

# FCRDB


**"Food Csv Reader And DB"**



I AM YOUR FATHER



# MAIN



```
int main(int argc, char* argv[])
{
    try
    {
        // Pegando dos parametros onde esta o arquivo
        std::string filePath = argc >= 2 ? argv[1] : "filtered_data.csv";

        ArvoreBST bst = ArvoreBST();
        // Objeto da classe que realiza o armazenamento do Csv dentro da BST
        CsvStorage storage = CsvStorage(filePath, &bst);
        // Metodo que armazena o Csv na BST
        storage.OpenFileAndStorage();
        // Chama o menu (metodo estatico da classe BSTViewer)
        BSTViewer::Menu(bst);

        return 0;
    }
    catch (std::exception ex)
    {
        std::cerr << "Exception ao rodar o FCRD: \n" << ex.what();
        exit(1);
    }
}
```

# BST



```
// ArvoreBST
class ArvoreBST
{
private:
    No* raiz;

public:
    ArvoreBST();
    // Setters
    void setRaiz(No* root) { raiz = root; }
    // Getters
    No* getRaiz() const { return raiz; }
    // Methods
    void inserir(Food food);
    void inserirAux(No* no, Food food);
    void emOrdem(No* no) const;
    void preOrdem(No* no) const;
    void posOrdem(No* no) const;
    No* Pesquisar(std::string dado, No* no) const;
    No* PesquisarRec(No* r, std::string k) const;
    int contarNos(No* atual) const;
    int altura(No* atual) const;
    No* excluir(No* t, std::string key);
    int folhas(No* atual) const;
    std::string min() const;
    std::string max() const;
    No* findMin(No* t) const;
    No* findMax(No* t) const;
    void infs(No* r);
    void printDadosArvore();
    void printDadosArvore(No* no);
    void print() const;
    void print(No* no, int space) const;
    void gerarInformacoesNutricionais(std::list<std::string> meal);
    int totalDeCaloriasConsumidas(std::list<std::string> meal);

    // Friend operator para o overload do operador bitwise
    friend ostream &operator<<(ostream &os, const ArvoreBST &bst);
};
```

```
// No
class No
{
private:
    No* esq, * dir;
    std::string chave;
    Food dado;

public:
    No(Food food);
    // funcoes getters e setters
    std::string getChave() const { return chave; }
    No* getEsq() const { return esq; }
    No* getDir() const { return dir; }
    Food getDado() const { return dado; }
    void setEsq(No* no) { esq = no; }
    void setDir(No* no) { dir = no; }
    void setChave(std::string k) { chave = k; }
    void setDado(Food& dado) { this->dado = dado; }
};
```

# Food.h

```
class Food
{
public:
    // Constructors
    Food(std::string foodAndServing, int calories, int caloriesFromFat, double totalFatG, int totalFatDv,
         int sodiumG, int sodiumDv, int potassiumG, int potassiumDv, int totalCarboHydrateG, int totalCarboHydrateDv, int dietaryFiberG,
         int dietaryFiberDv, int sugarsG, int proteinG, int vitaminADv, int vitaminCDv, int calciumDv, int eeironeeDv,
         double saturatedFatDv, int saturatedFatMgE, int choleSterolDv, int choleSterolMgE,
         std::string foodType);
    Food() {};

    // Getters
    std::string getFoodAndServing() const { return _foodAndServing; }
    int getCalories() const { return _calories; }
    int getCaloriesFromFat() const { return _caloriesFromFat; }
    double getTotalFatG() const { return _totalFatG; }
    int getTotalFatDv() const { return _totalFatDv; }
    int getSodiumG() const { return _sodiumG; }
    int getSodiumDv() const { return _sodiumDv; }
    int getPotassiumG() const { return _potassiumG; }
    int getPotassiumDv() const { return _potassiumDv; }
    int getTotalCarboHydrateG() const { return _totalCarboHydrateG; }
    int getTotalCarboHydrateDv() const { return _totalCarboHydrateDv; }
    int getDietaryFiberG() const { return _dietaryFiberG; }
    int getDietaryFiberDv() const { return _dietaryFiberDv; }
    int getSugarsG() const { return _sugarsG; }
    int getProteinG() const { return _proteinG; }
    int getVitaminADv() const { return _vitaminADv; }
    int getVitaminCDv() const { return _vitaminCDv; }
    int getCalciumDv() const { return _calciumDv; }
    int getEeironeeDv() const { return _eeironeeDv; }
    double getSaturatedFatDv() const { return _saturatedFatDv; }
    int getSaturatedFatMgE() const { return _saturatedFatMgE; }
    int getCholeSterolDv() const { return _choleSterolDv; }
    int getCholeSterolMgE() const { return _choleSterolMgE; }
    std::string getFoodType() const { return _foodType; }

    // Setters: não há necessidade, uma vez que definimos como privado, não queremos que qualquer trecho altere um atributo a toa:
    // Uma vez que os dados são consistentes e lidos pelo csv, poder altera-los seria um tiro no pé.

    // Friend keyword (para poder disponibilizar os atributos privados para o operador bitwise)
    friend std::ostream &operator<<(std::ostream &os, const Food &food);
};
```

```
private:
    std::string _foodAndServing;
    int _calories;
    int _caloriesFromFat;
    double _totalFatG;
    int _totalFatDv;
    int _sodiumG;
    int _sodiumDv;
    int _potassiumG;
    int _potassiumDv;
    int _totalCarboHydrateG;
    int _totalCarboHydrateDv;
    int _dietaryFiberG;
    int _dietaryFiberDv;
    int _sugarsG;
    int _proteinG;
    int _vitaminADv;
    int _vitaminCDv;
    int _calciumDv;
    int _eeironeeDv;
    double _saturatedFatDv;
    int _saturatedFatMgE;
    int _choleSterolDv;
    int _choleSterolMgE;
    std::string _foodType;
};
```



