Projeto Prático 01 (2019.2)

Anna Caroline de Oliveira Sousa - 2190346; Erika Burei Alves - 2192900.

Universidade Tecnológica Federal do Paraná (UTFPR)

Curitiba – PR – Brasil

annaosousa@outlook.com; erika10011@live.com

Resumo. *O projeto prático 01 desenvolvido, tem o intuito de converter questões da vida cotidiana em problemas de programação, os quais podem ser desenvolvidos com os conteúdos apresentados na disciplina de fundamentos de programação.*

1.Calendário

1.1. Descrição do problema

A necessidade se encontra em construir um programa que executa um calendário de acordo com o ano que o usuário digitar, sendo que este caléndario deve estar de acordo com as características necessárias. Ele deverá: levar em conta quando o ano é bissexto, os meses com diferentes términos (28, 29, 30 ou 31), o dia da semana que se inicia cada mês e assim por diante.

1.2. Técnicas para desenvolvimento do programa

No decorrer do programa foram utilizadas técnicas e funções de diferentes indivíduos. O principal algoritmo utilizado foi baseado na congruência de Zeller, e implementado por Juan Carlos Zuluaga em seu canal do Youtube. As demais etapas do programa estão baseadas em funções encadeadas e desenvolvidas com o objetivo de identificar as características do calendário.

1.3. Principais dificuldades

A dificuldade que mais se destacou foi encontrar um algoritmo que calcula que dia da semana começa cada mês e a sua execução (devido ao algoritmo considerer os meses de janeiro e fevereiro do ano anterior ao solicitado). Outras dificuldades surgiram no decorrer do processo, como erro de posições de chaves, a contagem dos dias da semana (que teria que pular linha após 7 números), entre outros.

2.Programa pronto

2.1. Descrição do problema

O problema apresentado no exercício 2 solicitava o desenvolvimento de um teste de mesa, dado um programa, além disso se fez necessério a compreensão de dois operadores (& e <<), o que muda se for alterado um número em uma constante . Concomitantemente, foi questionado a existência de outra forma de solucionar o problema.

Em tese, este exercício requiria imprescindivelmente o entendimento do programa para responder todas as perguntas propostas.

2.2. Técnicas para desenvolvimento do programa

O programa envolveu a criação de uma constante, a qual tinha o valor alterado por um operador de fluxo e também englobou técnicas de comparação de número binário em sua respectiva posição (operador &), entre outras funções e comandos.

Para o teste de mesa primeiramente foi necessário identificar o padrão e a lógica do programa, para assim conseguir entender suas atribuições e restrições - valores de entrada e saída. Além do mais, a respeito dos operadores & e << se fez necessário pesquisar sobre ambos para assim compreender suas funções. Sobre o número que foi alterado – valor da constante, tivemos que modificar e executá-lo para notar a variação. Em relação a outra forma de solucionar o problema, foi utilizado ferramentas já aprendidas em sala de aula, como por exemplos as estruturas: for, if, else, e encadeamentos de repetições.

2.3. Principais dificuldades

A compreensão do problema e do programa em si foram consideravelmente dificieis, pois o mesmo exigia conceitos novos, após pesquisar na web, livros e perguntar para várias pessoas – as quais notamos possuir conhecimento sobre programação – foi possível inferir o seu funcionamento. Não obstante, o maior desafio foi entender o encadeamentos das estruturas de repetições (quando cada uma deve ser executada), além de compreender o funcionamento da última função.

3.Codificação de mensagens

3.1. Descrição do problema

A proposta do exercício três (3) conta com o auxílio de uma história, a qual é narrada entre dois personagens que possuem como objetivo comunicar-se via mensagem, entretanto, como há uma certa distância entre os mesmos, existem corrompimentos dentro da mensagem. Com base nisso, uma das personagens tenta solucionar este problema, propondo a transmissão dos símbolos usando os valores da tabela ASCII a fim de identificar se a mensagem é corrompida ou não.

