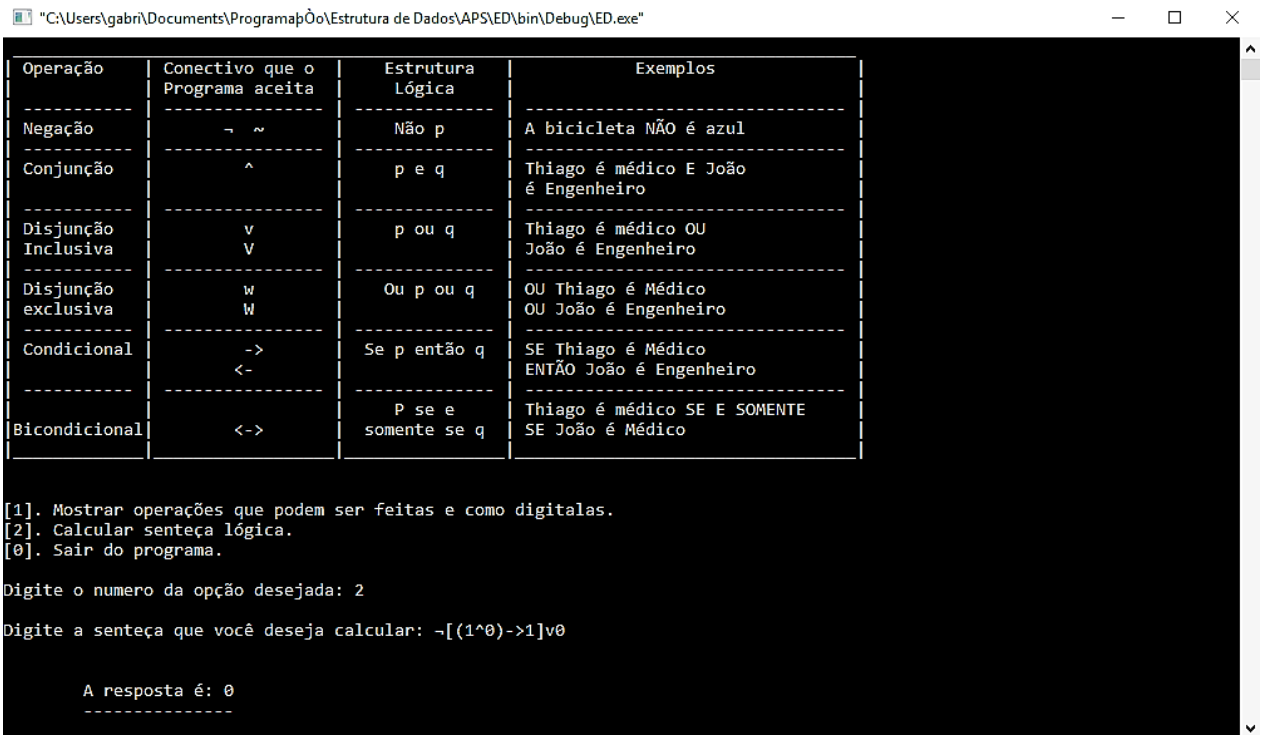
A.P.S de Estrutura de Dados

(Calculadora Lógica)



Nome: Gabriel Alexandre de Souza Braga.

Sumário

1. Introdução

Com o intuito de aplicar os nossos conhecimentos da matéria Estrutura de Dados, o professor da Universidade Tecnológica Federal do Paraná (UTFPR) do campus de toledo, Willian Douglas Ferrari Mendonça, nos pediu para fazer uma Atividade Prática Supervisionada (A.P.S).

A atividade em questão era fazer uma calculadora lógica, esta seria individual, e teríamos um pouco mais de um mês para fazê-la, deveríamos fazer o programa, e junto com o mesmo entregar o relatório.

Uma calculadora lógica, é uma calculadora capaz de efetuar cálculos lógicos, como tabelas verdades, preposições lógicas e etc. Esta seria limitada apenas a sentenças lógicas (Ex: $0 \rightarrow 1$, resposta é 1), onde você insere a sentença e ela te dirá se a resposta é verdadeira ou falsa, em outras palavras 1 ou 0.