# Food.cpp (Implementação)

```
#include "Food.h"
// Constructor
Food::Food(std::string foodAndServing, int calories, int caloriesFromFat, double totalFatG, int totalFatDv,
int sodiumG, int sodiumDv, int potassiumG, int potassiumDv, int totalCarboHydrateG, int totalCarboHydrateDv,
int dietaryFiberG, int dietaryFiberDv, int sugarsG, int proteinG, int vitaminADv, int vitaminCDv,
int calciumDv, int eeironeeDv, double saturatedFatDv, int saturatedFatMgE, int choleSterolDv,
int choleSterolMgE, std::string foodType) : _foodAndServing(foodAndServing), _calories(calories),
_caloriesFromFat(caloriesFromFat), _totalFatG(totalFatG),
_totalFatDv(totalFatDv), _sodiumG(sodiumG),
_sodiumDv(sodiumDv), _potassiumG(potassiumG),
_potassiumDv(potassiumDv), _totalCarboHydrateG(totalCarboHydrateG),
_totalCarboHydrateDv(totalCarboHydrateDv),
_dietaryFiberG(dietaryFiberG),
_dietaryFiberDv(dietaryFiberDv), _sugarsG(sugarsG),
_proteinG(proteinG), _vitaminADv(vitaminADv),
_vitaminCDv(vitaminCDv), _calciumDv(calciumDv),
_eeironeeDv(eeironeeDv), _saturatedFatDv(saturatedFatDv),
_saturatedFatMgE(saturatedFatMgE),
_choleSterolDv(choleSterolDv),
_choleSterolMgE(choleSterolMgE), _foodType(foodType)
{
}
```


# Food.cpp: Bitwise shift left (<<) overloading

```
// Ostream operator
// Aqui imprimimos a tabela nutricional ao chamar o cout em um objeto da classe
std::ostream &operator<<(std::ostream &os, const Food &food)
{
    std::string card = "|" + food._foodAndServing + "|";
    os << std::setfill('_') << std::setw(food._foodAndServing.size() + 2) << "" << std::endl;
    os << std::left << std::setfill('_') << std::setw(20) << card << std::setw(67) << "" << std::endl;
    os << std::setfill(' ');
    os.flush();
    os << std::left << std::setw(30) << "Calories" << "|" << std::setw(55) << food._calories << "|" << std::endl
    << std::left << std::setw(30) << "CaloriesFromFat" << "|" << std::setw(55) << food._caloriesFromFat << "|" << std::endl
    << std::left << std::setw(30) << "TotalFatG (g)" << "|" << std::setw(55) << food._totalFatG << "|" << std::endl
    << std::left << std::setw(30) << "TotalFatDv (%)" << "|" << std::setw(55) << food._totalFatDv << "|" << std::endl
    << std::left << std::setw(30) << "SodiumG (g)" << "|" << std::setw(55) << food._sodiumG << "|" << std::endl
    << std::left << std::setw(30) << "SodiumDv (%)" << "|" << std::setw(55) << food._sodiumDv << "|" << std::endl
    << std::left << std::setw(30) << "PotassiumG (g)" << "|" << std::setw(55) << food._potassiumG << "|" << std::endl
    << std::left << std::setw(30) << "PotassiumDv (%)" << "|" << std::setw(55) << food._potassiumDv << "|" << std::endl
    << std::left << std::setw(30) << "TotalCarboHydrateG (g)" << "|" << std::setw(55) << food._totalCarboHydrateG << "|" << std::endl
    << std::left << std::setw(30) << "TotalCarboHydrateDv (%)" << "|" << std::setw(55) << food._totalCarboHydrateDv << "|" << std::endl
    << std::left << std::setw(30) << "DietaryFiberG (g)" << "|" << std::setw(55) << food._dietaryFiberG << "|" << std::endl
    << std::left << std::setw(30) << "DietaryFiberDv (%)" << "|" << std::setw(55) << food._dietaryFiberDv << "|" << std::endl
    << std::left << std::setw(30) << "SugarsG (g)" << "|" << std::setw(55) << food._sugarsG << "|" << std::endl
    << std::left << std::setw(30) << "ProteinG (g)" << "|" << std::setw(55) << food._proteinG << "|" << std::endl
    << std::left << std::setw(30) << "VitaminADv (%)" << "|" << std::setw(55) << food._vitaminADv << "|" << std::endl
    << std::left << std::setw(30) << "VitaminCDv (%)" << "|" << std::setw(55) << food._vitaminCDv << "|" << std::endl
    << std::left << std::setw(30) << "CalciumDv (%)" << "|" << std::setw(55) << food._calciumDv << "|" << std::endl
    << std::left << std::setw(30) << "EeironDv (%)" << "|" << std::setw(55) << food._eeironDv << "|" << std::endl
    << std::left << std::setw(30) << "SaturatedFatDv (%)" << "|" << std::setw(55) << food._saturatedFatDv << "|" << std::endl
    << std::left << std::setw(30) << "SaturatedFatMgE (mg)" << "|" << std::setw(55) << food._saturatedFatMgE << "|" << std::endl
    << std::left << std::setw(30) << "CholeSterolDv (%)" << "|" << std::setw(55) << food._choleSterolDv << "|" << std::endl
    << std::left << std::setw(30) << "CholeSterolMgE (mg)" << "|" << std::setw(55) << food._choleSterolMgE << "|" << std::endl
    << std::left << std::setw(30) << "FoodType" << "|" << std::setw(55) << food._foodType << "|" << std::endl;
    os << "|";

    return os;
}
```

