principal, é desalocado da memória o vetor jogos.

```
292  int main() {
293
294      int tam = 5, pos = 0, nJogos=0;
295      string campoOrdenacaoAtual = "identificador";
296      Jogo *jogos = new Jogo[tam];
297      bool controle = true;
298      do{
299          system("clear");
300          cout << "Tamanho do vetor atual: " << tam << endl;
301          cout << "Escolha uma opcao:" << endl;
302          cout << "1 - Adicionar um jogo" << endl;
303          cout << "2 - Excluir um jogo" << endl;
304          cout << "3 - Listar todos os jogos. Numero de jogos atualmente: "<< nJogos << endl;
305          cout << "4 - Alterar um dos dados de algum jogo" << endl;
306          cout << "5 - Alterar a ordem da lista. Campo de ordenacao atual: " << campoOrdenacaoAtual << endl;
307          cout << "0 - Sair do programa" << endl;
308          int opcao = -1;
309          bool aux = true, excluido;
310          while(!(cin >> opcao) or !(opcao >= 0 and opcao <= 5)){
311              cin.clear();
312              cin.ignore(10000, '\n');
313              cout << "Digite uma opcao valida: ";
314          }
315          switch(opcao){
```

```
315          switch(opcao){
316              case 1:
317                  jogos = inserir(jogos, tam, pos, campoOrdenacaoAtual);
318                  aux = false;
319                  nJogos++;
320                  break;
321              case 2:
322                  excluido = excluir(jogos, pos, campoOrdenacaoAtual);
323                  if(excluido) nJogos--;
324                  break;
325              case 3:
326                  if(nJogos == 0)
327                      cout<< "Nenhum jogo no estoque. Compre algum jogo imediatamente!!!" << endl << endl;
328                  else{
329                      listar(jogos, pos);
330                      if(nJogos == 1)
331                          cout << "Voce consegue se divertir com apenas um jogo?" << endl << endl;
332                      else
333                          cout << "Que legal! Voce tem " << nJogos << " jogos!" << endl << endl;
334                  }
335                  break;
336              case 4:
337                  if(nJogos == 0)
338                      cout<< "Nenhum jogo no estoque. Compre algum jogo imediatamente!!!" << endl << endl;
339                  else
340                      if(alterar(jogos, pos, campoOrdenacaoAtual)) aux = false;
341                  break;
342              case 5:
```

```
342              case 5:
343                  campoOrdenacaoAtual = alterarOrdenacao(jogos, pos, campoOrdenacaoAtual);
344                  if(nJogos > 0) listar(jogos, pos);
345                  break;
346              case 0:
347                  cout << "Obrigado por utilizar nosso programa!" << endl;
348                  controle = false;
349                  break;
350              default:
351                  cout << "Selecione uma opcao valida!" << endl;
352                  break;
353          }
354          cout << "Pressione enter para continuar...";
355          if(aux) cin.ignore();
356          cin.get();
357
358      }while(controle);
359
360      delete[] jogos;
361
362      return 0;
363  }
364
```

Conclusão

Com este projeto prático percorremos por diversas estruturas básicas de programação e aprendemos a usá-las de uma maneira mais real, voltada para um usuário. Este desenvolvimento é de extrema importância para a revisão e aperfeiçoamento de toda a matéria de Introdução aos Algoritmos que estudamos até o momento, pois incentiva a busca de novos conhecimentos - no caso deste trabalho, o método de ordenação - e a correção de possíveis erros, além de aprimorar a lógica em programação dos estudantes.