O programa deveria ser feito usando a linguagem de programação C, e também usando os conceitos básicos da matéria de Estrutura de dados, conceitos estes como pilha, lista, fila e etc. Usando também boas praticas de programação, também sempre priorizando a otimização do software, assim tendo um bom desempenho em questão de memória e processamento.

2. Objetivo

O objetivo desta A.P.S é demonstrar o conhecimento adquirido pela matéria de Estrutura de Dados, mostrando assim que nós sabemos como aplicá-lo para nos ajudar a resolver problemas de maneira mais rápida, Também demonstrando que somos capazes de desenvolver um programa com os conhecimentos passados à gente, assim confirmando que somos aptos a entender quais são os melhores métodos de estruturação de dados para cada tipo de situação.

Demonstraremos isto por meio da calculadora lógica feita por nós, assim deduzindo que temos este como nosso objetivo final, mas o objetivo inicial é conseguir fazer a calculadora lógica, esta deve ser uma tautologia, funcionando para todos os casos, com isso em mente deveríamos fazer o melhor programa possível e um relatório.

3. Fundamentos teóricos

É importante que vocês tenham noção do que eu estou usando, então com base nisso eu decidi fazer uma breve explicação das coisas mais importantes usadas nesta atividade.

3.1 IDE (Ambiente de Desenvolvimento Integrado)

IDE, como o nome já diz, resumidamente este é um programa que tem ferramentas de suporte para ajudar no desenvolvimento de outros softwares.

A IDE utilizada por mim nesta atividade foi o Code::Blocks este é um programa que tem suporte para a linguagem de programação C, que é a que estamos utilizando para esta atividade, este programa é gratuito, e pode ser baixado neste site http://www.codeblocks.org/.

3.2 Pilhas

para saber oque é uma pilha primeiramente você deve saber oque é uma lista, uma lista é basicamente uma forma de estrutura de dados, em outras palavras, lista é uma forma de interligar elementos de um mesmo conjunto, ou seja, é uma forma de agrupar informações (dados) de elementos que se relacionam entre si de alguma forma.

Pilha é basicamente a mesma coisa que uma lista a única diferença é que esta possui uma regra de inserção e exerção, que deve ser seguida, pois caso contrário ela não será uma pilha, a regra no caso é L.I.F.O que é uma abreviação para as palavras em inglês Last In First Out, que traduzindo ficaria U.E.P.S que seria Ultimo a Entrar Primeiro a Sair, isto quer dizer que o último dado que entrar na pilha deve ser o primeiro a sair da pilha, assim como é na vida real, tentem imaginar que vocês estão colocando um livro em cima do outro, ou seja, empilhando os livros, o último livro a ser posto será oque você terá melhor acesso, e o primeiro que você colocou será o mais difícil de pegar, logo você teria que desempilhar para consegui acesso a ele, a pilha em Estrutura de Dados funciona da mesma maneira.

3.3 Funções ou Rotinas

É importante o seu entendimento nisto, pois assim facilitará a próxima explicação, e também o entendimento do programa em um contexto geral. Rotina também são denominadas de Funções, para ser sincero eu prefiro chamá-las de funções, então a partir de agora toda vez que eu dizer função associem também a rotina.

Uma função nada mais é do que uma sub-rotina de passos usados em um programa, de forma mais clara é um escopo separado do escopo principal (main) com vários comandos, focado em resolver um tipo único de tarefa.

3.4 Biblioteca

Biblioteca é um tipo abstrato de dados, TAD, é basicamente uma forma de dado onde o programador nem sempre tem acesso a todas as funções dela, ou não tem como editá-la, pois nem sempre você precisa saber como funciona, você só precisa que funcione e esteja lá, assim também evita com que o usuário mecha no que não entende e acabe fazendo algo dar errado.

TADs, geralmente são separadas em duas partes, a de definição e a de declaração, mas também elas podem ser feitas no mesmo arquivo apesar de não ser aconselhável, o arquivo de declaração é o que tem extensão .h, o h é de header, e o de definição é o .c.