SÓ É POSSÍVEL POR CONTA DA KEYWORD "FRIEND"

# MAIN



```
int main(int argc, char* argv[])
{
    try
    {
        // Pegando dos parametros onde esta o arquivo
        std::string filePath = argc >= 2 ? argv[1] : "filtered_data.csv";

        ArvoreBST bst = ArvoreBST();
        // Objeto da classe que realiza o armazenamento do Csv dentro da BST
        CsvStorage storage = CsvStorage(filePath, &bst);
        // Metodo que armazena o Csv na BST
        storage.OpenFileAndStorage();
        // Chama o menu (metodo estatico da classe BSTViewer)
        BSTViewer::Menu(bst);

        return 0;
    }
    catch (std::exception ex)
    {
        std::cerr << "Exception ao rodar o FCRD: \n" << ex.what();
        exit(1);
    }
}
```



# CsvStorage.h

```
class CsvStorage
{
private:
    // Atributos
    // Caminho do arquivo
    std::string _filePath;
    // Ponteiro para a arvore que sera populada
    ArvoreBST* _bst;
    // Quantidade de headers do CSV
    unsigned int _headerQuantity;
    // Quantidades de atributos de cada comida
    static const unsigned int _atributesQuantity = 24;
    // Delimitador do CSV
    char _delimiter;


    // Metodos Privados (explicacao de cada um em suas implementacoes)
    void StorageStreamIntoBST(std::stringstream& stream);
    Food ConvertAtributesIntoFood(std::string* atributes);

public:
    // Metodos Publicos
    void OpenFileAndStorage();

    // Construtor
    CsvStorage(std::string filePath, ArvoreBST* bst);
};
```



# MAIN



```
int main(int argc, char* argv[])
{
    try
    {
        // Pegando dos parametros onde esta o arquivo
        std::string filePath = argc >= 2 ? argv[1] : "filtered_data.csv";

        ArvoreBST bst = ArvoreBST();
        // Objeto da classe que realiza o armazenamento do Csv dentro da BST
        CsvStorage storage = CsvStorage(filePath, &bst);
        // Metodo que armazena o Csv na BST
        storage.OpenFileAndStorage();
        // Chama o menu (metodo estatico da classe BSTViewer)
        BSTViewer::Menu(bst);

        return 0;
    }
    catch (std::exception ex)
    {
        std::cerr << "Exception ao rodar o FCRD: \n" << ex.what();
        exit(1);
    }
}
```

# CsvStorage.cpp

```
/// <summary>
/// Esse metodo abrir o CSV, pular os headers e então criar uma stream com os dados desse arquivo
/// Atraves dessa stream, o método StorageStreamIntoBST() é chamado.
/// </summary>
void CsvStorage::OpenFileAndStorage()
{
    // Aqui estamos criando um "Input File Stream" para podermos ler o arquivo no caminho (configurado no atributo)
    // std::ifstream::in -> Modo de leitura (input)
    std::ifstream inputFile(_filePath, std::ifstream::in);
    if (!inputFile.is_open())
    {
        std::string message = "Erro tentando abrir o arquivo CSV\n Por favor verifique se o caminho do mesmo esta configurado corretamente.";
        throw std::invalid_argument(message);
    }
    // Stream que vai armazenar o conteudo do buffer.
    std::stringstream ss;
    // Leitura do buffer para dentro da stream
    ss << inputFile.rdbuf();

    // Aqui ignoramos os dois headers
    for (int i = 0; i < _headerQuantity; i++)
    {
        // String que vai armazenar o conteudo da linha da stream (temporario)
        std::string content;
        getline(ss, content);
        // Declaracao do delimitador do csv (baseado na separacao do primeiro header)
        if (i == 0)
        {
            _delimiter = content.find(',') != string::npos ? ',' : ';';
        }
    }
    // Chamada do método que vai armazenar o conteúdo da stream na BST
    StorageStreamIntoBST(ss);


    // Após terminarmos o trabalho, precisamos fechar o arquivo.
    if (inputFile.is_open())
    {
        inputFile.close();
    }
}
```

```
// Construtor
CsvStorage::CsvStorage(std::string filePath, ArvoreBST* bst)
: _filePath(filePath), _bst(bst)
{
    _headerQuantity = 2;
    _delimiter = ',';
}
```

