Explicando o que caralhos é isso que nem eu mesmo entendo parte 1:

pq eu to fazendo isso?

R1: Os arquivo tao mt confuso e ngm vai entender essa joça (ta mt bagunçado)

R2: Eu vivo me perdendo no Header.cpp, entao isso vai me dar uma noção de como o arquivo ta organizado, e provavelmente vai me dar ideias de como melhora-lo

Sumario: (dividido em arquivos)

1.0 - Main.cpp

1.1 O inicio do arquivo

```cpp
#include <iostream>
#include <fstream>
#include <algorithm>
#include <vector>
#include "Headers/Header.h"


std::vector<std::string> ops{"=","+","-","*","/"};

                                //IF       THEN      ENDIF      INT        print   STRING  (           )          SOMA
std::vector<std::string> edWords{"Jogue","ValeTudo","ValeNada","Melancia","Ed:","Lenha","ValeNada","ValeTudo","Chico+",
                                 "Chico-","Chico*","Chico/"};
```

Ok, no inicio da pra ver um monte de include, eu vou falar o que cada um deles faz ao decorrer do arquivo, mas ja da pra perceber um bem importante nesse inicio.

1. vector
   - Com vector basicamente da pra criar vetores de qualquer tipo, que é o que eu uso pra armazenar todas as palavras de um arquivo.
   - Não uso uma lista de string pq o vector traz umas coisas interessantes que facilitam na hora de tirar um elemento dele.
2. A lista "ops"
   - é inutil, tenho que apagar
3. A lista "EdWords"
   - São palavras que são aceitas pela linguagem, se for identificado alguma delas, o programa vai chamar a função dentro de Header.cpp

1.2 O inicio do main()

```
int main(){

  std::string arqName;
  std::cout<<"Nome do arquivo: ";
  std::cin>> arqName;
  std::ifstream arq(arqName);

  std::vector<std::string> pals;

  if(arq){
    std::string txt;
    while(arq >> txt){
      txt.erase(remove(txt.begin(), txt.end(), ' '),txt.end());
      pals.push_back(txt);

    }
  }
```

ok, não tem muito o que explicar aqui, eu basicamente uso o fstream pra abrir um arquivo que tem um nome fornecido pelo usuario, depois com o if eu tiro todos os espaços que tem e salvo palavra por palavra (nao linha por linha, ou qualquer coisa do tipo), justamente por isso o modo como voce escreve o codigo nao importa, desde que tenha espaço separando as palavras, pois as palavras vao ser salvas individualmente, levando em consideração apenas a ordem com que foram escritas

---

1.3 - Fim do main e abertura das portas do inferno

```
  for(int i = 0; i<pals.size(); i++){
    for(int j = 0;j<edWords.size();j++){
      if(pals[i] == edWords[j]){
        i+=IdFunc(pals,i);
      }
    }

  }



  return 0;
}
```

nessa parte, podemos ver 2 for loops, um que tem como referencia as palavras do arquivo e outras que usam as edWords como referencia

entao, se o programa ve que uma das palavras escritas é uma EdWord, ele chama a função que está no Header.cpp, passando dois argumentos

1. a lista de palavras do arquivo (pals)

2. a posição da palavra que é uma palavra chave (i)

(desse modo, teoricamente toda coisa escrita que não contesse nenhuma palavra especial seria um comentario, ja que o programa iria simplesmente ignorar)

OBS: é justamente nesse caso (ler parenteses acima) que eu deveria implementar uma mensagem de erro caso uma mensagem nao fosse valida

---

2.0 Header.cpp

ok, acabamos de entrar num lindo paraiso de desorganização do cacete

ok, isso necessita de um sumario:

Sumario (yey):

2.1 Uma bela olhada num belo arquivo "Header.h"

ok, esse arquivo nem é tao desorganizado, então la vamos nos

```h
#pragma once
#include <vector>
#include <iostream>

int IdFunc(std::vector<std::string> pals, int pal);

class Funcs{
  public:
    //void if_();
    //
    void print(std::string inp);
    void print(int inp);
    void print(float inp);

    //gets
    std::vector<float> intV_get();
    std::vector<std::string> intN_get();
    std::vector<std::string> strV_get();
    std::vector<std::string> strN_get();
    int intS_get();


    //sets
    void intV_set(float inp);
    void intV_change(float inp, int pos);
    void intN_set(std::string inp);
    void intS_set(float inp);
    void strN_set(std::string inp);
    void strV_set(std::string inp);
    void strV_change(std::string inp, int pos);


  private:
    std::vector<float> intVars;
    std::vector<std::string> intNames;
    std::vector<std::string> strNames;
    std::vector<std::string> strVars;
    int intSize;
};
```

ok, eu n acho que eu precise explicar muita coisa, é o basico de classe em c++ de um jeito bem nhe

mas ok

1. Como valores e nomes sao armazenados
    - sim, eu criei um vetor com os nomes de todas as variaves (floats e strings)
    - sim, eu criei um vetor pra armazenar os valores de todas as variaveis de um jeito que corresponda a seu nome

- sim, eu sei que isso nao é pratico

FAQ (frenquentamente Asked questões que eu faço a mim mesmo): o que caralhos é esse int intSize;?