Então basicamente biblioteca é um aglomerado de várias funções, que tem como finalidade auxiliar o programa em uma tarefa específica. Um exemplo é a math.h que possui várias funções de matemática, assim facilitando sua vida para fazer um programa que usa vários cálculos matemáticos, outro seria o stdio.h que tem funções de out put e in put, para te ajudar a fazer leitura de dados e imprimir dados na tela. Também é importante mencionar que bibliotecas padrão da linguagem C quando são incluídas ao programa usa <nome da biblioteca>, e a criados por nós usam “nome da biblioteca”.

4. Os porquês

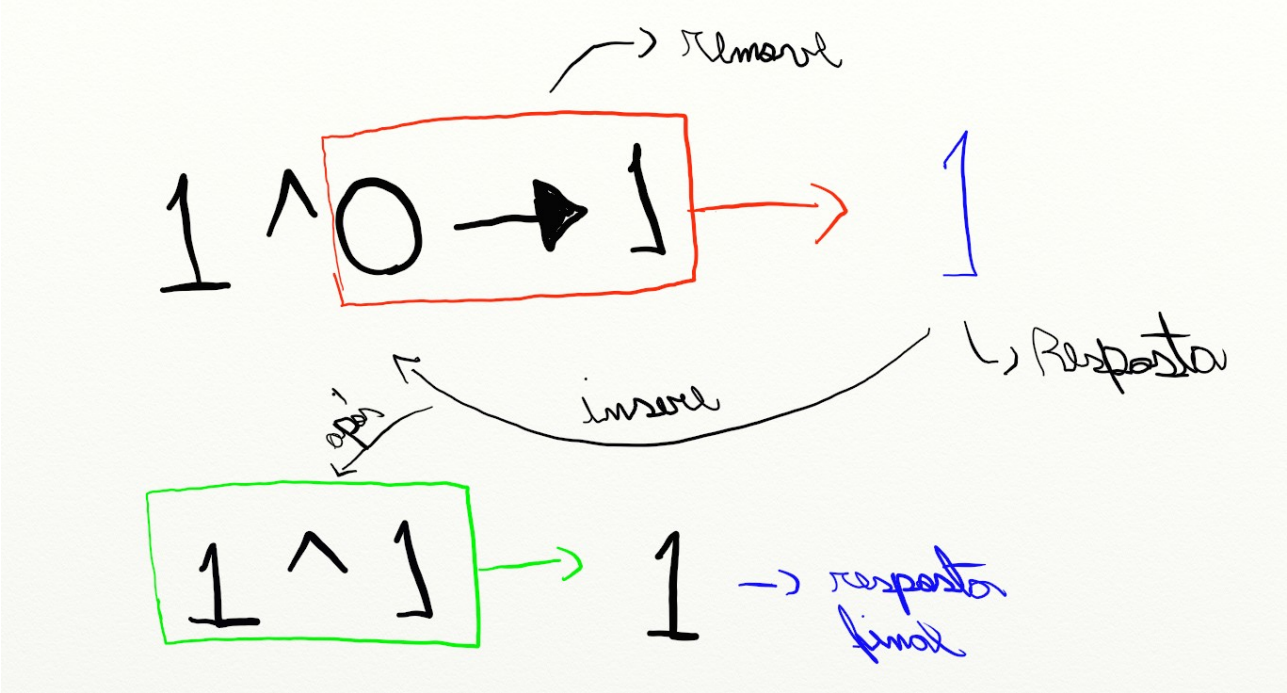
Após o entendimento de alguns conceitos básicos de programação vocês estam prontos para oque vem aqui, neste pedaço eu explicarei os motivos de ter usado as coisas que eu usei, é claro que farei a explicação apenas das coisas mais importantes, então sem mais delongas vamos começar.

4.1 Porquê da pilha

Bom como vocês já sabem existem várias formas de estrutura de dados, existe lista, filas, pilhas e etc. A questão aqui é que algumas dessas formas são melhores que outras em determinados tipos de problemas, por exemplo se você for fazer um programa para um restaurante onde ele chamara a senha para retirar a comida, você quer que o primeiro cliente que chegou seja o primeiro a ser atendido, logo você usar pilha neste programa não compensa, o melhor seria usar fila, pois fila tem uma regra de inserção e exerção que é F.I.F.O (primeiro a entrar primeiro a sair), neste caso o uso de fila ira facilitar sua vida ao fazer o código e também ira ter um melhor desempenho do que as outras opções.