# CsvStorage.cpp

```
/// <summary>
/// Esse método recebe uma stream, que através dela realizará o filtro de cada linha armazenada e criará um "Food"
/// para cada linha armazenada (com seus respectivos atributos)
/// </summary>
/// <param name="stream"></param>
void CsvStorage::StorageStreamIntoBST(std::stringstream& stream)
{
    std::string content;
    while (getline(stream, content))
    {
        // Criacao da conteudo da linha
        istringstream line(content);
        // Para cada linha, vamos armazenar os atributos em um buffer (tamanho do buffer estatico e constante definido na propria classe)
        std::string* atributes = new std::string[_atributesQuantity];

        unsigned short attributePosition = 0;
        while (getline(line, content, _delimiter))
        {
            content = (content == "") ? "0" : content;
            atributes[attributePosition] = content;
            attributePosition++;
        }
        // Obtem o objeto atraves dos conversos de atributos
        Food food = ConvertAtributesIntoFood(atributes);
        _bst->inserir(food);
        delete[] atributes;
    }
}
```





# CsvStorage.cpp

```
/// <summary>
/// Atraves de um array, os atributos são armazenados em uma variável, que por sua vez serão utilizadas na criação do objeto
/// </summary>
/// <param name="atributes">: buffer</param>
/// <returns>Food</returns>
Food CsvStorage::ConvertAttributesIntoFood(std::string* atributes)
{
    // Poderíamos ter construído o objeto passando já a posição do array para o construtor:
    // ex: Food food = Food(atributes[0], stoi(atributes[1]), ..., stoi(atributes[N]))
    // Mas com o intuito de promover a organização do código, mesmo com o custo de mais linhas, atribuímos cada valor em uma variável
    // através disso, tornamos o nosso código mais legível
    std::string foodAndServing = atributes[0];
    int calories = stoi(atributes[1]);
    int caloriesFromFat = stoi(atributes[2]);
    double totalFatG = stod(atributes[3]);
    int totalFatDv = stoi(atributes[4]);
    int sodiumG = stoi(atributes[5]);
    int sodiumDv = stoi(atributes[6]);
    int potassiumG = stoi(atributes[7]);
    int potassiumDv = stoi(atributes[8]);
    int totalCarboHydrateG = stoi(atributes[9]);
    int totalCarboHydrateDv = stoi(atributes[10]);
    int dietaryFiberG = stoi(atributes[11]);
    int dietaryFiberDv = stoi(atributes[12]);
    int sugarsG = stoi(atributes[13]);
    int proteinG = stoi(atributes[14]);
    int vitaminADv = stoi(atributes[15]);
    int vitaminCDv = stoi(atributes[16]);
    int calciumDv = stoi(atributes[17]);
    int eeironeeDv = stoi(atributes[18]);
    double saturatedFatDv = stod(atributes[19]);
    int saturatedFatMgE = stoi(atributes[20]);
    int choleSterolDv = stoi(atributes[21]);
    int choleSterolMgE = stoi(atributes[22]);
    std::string foodType = atributes[23];


    Food food = Food(foodAndServing, calories, caloriesFromFat, totalFatG, totalFatDv, sodiumG, sodiumDv, potassiumG, potassiumDv,
        totalCarboHydrateG, totalCarboHydrateDv, dietaryFiberG, dietaryFiberDv, sugarsG, proteinG, vitaminADv, vitaminCDv, calciumDv, eeironeeDv, saturatedFatDv,
        saturatedFatMgE, choleSterolDv, choleSterolMgE, foodType);
    return food;
}
```

# CsvStorage.cpp

```
/// <summary>
/// Esse método recebe uma stream, que através dela realizará o filtro de cada linha armazenada e criará um "Food"
/// para cada linha armazenada (com seus respectivos atributos)
/// </summary>
/// <param name="stream"></param>
void CsvStorage::StorageStreamIntoBST(std::stringstream& stream)
{
    std::string content;
    while (getline(stream, content))
    {
        // Criacao da conteudo da linha
        istringstream line(content);
        // Para cada linha, vamos armazenar os atributos em um buffer (tamanho do buffer estatico e constante definido na propria classe)
        std::string* atributes = new std::string[_atributesQuantity];

        unsigned short attributePosition = 0;
        while (getline(line, content, _delimiter))
        {
            content = (content == "") ? "0" : content;
            atributes[attributePosition] = content;
            attributePosition++;
        }
        // Obtem o objeto atraves do conversos de atributos
        Food food = ConvertAtributesIntoFood(atributes);

        _bst->inserir(food);
        delete[] atributes;
    }
}
```



