Cristopher 2.0

Autores: Yurik Alexsander, Samuel Gonçalves, Victor Ruinivan e Vitor Hugo Sena Dos Reis.

Resumo

O projeto tem como objetivo principal trabalhar com um conjunto de caracteres para transformá-los de maneira eficiente. Primeiro, queremos separar esses caracteres em pares. Depois, vamos converter esses pares que estão em hexadecimal para decimal. Uma parte importante do processo é fazer com que o código interrompa a execução ao encontrar um par de zeros.

Em seguida, precisamos garantir que a fórmula funcione corretamente, eliminando caracteres indesejáveis. Após isso, o programa deve transformar os números decimais em ASCII e mostrar a frase decifrada por completo. Também queremos que o código consiga decifrar múltiplos códigos, de acordo com a necessidade da equipe de inteligência.

Por fim, a ideia é colocar a fórmula que exclui caracteres indesejáveis em um arquivo separado, mas mantendo um link entre os dois, para facilitar a manutenção e a organização do trabalho.

Introdução

O tema é a manipulação e decodificação de conjuntos de caracteres, com um foco particular na transformação de dados entre diferentes formatos, como hexadecimal, decimal e usar a tabela ASCII. Onde a interpretação correta dos dados é crucial.

A relevância deste estudo busca entender como decifrar e manipular dados com a quantidade crescente de informações sendo recebidas pelo programa, mas também fortalece a capacidade de resposta do programa ao decifrar o código, buscando assim, contribuir para o desenvolvimento de uma ferramenta que possam ser mais úteis no processo de interpretar uma mensagem codificada, aumentando a eficácia das ações nesse campo.

Os objetivos da pesquisa incluem a separação dos caracteres em pares, a conversão de hexadecimal para decimal e a implementação de uma lógica que interrompa o processo ao detectar pares de zeros. Além disso, pretende-se garantir que a fórmula utilizada para a decodificação funcione corretamente, excluindo caracteres indesejáveis. O programa também deve ser capaz de converter os valores decimais em ASCII e exibir claramente a mensagem decifrada, permitindo a

manipulação de múltiplos códigos conforme a necessidade da equipe de inteligência.

Metodologia

1. Separação de Caracteres em Pares

O primeiro passo consistiu em desenvolver um algoritmo em C para separar o conjunto de caracteres em pares. Utilizamos um loop for para iterar sobre a string original, agrupando os caracteres em pares utilizando os próprios parâmetros do for para auxiliar na separação. Essa separação foi armazenada em uma string auxiliar.

2. Conversão de Hexadecimal para Decimal

Em seguida, implementamos uma função para converter cada par de caracteres de hexadecimal para decimal. Utilizamos a função strtol() da biblioteca <stdlib.h>, que permite a conversão direta de strings em diferentes bases.

3. Interrupção ao Encontrar Pares de Zeros

Para garantir que o código interrompesse a execução ao detectar um par de zeros, implementamos uma verificação dentro do loop que processava os pares. Ao encontrar "00", utilizamos a instrução break para sair do loop imediatamente.

4. Exclusão de Caracteres Indesejáveis

A exclusão dos caracteres não depende do código em si, apenas da posição do par de caracteres separados e do código de inteligência fornecido pelo agente da inteligência, que se tornam parâmetros para a fórmula fornecida que realiza o cálculo e os exclui.

5. Conversão de Decimal para ASCII

A conversão é feita quase de forma autônoma, apenas colocamos a variável inteira que guarda o valor do código ascii e forçamos ela ser representada como caracter.

6. Apresentação da Mensagem Decifrada

Para exibir a mensagem decifrada, utilizamos printf para formatar a saída de maneira clara e intuitiva, apresentando a mensagem final ao usuário, apresentando de carácter em carácter por meio de um loop, porém sendo mostrado tudo em conjunto na prompt.

7. Teste de Múltiplos Códigos

Realizamos testes extensivos com diversos conjuntos de dados. Cada parte do código foi revisada e ajustada conforme necessário, garantindo a precisão das saídas. A robustez do algoritmo foi avaliada com casos de teste variados fornecidos.