Sabendo disso, você imagina qual a melhor opção para fazer uma calculadora lógica?, a resposta é pilha, pelo menos usando o método que eu usei para resolver o problema pilha foi a que se encaixou melhor. A como já dito anteriormente a regra que a pilha segue é L.I.F.O, e isso facilita as coisas, pois quando o meu programa vai resolver a

sentença lógica ele a divide em partes, e no momento em que ele a divide ele já remove o pedaço que ele utilizara da pilha, o colocando em uma string, em seguida ele resolve esse pedaço e depois insere o resultado na pilha, com o resultado adquirido ele é capaz de resolver o outro pedaço, e assim vai até ele chegar na resposta final. Mas se o programa não usasse pilha ele não seria capaz de fazer tal operação, usando lista como exemplo a hora que eu inserisse o resultado o programa não poderia usá-lo de imediato para resolver o próximo pedaço pois o último a ser inserido é o último a sair, mas em pilha é o ao contrário, o último a ser inserido é o primeiro a sair, logo abaixo há uma imagem para ajudar no entendimento.



4.2 Porquê das bibliotecas

4.2.1 <stdio.h>

Bom o uso da stdio é quase que indispensável, pois esta biblioteca é a responsável pelos comandos de saída e de entrada da linguagem C, comandos estes como printf ou scanf por exemplo, então o motivo de eu ter usado ela está mais do que claro, eu precisava imprimir coisas na tela, e também de comando de entrada para poder receber as variáveis do usuário, então para isso eu tive que usá-la.

4.2.2 <stdlib.h>

Esta biblioteca aqui é a responsável pelo controle de memória, ela é usada para fazer alocações dinâmicas da memória.

Ela foi necessária para fazer a biblioteca da pilha, o somente para isso, mas não pensem que ela é dispensável, pois como o programa foi feito pensando no uso da pilha se não tivesse essa biblioteca ele não funcionaria, pois para criar a pilha e também removê-la ou liberá-la, eu necessito de comandos desta biblioteca.

4.2.3 <locale.h>

Com o intuito de usar os acentos que a língua portuguesa (Brasil) possui nos comandos de saída do programa eu tive que incluir esta biblioteca ela foi usada apenas uma vez e foi usado apenas um comando que é o setlocale, mais para frente eu explicarei melhor oque este comando faz, mas eu acredito que você já devem ter deduzido oque ele faz.

4.2.4 “pilha.h”

Como já avia dito decidi que usaria pilha para resolver os problemas desta atividade, visando os bons métodos de programação decidi que criaria uma biblioteca para me auxiliar e facilitar o controle sobre as pilhas usadas neste programa. Com isso em mente eu criei esta biblioteca pilha.h, que possui funções para o uso da pilha, funções como inserir, remover, liberar, imprimir e etc. Isto facilitou bastante o uso de pilha tornado as coisas mais rápidas e organizadas.

4.2.5 “CalculadoLogica.h”

É aqui onde praticamente toda mágica acontece, sabendo que teria que fazer uma série de funções para resolver as questões da calculadora lógica, decidi que a melhor maneira de fazer isso seria fazendo uma biblioteca para elas. Nesta biblioteca contem apenas três funções que é acessível pelo usuário, estas são CalculaSentenca que fara os cálculos lógicos, InsereSenteca, que auxiliara a fazer e inserção das variáveis, e a função display, que fara a interface do programa, note que estas são funções acessíveis pelo usuário, não quer dizer que a biblioteca contém apenas três funções, ele si ela contém exatamente 14 funções.

Então visando um bom método de programação eu senti a necessidade de fazer uma biblioteca só para os cálculos lógicos, também respeitando os fatos que a função principal (main) tem que ficar o mais “limpa” e organizada possível, e para isso eu não poderia ter 14 funções nela pois ficara uma bagunça.