# MAIN

```
int main(int argc, char* argv[])
{
    try
    {
        // Pegando dos parametros onde esta o arquivo
        std::string filePath = argc >= 2 ? argv[1] : "filtered_data.csv";

        ArvoreBST bst = ArvoreBST();
        // Objeto da classe que realiza o armazenamento do Csv dentro da BST
        CsvStorage storage = CsvStorage(filePath, &bst);
        // Metodo que armazena o Csv na BST
        storage.OpenFileAndStorage();
        // Chama o menu (metodo estatico da classe BSTViewer)
        BSTViewer::Menu(bst);

        return 0;
    }
    catch (std::exception ex)
    {
        std::cerr << "Exception ao rodar o FCRD: \n" << ex.what();
        exit(1);
    }
}
```





# BSTViewer.h

```
#include <cstdlib>
// Macro de compilacao para identificar o sistema na hora de limpar a tela
#ifdef _WIN32
#define CLEAR_SCREEN() system("cls");
#else
#define CLEAR_SCREEN() system("clear");
#endif

#include "ArvoreBinaria.h"
#include <list>
#include <iostream>


class BSTViewer
{
public:
    static void Menu(ArvoreBST& bst);
};
```

# BSTViewer.cpp

```
#include "BSTViewer.h"
/// <summary>
/// Imprime o menu na tela
/// </summary>
/// <param name="bst"></param>
void BSTViewer::Menu(ArvoreBST& bst)
{
    int option = 999;
    std::cout << "Bem vindo ao FCRDB (Food Csv Reader And DB)!" << std::endl;
    do
    {
        std::cout << "\t\t======" << std::endl;
        std::cout << "\t\t< FCRDB Menu >" << std::endl;
        std::cout << "\t\t======" << std::endl;
        std::cout << "1- ~> Consultar refeicao." << std::endl;
        std::cout << "2- ~> Informacoes da Arvore." << std::endl;
        std::cout << "3- ~> Visualizar alimentos armazenados. " << std::endl;
        std::cout << "4- ~> Visualizar tabela nutricional de todos os alimentos armazenados. " << std::endl;
        std::cout << "0- ~> Sair." << std::endl;
        std::cout << "$ ";
        std::cin.clear();
        std::cin >> option;
        switch (option)
        {
```

# BSTViewer.cpp

```
switch (option)
{
case 1:
{
    // Consultar refeicao
    int qty = 0;
    std::cout << "Por favor, informe a quantidade de alimentos da sua refeicao: " << std::endl;
    std::cin >> qty;
    std::cin.ignore();
    std::list<std::string> meal;
    std::string currentFood;
    for (int i = 0; i < qty; i++)
    {
        std::cout << "Informe a refeicao n" << i + 1 << " : " << std::endl ;
        std::cout << "$ ";
        getline(std::cin, currentFood);
        meal.push_back(currentFood);
    }
    // Imprime as informacoes nutricionais da refeicao
    bst.gerarInformacoesNutricionais(meal);
    std::cout << "Total de calorias consumidas: " << bst.totalDeCaloriasConsumidas(meal);
    std::cout << std::endl << "Digite uma tecla para continuar: " << std::endl;
    std::cin.ignore();
    std::cin.get();
    int clear = CLEAR_SCREAM();
    break;
}
```






# BST.cpp

```
void ArvoreBST::gerarInformacoesNutricionais(std::list<std::string> meal)
{
    // Para cada elemento dentro da lista de refeicao (iterator) - https://cplusplus.com/reference/iterator/
    for (list<std::string>::iterator it = meal.begin(); it != meal.end(); it++)
    {
        // Procura o alimento na BST
        No* no = Pesquisar(*it, raiz);
        if (no == NULL)
        {
            // Se nao encontrar, ira para o proximo
            std::cout << "Alimento " << *it << " nao encontrado na arvore!" << std::endl;
            continue;
        }
        Food food = no->getDado();
        std::cout << "Dados do alimento: " << *it << std::endl << food << std::endl;
    }
}
```

# BSTViewer.cpp

```
switch (option)
{
case 1:
{
    // Consultar refeicao
    int qty = 0;
    std::cout << "Por favor, informe a quantidade de alimentos da sua refeicao: " << std::endl;
    std::cin >> qty;
    std::cin.ignore();
    std::list<std::string> meal;
    std::string currentFood;
    for (int i = 0; i < qty; i++)
    {
        std::cout << "Informe a refeicao n" << i + 1 << " : " << std::endl ;
        std::cout << "$ ";
        getline(std::cin, currentFood);
        meal.push_back(currentFood);
    }

    // Imprime as informacoes nutricionais da refeicao
    bst.gerarInformacoesNutricionais(meal);
    std::cout << "Total de calorias consumidas: " << bst.totalDeCaloriasConsumidas(meal);
    std::cout << std::endl << "Digite uma tecla para continuar: " << std::endl;
    std::cin.ignore();
    std::cin.get();
    int clear = CLEAR_SCREAM();
    break;
}
```



