Universidade Federal de São Carlos
Departamento de Computação

Disciplina: Estruturas de Dados
Professor Roberto Ferrari

JOGO 2- DCkinator

Integrantes:
Bruna Zamith
João Victor Pacheco
Marcos Faglioni
Rodrigo Salmen

Desenvolvedores:

Bruna Zamith	bruna.zamith@hotmail.com
João Victor Pacheco	jvp1805@gmail.com
Marcos Faglioni	marcosfagli@hotmail.com
Rodrigo Salmen	rodrigosalmen2012@hotmail.com

OBJETIVOS

A proposta inicial deste jogo era a de implementar um jogo no estilo Akinator, só que ao invés de haver uma “inteligência” que descobre o que o usuário está pensando, o DCkinator funciona como uma espécie de quizz. Deste modo, o jogador deve optar por uma dentre as duas opções de respostas para cada pergunta. No final, é exibida uma tela com o resultado, o qual visa direcionar a área da computação de maior afinidade do jogador.

METODOLOGIA

Para desenvolver as TAD's deste projeto, utilizamos a linguagem C++. Implementamos inicialmente uma classe Nó e um tipo ponteiro para nó (ÁrvoreBinária).
Um nó é uma estrutura utilizada como elemento de uma Árvore, e que é composta por um campo de informação, um ponteiro para outro nó à esquerda e um ponteiro para outro nó à direita. Já a Árvore Binária é um conjunto de nós, que seguem uma hierarquia. Por ser binária, cada nó pode ter no máximo dois filhos. Ainda, a árvore utilizada pode ser considerada de busca no que diz respeito à função "insere".
A árvore utilizada conta com os seguintes parâmetros e funções:

int info	retorna a informação do Nó
Node* dir	ponteiro para o nó filho direito
Node* esq	ponteiro para o nó filho esquerdo
Node()	construtor padrão de um nó
Node(int i)	construtor de um nó que recebe o info
typedef Node* ArvoreBinaria	tipo ArvoreBinária
busca(ArvoreBinaria R, int x)	busca se x está na ArvoreBinaria R
void imprimeTodosPreOrdem(ArvoreBinaria R)	imprime os elementos de da ArvoreBinaria R pré-ordem
void imprimeTodosInOrdem(ArvoreBinaria R)	imprime os elementos de da ArvoreBinaria R in-ordem
int soma(ArvoreBinaria R)	soma os elementos da ArvoreBinaria R
int numeroUnicoFilho(ArvoreBinaria R)	número de nós com um único filho
void insereNo(ArvoreBinaria &R, int x)	insere um nó com info x em R
bool ehfinal(Node *novo)	verifica se o nó não tem filhos

Na imagem a seguir podemos ver a implementação da classe Pilha, do tipo ArvoreBinaria e suas funções.

```
#pragma once
#include<iostream>


//##############CLASSE NÓ##############
class Node {
public:
        int info; //informação
        Node* dir; //direita
        Node* esq; //esquerda
        Node() { //construtor padrão
                info = NULL;
                dir = NULL;
                esq = NULL;
        }
        Node(int i) { //construtor com info
                info = i;
                dir = NULL;
                esq = NULL;
        }
};

//##############TIPO PONTEIRO PARA NÓ##############
typedef Node* ArvoreBinaria;

//##############X ESTÁ NA ARVORE?##############
bool busca(ArvoreBinaria R, int x) {
        if (R == NULL)
                return false;
        else if (R->info == x)
                return true;
        else if (R->info > x)
                return busca(R->esq, x);
        else
                return busca(R->dir, x);
}

//##############IMPRIME TODOS PRÉ-ORDEM##############
void imprimeTodosPreOrdem(ArvoreBinaria R) {
        if (R != NULL) {
                cout << R->info << endl;
                imprimeTodosPreOrdem(R->esq);
                imprimeTodosPreOrdem(R->dir);
        }
}

//##############IMPRIME TODOS IN ORDEM##############
void imprimeTodosInOrdem(ArvoreBinaria R) {
        if (R != NULL) {
                imprimeTodosInOrdem(R->esq);
                cout << R->info << endl;
                imprimeTodosInOrdem(R->dir);
        }
}

//##############SOMA DOS ELEMENTOS##############
int soma(ArvoreBinaria R) {
        if (R == NULL)
                return 0;
        else {
                return R->info + soma(R->dir) + soma(R->esq);
        }
}
```

```
//###############INSERE NÓ###############
void insereNo(ArvoreBinaria &R, int x) {
        ArvoreBinaria P;
        if (R == NULL) {
                P = new Node;
                P->info = x;
                P->dir = NULL;
                P->esq = NULL;
                R = P;
                P = NULL;
        }
        else if (R->info > x) {
                insereNo(R->esq, x);
        }
        else if (R->info < x) {
                insereNo(R->dir, x);
        }
}

//###############É FINAL###############
bool ehfinal(Node *novo) {
        if (novo->dir == NULL && novo->esq == NULL)
                return true;
        else
                return false;
}
```

DESENVOLVIMENTO

O jogo foi desenvolvido em linguagem C++, com a biblioteca de interface gráfica SFML.
A primeira função chamada na main é a de menu inicial:

Os controles são feitos com a tecla para cima (↑) e para baixo (↓). Para selecionar uma opção, o jogador deve pressionar barra de espaço.
Ao selecionar “Play”, o jogo é inicializado. Cada tela do jogo chama uma pergunta correspondente ao nó da árvore e as duas respostas possíveis. Se o jogador selecionar a primeira opção, o ponteiro “Atual” aponta para o próximo nó a esquerda. Se o jogador selecionar a segunda opção, o ponteiro “Atual” aponta para o próximo nó a direita.