R: eu nao faço a minima ideia, os egipcios são alienigenas, eles que colocaram isso ai

---

2.2 inicio do header.cpp.

```
#include <iostream>
#include <cctype> //isdigit
#include <vector>
#include <string>
#include "Header.h"

Funcs fun;
int IdFunc(std::vector<std::string> pals, int pal);
```

ok, nao parece tao ruim assim

como de costume, temos nossos includes e chamamos nossa função do outro arquivo com o nome "fun" (abreviação da abreviação de funções)

mas certo, tambem temos essa função "IdFunc", que nao tem nada, por enquanto.

1. ela necessita de 2 argumentos
    1. um vetor do tipo string
    2. uma int chamada pal (sim, mas nao seja uma criança de 2 anos e 23 semanas)
2. essa função foi utilizada no main.cpp :o :O :o
    1. exatamente, é essa a função que utilizamos no main.cpp quando percebemos que uma palavra faz parte das "EdWords"

2. ai a gente passava 2 argumentos pra ela, a lista de palavra e a posição da palavra que tinha nas EdWords.

3. Essa função é uma int, se a gente ver no main.cpp, a gente tem a seguinte linha:

```
i+=IdFunc(pals,pal);
```

isso significa que caso uma palavra seja detectada, a gente avança pra proxima, porem, o que define qual será a proxima palavra é essa função (IdFunc)

exemplo: se eu tiver o seguinte codigo:

```
Melancia Dois 2
Ed: Dois
```

O programa vai ter identificado a palavra melancia e vai ter chamado a função, mas depois q a função for executada, eu n quero que ele passe pela palavra "Dois" e depois "2", eu quero que ele pule diretamente para a proxima palavra chave, que seria o Ed:

isso é simples de se fazer:

- **Como o return funciona:**

  o return retorna quantas palavras o programa vai ter que ignorar, por exemplo, no exemplo

  ```
  Melancia Dois 2
  Ed: Dois
  ```

ao ver a palavra Melancia, o programa chama a função melancia que vai necessitar da palavra dois e do numero 2, assim, o programa nao precisa passar denovo por essas palavras, entao melancia retorna o valor 2, correspondente ao

numero de argumentos que ela necessita. Isso varia com outras funções da lingua.

- ## 2.3 Melancia

  isso será uma analise do que acontece quando o programa identifica a palavra melancia no programa.

  lembrando que melancia requer dois argumentos:

  Melancia melancia

```
33 int IdFunc(std::vector<std::string> pals, int pal){
32
31   bool intJaExiste = false;
30   int posInt;
29
28   if(pals[pal] == "Melancia"){
27     //verifica se todos caracteres sao inteiros
26     int d = 0;
25     for(int i =0; i<pals[pal+2].size();i++){
24
23       if(isdigit(pals[pal+2][i]))
22         d++;
21       else if(pals[pal+2][i] == '.')
20         d++;
19     }
18
17     for(int i = 0; i<fun.intN_get().size(); i ++){
16       if (pals[pal+1] == fun.intN_get()[i]){
15         posInt = i;
14         intJaExiste = true;
13       }
12     }
11
10
 9     if (intJaExiste){
 8       fun.intN_set(pals[pal+1]);
 7       fun.intV_change(stof(pals[pal+2]),posInt);
 6       return 2;
 5     }
 4     else{
 3       if(d == pals[pal+2].size()){
 2         fun.intV_set(stof(pals[pal+2]));
 1         fun.intN_set(pals[pal+1]);
43       }
 1       else
 2         std::cout<<"Valor da string nao é numerico\nPalavra: "<<pals[pal+2]<<"\n";
 3      return 2;
 4     }
 5
```

1. ok, primeiramente ele checa se a palavra que o main identificou é melancia (por isso precisamos da lista e da posição do elemento nela).
2. depois ele procura saber se todos os caracteres que vem depois do nome sao numero. Pra fazer isso, ele passa por todos os caracteres da terceira palavra e verifica se eles sao numeros, se forem, tudo da certo e a gente pode proseeguir (isso ocorre no primeiro for)
3. depois, o programa precisa saber se esse float ja existe ou nao, caso exista, a unica coisa que ele fará é mudar o valor do float ja existente.
    1. caso ja exista (primeiro if): ele procura a posição do float que ja existe (a posição na lista dos nomes sempre será a mesma na lista dos valores) e passa para a função change(), que será vista mais a frente
        1. SIM EU PRECISO TIRAR URGENTEMENTE ESSA LINHA, ELA BASICAMENTE CONTINUA CRIANDO UM FLOAT NOS NOMES E ISSO VAI, DEFINITIVAMENTE, BUGAR QUANDO FORMOS BUSCAR O VALOR DO FLOAT, JA QUE ELE JA ESTA CRIADO E NAO PRECISA SER CRIADO DE NOVO.
        2. ```
           fun.intN_set(pals[pal+1]);
           ```
    2. caso nao exista, ele cria um float e coloca seu respectivo valor
    3. caso nao seja nenhuma dessas coisas, ele retorna uma mensagem de erro.
           ```