# BST.cpp

```
int ArvoreBST::totalDeCaloriasConsumidas(std::list<std::string> meal)
{
    int total = 0;
    // Para cada elemento dentro da lista de refeicao (iterator) - https://cplusplus.com/reference/iterator/
    for (list<std::string>::iterator it = meal.begin(); it != meal.end(); it++)
    {
        // Procura o alimento na BST
        No* no = Pesquisar(*it, raiz);
        if (no == NULL)
        {
            // Se nao encontrar, ira para o proximo
            continue;
        }
        Food food = no->getDado();
        total += food.getCalories();
    }
    return total;
}
```



# BSTViewer.cpp

```
case 2:
{
    // Informacoes da arvore
    std::cout << bst;
    std::cout << std::endl << "Digite uma tecla para continuar: " << std::endl;
    std::cin.ignore();
    std::cin.get();
    int clear = CLEAR_SCREEN();
    break;
}

case 3:
{
    // Imprime chaves em ordem
    bst.emOrdem(bst.getRaiz());
    std::cout << std::endl << "Digite uma tecla para continuar: " << std::endl;
    std::cin.ignore();
    std::cin.get();
    int clear = CLEAR_SCREEN();
    break;
}

case 4:
{
    // Tabelas nutricionais
    bst.printDadosArvore();
    std::cout << std::endl << "Digite uma tecla para continuar: " << std::endl;
    std::cin.ignore();
    std::cin.get();
    int clear = CLEAR_SCREEN();
    break;
}
```

# BST.cpp

```
void ArvoreBST::printDadosArvore()
{
    printDadosArvore(raiz);
}

// Ao chamar esse metodo, cada objeto armazenado na arvore eh impresso
void ArvoreBST::printDadosArvore(No* no)
{
    if (no != NULL)
    {
        printDadosArvore(no->getEsq());
        cout << no->getDado() << std::endl;
        printDadosArvore(no->getDir());
    }
}
```

# EXECUÇÃO



C:\temp\estrutura-de-dados-p1\Estrutura de Dados II\Atividade N1\build\Debug\FCRDB (Food Csv Reader And DB).exe

Bem vindo ao FCRDB (Food Csv Reader And DB)!

=====

< FCRDB Menu >

=====

- 1- ~> Consultar refeicao.
- 2- ~> Informacoes da Arvore.
- 3- ~> Visualizar alimentos armazenados.
- 4- ~> Visualizar tabela nutricional de todos os alimentos armazenados.
- 0- ~> Sair.

\$ \_

<https://drive.google.com/file/d/1O1crBSkd5a3JyCmJBEayYaPXkuTMIu-L/view?usp=sharing>

# EXECUÇÃO

```
C:\temp\estrutura-de-dados-p1\Estrutura de Dados II\Atividade N1\build\Debug\FCRDB (Food Csv Reader And DB).exe
Bem vindo ao FCRDB (Food Csv Reader And DB)!
=====
< FCRDB Menu >
=====
1- ~> Consultar refeicao.
2- ~> Informacoes da Arvore.
3- ~> Visualizar alimentos armazenados.
4- ~> Visualizar tabela nutricional de todos os alimentos armazenados.
0- ~> Sair.
$ 1
Por favor, informe a quantidade de alimentos da sua refeicao:
3
Informe a refeicao n1 :
$ Apple
Informe a refeicao n2 :
$ Grapes
Informe a refeicao n3 :
$ MaisNada
```



Dados do alimento: Apple

Apple	
Calories	130
CaloriesFromFat	0
TotalFatG (g)	0
TotalFatDv (%)	0
SodiumG (g)	0
SodiumDv (%)	0
PotassiumG (g)	260
PotassiumDv (%)	7
TotalCarboHydrateG (g)	34
TotalCarboHydrateDv (%)	11
DietaryFiberG (g)	5
DietaryFiberDv (%)	20
SugarsG (g)	25
ProteinG (g)	1
VitaminADv (%)	2
VitaminCDv (%)	8
CalciumDv (%)	2
FeironeeDv (%)	2
SaturatedFatDv (%)	0
SaturatedFatMgE (mg)	0
CholeSterolDv (%)	0
CholeSterolMgE (mg)	0
FoodType	Fruits ServingESize (gramEweight/ounceEweight)

Dados do alimento: Grapes

Grapes	
Calories	90
CaloriesFromFat	0
TotalFatG (g)	0
TotalFatDv (%)	0
SodiumG (g)	15
SodiumDv (%)	1
PotassiumG (g)	240
PotassiumDv (%)	7
TotalCarboHydrateG (g)	23
TotalCarboHydrateDv (%)	8
DietaryFiberG (g)	1
DietaryFiberDv (%)	4
SugarsG (g)	20
ProteinG (g)	0
VitaminADv (%)	0
VitaminCDv (%)	2
CalciumDv (%)	2
FeironeeDv (%)	0
SaturatedFatDv (%)	0
SaturatedFatMgE (mg)	0
CholeSterolDv (%)	0
CholeSterolMgE (mg)	0
FoodType	Fruits ServingESize (gramEweight/ounceEweight)

Alimento MaisNada nao encontrado na arvore!