Com este apoio, o método utilizado pelos mesmos foi contar a quantidade de um (1) em binário que corresponde ao caractere ou número decimal digitado, caso seja par não é corrompido. Desta forma, se for ímpar, o bit mais à esquerda deverá ser alterado para um (1) e terá que ser impresso esse valor – na hipótese de ter que codificar a mensagem, opção A - ou um asterisco – no contexto em seja necessário descodificar a mensagem, caso B. Além de que, ao final, deverá ser impresso a descodificação com suas alterações ou a codificação com suas modificações, tudo de acordo com o proposto.

3.2. Técnicas para desenvolvimento do programa

Para o desenvolvimento do programa foi utilizado conhecimentos adquiridos na aula - como comentários da professora e listas de exercícios. Além de que, foi comentado na aula sobre o “bug” que ocorre se ao invés de passar o tipo original da variável, for

passado um outro tipo (como por exemplo: char com int ou int com char); já que ocorrerá a conversão correspondente na tabela ASCII, a qual o C foi programado. Este saber foi usado na prática para converter um valor digitado pelo usuário em caractere sendo o oposto disto também válido.

Concomitantemente, outras ténicas foram utilizadas como a divisão do número original por dois - até que este se torne igual a zero - e uma variável acumuladora - para receber o valor correspondente sempre que o resto da divisão não fosse igual a zero - ou seja, o mesmo conta quantos algarismos correlatos ao número um (1) existem e realiza o armazenamento desse valor. Logo após, essa variável passa por uma análise a qual permite identifcar se o bit da sétima (7) posição à esquerda é igual à um (1), caso seja, o número é subtraído 128 unidades - o que não o deixa entrar na condição de corrompido. Na circunstância de que o número seja ímpar o mesmo é deturpado e não entra nas condições somente fica dentro da função, que tem como objetivo verificar a condição do problema.

3.3. Principais dificuldades

Utilizar o fluxo de comando para percorrer o número e averiguar a existência do número um (1) na posição desejada.

4.Cartão de crédito

4.1. Descrição do problema

A proposta do problema consiste em criar um programa, o qual aborda os diversos aspectos necessários para a análise de um cartão de crédito dado informações a serem seguidas.

Com base nisso, a verificação aborda os seguintes aspectos: válidade, invalidade, caractere inválido, tamanho e operadora.

4.2. Técnicas para desenvolvimento do programa

Para o desenvolvimento do programa foi utilizado lógica a fim de criar funções que condicionam a execução do programa da maneira requerida, além de dispor do encadeamentos de funções. Portanto, as funções desenvolvidas auxiliaram a identificação da operadora a que o cartão pertencia – sendo necessário a identificação do prefixo – além de analisar o tamanho e a sua validade através da somatória dos digítos pares multiplicados por dois – e quando maior que nove (9), se fez necessário a separação dos mesmos – com os digítos nas posições ímpares.

4.3. Principais dificuldades

Um dos principais desafios encontrados foi a percepção da necessidade de utilizar o long long – devido estarmos executando uma operação com digítos além da capacitada armazenada pela variável inteira (int), para aumentar a capacidade de digítos armazenados. Não obstante, a compreensão de que era necessário separar os dígitos das posições pares maiores que nove (9). Além destes, outros fatores foram empecilhos para a execução do programa, como por exemplo a manipução dos algoritmos para quando a mensagem envolver caractere, tanto que se essa mensagem não for digitada na primeira

vez pelo usuário mas sim depois de alguma outra tentiva - seja com número grande ou pequeno - a saída será um looping; infelizmente mesmo utilizando o break em outros lugares que poderiam parar esse erro não conseguimos arrumar o programa corretamente, pois apenas alterava mais as outras funções para pior. Outra dificuldade encontrada foi para não aparecer a validade do cartão quando for "operadora desconhecida" porque a condição else sempre era verdadeira.

5.Referências

American Standards Association (1963) "American Standard Code for Information Interchange", https://pt.wikipedia.org/wiki/ASCII.

Moura, Arnaldo e Ferber, Daniel (2009) "Estruturas Condicionais".

Saade, Joel (2003) "Programando em C++" p. 61 e p. 196-203.

Zuluaga, J. C. (2014) "Ejercicio23 (C++) – CALENDARIO DE UN AÑO".