5. Procedimento de Criação

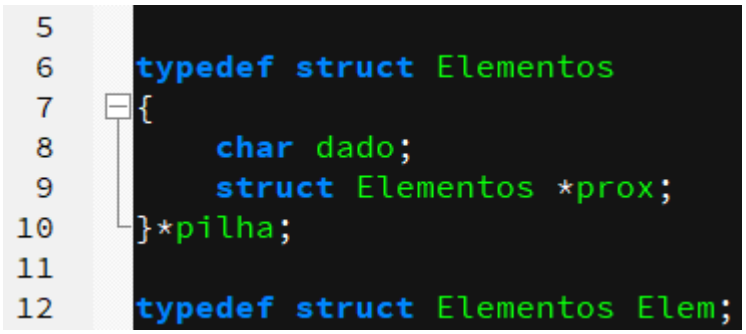
Finalmente chegamos na etapa final desse relatório, aqui falarei sobre cada passo que eu fiz para criar o programa, explicarei eu criei cada pedaço deste software e também o porque de cada coisa, é importante que você tenha lido e entendido oque foi escrito anteriormente, isto para facilitar no entendimento desse pedaço do relatório.

5.1 Criação da biblioteca pilha

Logo após a decisão de usar pilha vi que era necessário cria a biblioteca para ela, a biblioteca é dividida em dois arquivos, o .c e o .h, isto se deve ao fato de que tudo que vai no .h o usuário terá acesso e no .c não, no .c ficará o desenvolvimento de cada função que foi necessária e etc. Esta biblioteca contém 8 funções, mas apenas 6 são usadas de fato no programa, as outras funções foram criadas para me auxiliar na faze de desenvolvimento. Então aqui eu apenas falarei das 5 funções que foram usadas no programa, e também sobre como a pilha foi feita, as 5 funções são InserePilha, CriaPilha, RemovePilha, LiberaPilha e PilhaVazia.

Começaremos falando sobre como foi feito a estrutura da pilha. Ela foi feita no arquivo terminado com .h, pois era necessário que o usuário tivesse acesso a ela, a pilha em si é um ponteiro(*) para o tipo struct Elementos, que dentro dela contém a variável dado que é do tipo char, e prox que é um ponteiro para struct Elementos. Foi feito dois Typedefs, um para apelidar struct Elementos apenas para elem, assim facilitando a minha vida, e outro para apelidar struct Elementos* para pilha, também para agilizar as coisas, assim teria que digitar menos.

A variável dado dentro da struct elementos é do tipo char, isto se deve pelo motivo de que nós estamos trabalhando com strings (cadeias de caracteres), e assim facilitara na hora de armazenar o conteúdo da string na pilha, outra coisa é o prox, ele é um ponteiro para struct Elementos pois é ele que dirá que é o próximo valor da pilha, em outras palavras ele que faz o trabalho de interligar os dados, vocês podem ver na imagem abaixo como ficou a estrutura da pilha.



Para a criação dessa biblioteca foi usado duas bibliotecas, stdio e stdlib, a stdlib ajuda na criação e remoção das pilhas e a stdio a imprimi-la.

5.1.1 Funções da pilha.c

A função CriaPilha é a encarregada de criar a pilha, esta função tem um retorno ponteiro para pilha, dentro dela, eu declaro uma variável chamada pi do tipo ponteiro para pilha, e depois usando malloc eu a crio, após isso faço uma verificação para ver se ela foi criada mesmo de fato, se sim, igualo o conteúdo da pilha a nulo e depois retorno a variável pi.

A próxima função é a InserePilha, ela tem o retorno do tipo short int, pois como em C não dá para trabalhar com uma variável do tipo boleano igual a C# para economizar memória se cria uma short int. Como sabemos que pilha usa a regra LIFO então fazer a inserção é bem fácil, se cria uma variável com o nome nó do tipo ponteiro elem e depois é só igualar o no→dado ao dado inserido nos parâmetros da função e depois dizer que no→prox é igual ao conteúdo que está no topo da pilha, após isso simplesmente falei que o conteúdo do topo da pilha é igual ao nó.

Remoção da pilha é feita pela função RemovePilha, esta tem o retorno igual ao da função anterior isto se dá ao fato que se der algo errado retorna 0 e se a remoção for um sucesso retorna 1, a remoção é simples, você cria um nó, iguala ele ao topo da pilha e em seguida diz que o topo da pilha é igual ao seu próximo elemento, em seguida use o comando free para liberar o nó.