Total de calorias consumidas: 220

Digite uma tecla para continuar:

# EXECUÇÃO

```
=====
< FCRDB Menu >
=====
1- ~> Consultar refeicao.
2- ~> Informacoes da Arvore.
3- ~> Visualizar alimentos armazenados.
4- ~> Visualizar tabela nutricional de todos os alimentos armazenados.
0- ~> Sair.
$ 2
```

Informacoes gerais da arvore:

```
Altura da arvore: 22
Quantidade de folhas: 19
Quantidade de Nos: 61
Valor minimo: Apple
Valor maximo: Watermelon
Digite uma tecla para continuar:
```

# EXECUÇÃO

```
C:\Users\Ricar\temp\estrutura-de-dados-p1\Estrutura de Dados II\Atividade N1\src\FCRDB.exe
Bem vindo ao FCRDB (Food Csv Reader And DB)!
=====
< FCRDB Menu >
=====
1- ~> Consultar refeicao.
2- ~> Informacoes da Arvore.
3- ~> Visualizar alimentos armazenados.
4- ~> Visualizar tabela nutricional de todos os alimentos armazenados.
0- ~> Sair.
$ 3
Apple Asparagus Avocado Banana Bell Pepper Blue Crab Broccoli Cantaloupe Carrot Catfish Cauliflower Celery Clams Cod Cuc
umber Flounder/Sole Grapefruit Grapes Green Green Onion GreenCabbage Haddock Halibut HoneydewMelon Iceberg Lettuce Kiwif
ruit Leaf Lettuce Lemon Lime Lobster Mushrooms Nectarine Ocean Perch Onion Orange Orange Roughy Oysters Peach Pear Pinea
pple Plums Pollock Potato Radishes Rainbow Trout Rockfish Salmon Atlantic/Coho/Sockeye /Chinook Salmon Chum/Pink Scallop
s Shrimp Strawberries SummerSquash Sweet Corn Sweet Potato SweetCherries Swordfish Tangerine Tilapia Tomato) Tuna Waterm
elon
Digite uma tecla para continuar:
-
```

# EXECUÇÃO

[Tomato]	
Calories	25
CaloriesFromFat	0
TotalFatG (g)	0
TotalFatDV (%)	0
Sodium (g)	20
SodiumDV (%)	1
Potassium (g)	340
PotassiumDV (%)	10
TotalCarbohydrateG (g)	5
TotalCarbohydrateDV (%)	2
DietaryFiberG (g)	1
DietaryFiberDV (%)	4
SugarG (g)	3
ProteinG (g)	1
VitaminADiv (%)	20
VitaminCDiv (%)	40
CalciumDV (%)	2
Feironediv (%)	4
SaturatedFatDV (%)	0
SaturatedFatMg (mg)	0
CholesterolDV (%)	0
CholesterolMg (mg)	0
FoodType	Vegetables - Serving Size (gram weight/Eounce weight)

[Tuna]	
Calories	130
CaloriesFromFat	15
TotalFatG (g)	1.5
TotalFatDV (%)	2
Sodium (g)	40
SodiumDV (%)	2
Potassium (g)	480
PotassiumDV (%)	14
TotalCarbohydrateG (g)	0
TotalCarbohydrateDV (%)	0
DietaryFiberG (g)	0
DietaryFiberDV (%)	0
SugarG (g)	0
ProteinG (g)	26
VitaminADiv (%)	2
VitaminCDiv (%)	2
CalciumDV (%)	2
Feironediv (%)	4
SaturatedFatDV (%)	0
SaturatedFatMg (mg)	0
CholesterolDV (%)	50
CholesterolMg (mg)	17
FoodType	Seafood - Serving Size (84 g/3 oz)

[Watermelon]	
Calories	80
CaloriesFromFat	0
TotalFatG (g)	0
TotalFatDV (%)	0
Sodium (g)	0
SodiumDV (%)	0
Potassium (g)	270
PotassiumDV (%)	10
TotalCarbohydrateG (g)	21
TotalCarbohydrateDV (%)	17
DietaryFiberG (g)	1
DietaryFiberDV (%)	4
SugarG (g)	20
ProteinG (g)	1
VitaminADiv (%)	30
VitaminCDiv (%)	25
CalciumDV (%)	2
Feironediv (%)	4
SaturatedFatDV (%)	0
SaturatedFatMg (mg)	0
CholesterolDV (%)	0
CholesterolMg (mg)	0
FoodType	Fruits ServingSize (gramWeight/ounceWeight)