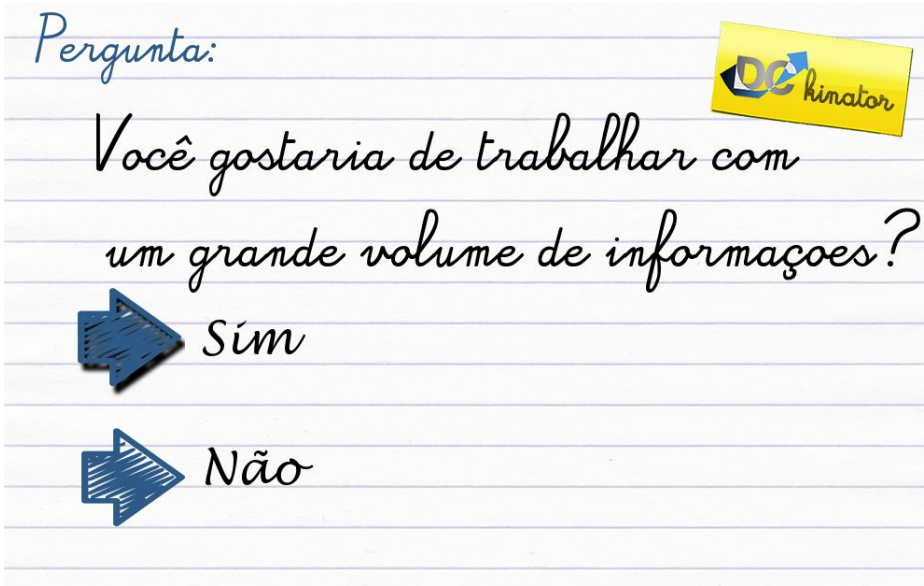
Os nós foram inseridos manualmente, com a função insere (árvore binária de busca). Deste modo, cada pergunta e tela de resultado tem um número que a representa.

```
ArvoreBinaria R;
R = NULL;
ArvoreBinaria Atual;
insereNo(R, 20);
insereNo(R, 10);
insereNo(R, 40);
insereNo(R, 6);
insereNo(R, 15);
insereNo(R, 22);
insereNo(R, 50);
insereNo(R, 4);
insereNo(R, 8);
insereNo(R, 12);
insereNo(R, 11);
insereNo(R, 14);
insereNo(R, 16);
insereNo(R, 21);
insereNo(R, 24);
insereNo(R, 45);
insereNo(R, 55);
insereNo(R, 3);
insereNo(R, 5);
insereNo(R, 7);
insereNo(R, 9);
insereNo(R, 23);
insereNo(R, 26);
insereNo(R, 25);
insereNo(R, 30);
Atual = R;
```

Na main, o processos estão ocorrendo da seguinte forma:
Verifica a info para que o Atual aponta -> Chama a tela de pergunta correspondente -> Espera o usuário selecionar a opção (armazena em “resposta”) -> Altera a posição do Atual

```
if (tela == 1) { //tela após o play
        resposta = telapergunta1(window);
        if (resposta == 1) {
                Atual = Atual->esq;
                resposta = telapergunta2(window);
                if (resposta == 1) {
                        Atual = Atual->esq;
                        resposta = telapergunta3(window);
                        if (resposta == 1) {
                                Atual = Atual->esq;
                                resposta = telapergunta4(window);
                                if (resposta == 1) {
                                        Atual = Atual->esq;
                                }
                                else {
                                        Atual = Atual->dir;
                                }
                        }
                        else {
                                Atual = Atual->dir;
                                resposta = telapergunta5(window);
                                if (resposta == 1) {
                                        Atual = Atual->esq;
                                }
                                else {
                                        Atual = Atual->dir;
                                }
                        }
                }
                else {
                        Atual = Atual->dir;
                        resposta = telapergunta6(window);
                        if (resposta == 1) {
                                Atual = Atual->esq;
                                resposta = telapergunta12(window);
```

A imagem a seguir é um exemplo de tela de pergunta:



Depois de percorrer a árvore, o main verifica para qual nó o Atual aponta. E, dependendo da info deste nó, chama sua tela de resultado correspondente:

```
if (Atual->info == 3) {
        telaresultado2(window, fcompnuvem1, fcompnuvem2);
}
if (Atual->info == 5) {
        telaresultado2(window, fbancodedados2, fbancodedados1);
}
if (Atual->info == 7) {
        telaresultado2(window, fbioinformatica1, fbioinformatica2);
}
if (Atual->info == 9) {
        telaresultado6(window, fia1, fia2, fia3, fia4, fia5, fia6);
}
if (Atual->info == 11) {
        telaresultado1(window, fempreendedorismo);
}
if (Atual->info == 14) {
        telaresultado4(window, fengsoftware1, fengsoftware2, fengsoftware3, fengsoftware4);
}
if (Atual->info == 16) {
        telaresultado2(window, fcompiladores1, fcompiladores2);
}
if (Atual->info == 21) {
        telaresultado4(window, fihc4, fihc2, fihc3, fihc1);
}
```

A tela de resultado é um conjunto de telas, sendo que o jogador deve ir pressionando barra de espaços e percorrendo os professores que trabalham com aquela área, até que o jogo finalize:

CONCLUSÕES

Neste trabalho aprendemos a implementação de Tipos Abstratos de Dados, especificamente árvore binária de busca. Aprimoramos nosso conhecimento em interface gráfica através do SFML e conseguimos a partir disso implementar um jogo no estilo Akinator/Quizz, cujo objetivo é direcionar o jogador a uma área de afinidade dentre as áreas de computação.