LiberaPilha, é muito simples, você faz a mesma coisa que no paragrafo anterior, mas em um ciclo, um while, no parâmetro do while você diz que enquanto no for diferente de nulo ele não para, após isso você libera a variável pi. Note que o remove libera apenas um elemento da pilha, e a deixa intacta, enquanto o libera, remove todos os elementos da pilha e depois libera a pilha.

A função PilhaVazia, esta é incumbida de checar se a pilha existe, ou se ela está vazia, para fazer isto é muito simples, apenas use um laço de repetição e veja se a negação de pilha for verdade retorne 1 se não então você checa, se a pilha for igual a nulo retorne 1 caso contraio retorne 0.

Por último a função TamanhoPilha, esta é muito simples, é criado um no, que é igualado ao conteúdo da pilha, e seguida utilizando um laço de repetição e uma variável do tipo int, se percorre a pilha até que no seja igual a nulo, e a variável do tipo inteiro ira contando toda vez que a pilha andar, após isso retorne a variável do tipo inteiro.

5.2 Criação da biblioteca CalculadoLogica

Aqui falaremos do próximo passo que eu fiz após a criação da biblioteca pilha, que foi a criação da biblioteca CalculadoLogica, esta é responsável por todo calculo da calculadora lógica.

A primeira coisa que foi feita ao criar esta biblioteca, foi adicionar algumas bibliotecas necessárias no aquivo .h, isto porque o usuário talvez necessite destas bibliotecas, elas foram “pilha.h” e <locale.h>.

A pilha.h foi adicionada pois para fazer os cálculos é necessário o uso de pilhas, e com essa biblioteca isso simplifica o contro sobre as pilhas, o locale.h foi pelo fato de que eu teria que usar os acentos da língua portuguesa (Brasil). Após isto foi criado as três funções que o usuario teria acesso, que são, CalculaSenteca, InsereSentenca e display, explicarei elas no próximo tópico.

5.2.1 Funções da CalculadoLogica.c

Será necessário eu explicar algumas coisas aqui, o aquivo .c contém 14 funções, isto é muita coisa para explicar de uma forma detalhada, então oque farei é explicar as 3 funções do arquivo .h de forma detalhada e dar uma explica meio simples das outras funções, conforme necessário.

5.2.1.1 CalculaSenteca

Começaremos com CalculaSenteca, essa como o nome já diz é a responsável por calcular as sentenças lógicas, o retorno dela é do tipo char, pois a resposta será dada em char, e esse retorno é a resposta, a função pede uma variável do tipo ponteiro para pilha como parâmetro, isto porque na pilha que ira conter os dados necessários para fazer a conta, já de início é declarado algumas variáveis, Svar[4], R, aux[3] todas estas do tipo char, R é a variável que ira receber as repostas dos cálculos conforme o programa roda, aux será usada apenas no caso de negações, ela contem tamanho de 3 caracteres pois este é o tamanho máximo que ela poderá ter, já que terá por exemplo: ¬1\0. Onde o \0 simboliza o final da string, e Svar é para os outros casos, o tamanho máximo que ela poderá ter é 4 caracteres, isto pois o programa divide a sentença em partes pequenas, isto para facilitar os cálculos.

Após isto é criado uma variável i do tipo register int, esta variável i é do tipo register int porque ela será usada com contador nos laços de repetição, e este tipo recebe prioridades e certos direitos na memória que permitem que ele funcione de forma mais rápida, assim tornando o programa mais rápido. Na próxima linha é criado um nó do tipo ponteiro elem, este já é igualado ao conteúdo da pilha, ele servirá para no auxiliar no controle da pilha, em seguida é criado uma variável do tipo inteiro chamada de size, que é igualada a função TamanhoPilha, assim nós saberemos o tamanho da pilha, que ira nos ajudar em algumas coisas.

Após isso é feito alguns laços de repetição, que verificará o tamanho da pilha e de acordo com o tamanho fará uma operação diferente, isto é para verifica já no início se a pessoa digitou apenas 1 ou 0, ou ¬1 ou uma sentença mais complexa como ¬[(1^0)→1]v0 se a pilha for maior ou igual a 3 ela entrará no laço, dentro dele contem um while que só ira para quando no→prox for igual a nulo, em seguida ele entrará em um for, que pegará cada caractere da pilha e inserir na string Svar, isso verificando se não há nenhum parêntese ou coxetes, caso haja ele fará uma chamada recursiva na função, e quando ele achar o parêntese ou coxete que fecha ele fará o retorno. Quando ele terminar de inserir na Svar ele chamará a função EscolheOperacao, esta estará incumbida de escolher qual é a operação correta para usar na Svar, o retorno dela será dado a variável R.