[Swordfish]	
Calories	120
CaloriesFromFat	50
TotalFatG (g)	6
TotalFatDV (%)	5
Sodium (g)	100
SodiumDV (%)	4
Potassium (g)	110
PotassiumDV (%)	9
TotalCarbohydrateG (g)	0
TotalCarbohydrateDV (%)	0
DietaryFiberG (g)	0
DietaryFiberDV (%)	0
SugarG (g)	0
ProteinG (g)	16
VitaminADiv (%)	2
VitaminCDiv (%)	2
CalciumDV (%)	0
Feironediv (%)	6
SaturatedFatDV (%)	1.5
SaturatedFatMg (mg)	8
CholesterolDV (%)	40
CholesterolMg (mg)	13
FoodType	Seafood - Serving Size (84 g/3 oz)

[Tangerine]	
Calories	50
CaloriesFromFat	0
TotalFatG (g)	0
TotalFatDV (%)	0
Sodium (g)	0
SodiumDV (%)	0
Potassium (g)	160
PotassiumDV (%)	5
TotalCarbohydrateG (g)	13
TotalCarbohydrateDV (%)	4
DietaryFiberG (g)	2
DietaryFiberDV (%)	8
SugarG (g)	9
ProteinG (g)	1
VitaminADiv (%)	5
VitaminCDiv (%)	45
CalciumDV (%)	4
Feironediv (%)	0
SaturatedFatDV (%)	0
SaturatedFatMg (mg)	0
CholesterolDV (%)	0
CholesterolMg (mg)	0
FoodType	Fruits ServingSize (gramWeight/ounceWeight)

[Tilapia]	
Calories	110
CaloriesFromFat	30
TotalFatG (g)	2.5
TotalFatDV (%)	4
Sodium (g)	30
SodiumDV (%)	1
Potassium (g)	100
PotassiumDV (%)	10
TotalCarbohydrateG (g)	0
TotalCarbohydrateDV (%)	0
DietaryFiberG (g)	0
DietaryFiberDV (%)	0
SugarG (g)	0
ProteinG (g)	22
VitaminADiv (%)	0
VitaminCDiv (%)	2
CalciumDV (%)	0
Feironediv (%)	12
SaturatedFatDV (%)	1
SaturatedFatMg (mg)	5
CholesterolDV (%)	25
CholesterolMg (mg)	25
FoodType	Seafood - Serving Size (84 g/3 oz)

[Green]	
Calories	20
CaloriesFromFat	0
TotalFatG (g)	0
TotalFatDV (%)	0
Sodium (g)	0
SodiumDV (%)	0
Potassium (g)	200
PotassiumDV (%)	6
TotalCarbohydrateG (g)	5
TotalCarbohydrateDV (%)	2
DietaryFiberG (g)	3
DietaryFiberDV (%)	12
SugarG (g)	2
ProteinG (g)	1
VitaminADiv (%)	4
VitaminCDiv (%)	10
CalciumDV (%)	4
Feironediv (%)	2
SaturatedFatDV (%)	0
SaturatedFatMg (mg)	0
CholesterolDV (%)	0
CholesterolMg (mg)	0
FoodType	Vegetables - Serving Size (gram weight/Eounce weight)

[Green Onion]	
Calories	10
CaloriesFromFat	0
TotalFatG (g)	0
TotalFatDV (%)	0
Sodium (g)	10
SodiumDV (%)	0
Potassium (g)	70
PotassiumDV (%)	2
TotalCarbohydrateG (g)	2
TotalCarbohydrateDV (%)	1
DietaryFiberG (g)	11
DietaryFiberDV (%)	4
SugarG (g)	1
ProteinG (g)	0
VitaminADiv (%)	3
VitaminCDiv (%)	8
CalciumDV (%)	2
Feironediv (%)	2
SaturatedFatDV (%)	0
SaturatedFatMg (mg)	0
CholesterolDV (%)	0
CholesterolMg (mg)	0
FoodType	Vegetables - Serving Size (gram weight/Eounce weight)

[GreenCabbage]	
Calories	25
CaloriesFromFat	0
TotalFatG (g)	0
TotalFatDV (%)	0
Sodium (g)	20
SodiumDV (%)	1
Potassium (g)	100
PotassiumDV (%)	5
TotalCarbohydrateG (g)	5
TotalCarbohydrateDV (%)	2
DietaryFiberG (g)	2
DietaryFiberDV (%)	12
SugarG (g)	3
ProteinG (g)	1
VitaminADiv (%)	0
VitaminCDiv (%)	70
CalciumDV (%)	4
Feironediv (%)	12
SaturatedFatDV (%)	0
SaturatedFatMg (mg)	0
CholesterolDV (%)	0
CholesterolMg (mg)	0
FoodType	Vegetables - Serving Size (gram weight/Eounce weight)





I AM YOUR FATHER