Vamos falar um pouco sobre a função EscolheOperacao antes de prosseguir, dentro dela e declarado uma variável c do tipo char, e é chamado a função EncontraCondicao, após encontrar a condição vira um switch, que de acordo com o conteúdo de c ele chamara uma função, as funções contidas são, e (^), ou (v), ou ou (w), condicional (> ou <), bicondicional (-) e por último negação (¬), tanto ¬, quanto ^ ou v, foi muito simples de ser feito pois estes conectivos lógico já existem em c, a única coisa que eu fiz foi converter o número que estava em caractere para inteiro e depois usei os conectivos existentes para solucionar o problema, agora a condicional, bicondicional e o w eu fiz um if.

Continuando no calcula sentença, após o retorno do EcolheOperacao ser atribuído a R, R é inserido na pilha, e este ciclo começa de novo, e vai até ter apenas uma variável na pilha, que seria a resposta, esta será atribuído a R e R será o retorno da função CalculaSenteca.

Caso pilha seja menor que 3 e igual a 2 é concluído que a operação é uma negação, então o programa insere oque está na pilha na variável aux, e depois chama a função EscolheOperacao, em seguida o retorno da função será dado a R e R será o retorno de CalculaSenteca.

Caso a pilha seja menor que 2, o programa deduz que há um número 1 ou 0, e ele retorna próprio como resposta.

5.2.1.2 InsereSenteca

Esta função será a responsável por verificar se a sentença lógica digitada esta certa e de inseri-la na pilha, ela pede uma string, e um ponteiro para pilha por parâmetro, note que a string está como ponteiro para const char, isto se dá pelo fato de que assim você protege a string, e tem certeza de que a função não alterará o seu conteúdo, esta função tem como retorno short int, pois ela ira retornar um caso a inserção for um sucesso e 0 caso tenha tido algum problema na inserção ou caso a sentença esteja digitada errada.

No início da função é declarado duas variáveis, uma do tipo short int com o nome de boleano, e outra do tipo register int com o nome de i. I será usada nos laços de repetição, e boleano é a que ira receber os uns e zeros que dirá se as operações foram um sucesso ou não, junto com a declaração da variável boleano ela já recebe o valor do VerificaSeteca.

Antes de continuarmos vamos explicar oque esta função faz, esta função tem o objetivo de determinar se a sentença digitada esta certa ou não, ela é dividida em 4 partes, a primeira verifica se todos os caracteres que foram digitados estão dentro dos permitidos pelo programa, a segunda verificará se o final da string esta digitada certa, a terceira verificará se a sintaxe foi digitada corretamente, ou seja, se não foi digitado por exemplo (1v0)←v1, isto esta errado, e a terceira parte que determinará se está errado ou não a sentença digitada, a quarta e última parte verificará se todos os coxetes e parênteses estão abrindo e fechando corretamente, e se não a coxetes e parênteses a mais, se ela passar por tudo isso sem problemas a função retornará 1 caso contrário 0.

Continuando com a função InsereSenteca, após chamar a função VerificaSentenca o retorno dela é atribuído a boleano, e então é chamado dois laços de condição, um vendo se a pilha realmente existe e outro vendo se boleano é igual a 0, caso boleano seja 0 e a negação de pilha for verdade o programa retornara 0 caso contrário 1, após isso o programa entre em um laço de repetição onde ele verificará cada caractere e substituirá pelos padrões do programa e inserirá na pilha, após isso, caso todas as inserções tenhma sido um sucesso, o programa retornará 1.

5.2.1.3 Display

Esta é a última função da biblioteca CalculadoLogica, e o seu retorno é do tipo void, ou seja, não há retorno nesta função, ela não pede nada nos parâmetros, e esta tem o objetivo de fazer a interface da calculadora lógica.

Já no início da função é usado o comando setlocale da biblioteca locale.h, este comando adicionará os acentos da língua portuguesa (Brasil) para o programa, após isso é declarado várias variáveis, do tipo short int há apenas uma, boleano, do tipo ponteiro para pilha, há apenas pi, do tipo char há StrVar[256], R, Controle[256]. StrVar é uma string, e essa receberá a sentença digitada pelo usuário, R receberá a reposta, e controle é mais uma string, esta é a chave de controle, que determinara qual foi a opção escolhida pelo usuário no display, éla é uma string, pois assim caso o usuário digite um caractere o programa não terá nenhum bug, e ficara fácil de identificar e dizer para o usuário digitar uma opção valida.

Quando a variável boleano é criada, é atribuído 0 a ela, isto se dá ao fato de visar boas maneiras de programação, assim sempre é bom iniciar as variáveis, assim fica facil de identificar algum erro e ajuda a não ter bugs, ao declarar pi, já é chamado a função CriaPilha, assim atribuindo a pi a região da memória onde esta a pilha.

Em seguida controle é definido como 1, assim quando entrar no laço de repetição já de início ele ira imprimir a tabela que contém as operações que o programa aceita e como digitá-las, no parâmetro do laço de repetição há a seguinte condição, enquanto controle for diferente de 0 faça oque há dentro do laço, entrando no laço há um switch, que pede controle como parâmetro, no switch há dois cases e um default, o primeiro case tem como condição 1, caso oque for digita seja 1 ele imprimirá a tabela com as operações aceitas e como digitá-las, no segundo case o a condição é 2, caso oque foi digitado seja dois ele assume que você deseja fazer um cálculo lógico, assim ele pedirá que você digite a sentença lógica, oque foi digitado será atribuído a string StrVar, depois ele chamará a função InsereSentenca, o retorno dela será atribuído a boleano, então com um laço de condição o programa verificará se boleano é igual a 0 se sim ele imprimirá entrada invalida, e pedirá para digitar de novo, se não o programa chamará a função CalculaSenteca o retorno desta será atribuído a R, e o programa imprimira a resposta que está em R, no default há apenas um “Por favor digite uma opção válida”, isto caso oque o usuário digitou não seja nem 0, 1 ou 2.

Saindo do switch há uma sequência de printf que imprimirá as opções que o usuário pode fazer, estas são 3, 1 Mostrar operações que podem ser feitas e como digitá-las, 2 Calcular sentença lógica e 0 Sair do programa, o usuário deve digitar o número que representa a opção desejada e este irá ser atribuído a variável controle.

Após isto é chamado a função VerificaControle, o retorno desta função é do tipo short int, pois ela retornará 0 ou 1, 0 se a variável controle é invalida e 1 se for valida, ela pede no parâmetro uma variável do tipo ponteiro para const char, ponteiro pois ela receberá uma string, e const para ter certeza que a função não alterará o conteúdo da variável, após isto é feito alguns laços de condição para verificar se a chave de controle esta digitada corretamente, caso ela contenha mas de 1 caractere e este caractere seja diferente de 1, 2 e 0 a função retornará 0, caso contrário 1.

O retorno da função VerificaControle é atribuído a variável boleano, com um laço de condição eu verifico, se boleano for igual a 0 controle vai ser igual a 3, e então o laço while de se repete, e fica neste ciclo até o usuário digitar 0, quando isto acontecer será chamado a função LiberaPilha, e depois disso a função display acaba.

5.3 Criação da função main

Após tudo isto que foi dito, veio a última parte e a mais simples, implementar a main, no arquivo da main foi adicionado apenas uma biblioteca, esta é a biblioteca CalculadoLogica, após isto visando sempre as melhores maneiras de programação possível, é conhecido que a função main tem que ser o mais simples possível, e ela deve na grande maioria das vezes apenas chamar funções e bibliotecas, sabendo disso eu adicionei apenas uma biblioteca a ela, que já foi dito anteriormente e dentro da função main há apenas uma função sendo chamada que é a display, após isto tem o return 0, que simboliza que o programa acabou.

6. Referencias

Para ser sincero não há referências, há apenas uma referência este é o livro C Completo e Total 3ª Edição, criado pelo Herbert Schildt e publicado pela editorá MAKRON Books, toda e qualquer referência que tenha neste relatório foi tirada deste livro.