Considerações Finais

Esses métodos e procedimentos em C formaram a base do trabalho, permitindo uma abordagem eficaz para a manipulação e decodificação de dados. A estrutura do código e a escolha dos algoritmos foram fundamentais para garantir a eficiência e a manutenção futura do programa.

Resultados

Foram interceptadas diversas mensagens em formato hexadecimal durante as operações de inteligência. Cada mensagem continha um valor b, que influenciava a decodificação. As mensagens foram processadas com base na função matemática fornecida, que utilizava coeficientes específicos para determinar a validade de cada caractere.

Discussões

Realizando o trabalho encontramos algumas inconsistências, como a necessidade de estendermos a tabela ascii, e a necessidade de arredondarmos (com round) o retorno que a função presente no arquivo que contém a função de exclusão de caracteres. Pois o cálculo resulta em um número do tipo float e a função específica nas instruções do trabalho que o resultado deve ser obrigatoriamente em número inteiro.

Caso 1:

Entrada:

(Quantos códigos:)1
(Informe o codigo da inteligencia desse código:)0
(Informe o código:)566F6388732073C66F2076656
E6365646F867265732C20766F
63C3887320636FBE6E7365677
5656D2E002DC6C921B7B87FCF

Saída:

Vocês são vencedores, vocês conseguem.

Análise: A mensagem foi totalmente decodificada com sucesso, mostrando a eficácia do algoritmo em interpretar caracteres válidos.

Caso 2:

Entrada:

(Quantos códigos:) 1
(informe o código da inteligência desse código:)3
(Informe o código:)5465636E6F336C6F676961206
46120496E666F726D6187C66F
2E00333333333333333333333
33333333333333333333333333

Saída:

Tecnologia da Informação.

Análise: A decodificação foi bem-sucedida, evidenciando a capacidade do sistema em extrair informações relevantes.

Caso 3:

Entrada:

(Quantos códigos:) 2
(Informe o codigo da inteligencia desse código:) 0
(Informe o código:)566F6388732073C66F2076656E
6365646F867265732C00566F
6388732073C66F2076656E636 5
646F867265732C00332C2C2C
(Informe o codigo da inteligencia desse código:) 3
(Informe o código:) 566F638873C320636F6E73656
775656D2E002DC6C921B7B87F
CF566F638873C320636F6E736
56775656D2E002DC6C921B7B8

Saída:

(“Primeira saída: ”)Vocês são vencedores,
(“Segunda saída: ”)Vocês conseguem.

Análise: Ambas as mensagens foram decodificadas com sucesso, demonstrando a eficácia do sistema em lidar com múltiplas entradas e extraindo informações válidas.

Conclusão

A pesquisa demonstrou a eficácia da manipulação e decodificação de conjuntos de caracteres utilizando a linguagem C. Os métodos desenvolvidos permitiram a separação adequada dos caracteres em pares, a conversão entre diferentes formatos numéricos e a apresentação clara da mensagem decifrada. A implementação de mecanismos de interrupção e filtragem garantiu que o

processamento fosse seguro e preciso, aumentando a robustez do código. Os testes realizados validaram a funcionalidade do algoritmo, confirmando sua capacidade de lidar com múltiplos conjuntos de dados de forma eficiente.

Resumo dos Principais Achados

1. **Separação Eficiente**: O algoritmo foi capaz de separar corretamente caracteres em pares, mesmo quando o número total de caracteres era ímpar.
2. **Conversão Precisa**: A utilização de funções como strtol() para conversão de hexadecimal para decimal demonstrou ser eficaz, permitindo a transformação correta dos dados.
3. **Controle de Fluxo**: A lógica de interrupção ao encontrar pares de zeros funcionou adequadamente, evitando processamento desnecessário.
4. **Filtragem de Dados**: A implementação de uma função para excluir caracteres indesejáveis garantiu que a saída final fosse limpa e relevante.
5. **Transformação para ASCII**: A conversão de valores decimais para ASCII foi realizada com sucesso, permitindo a apresentação clara da mensagem decifrada.
6. **Robustez Testada**: Os testes com múltiplos códigos confirmaram a eficácia do programa, ressaltando sua capacidade de adaptar-se a diferentes conjuntos de dados.

Referências

https://ava.catolica.edu.br/d2l/le/enhancedSequenceViewer/90086?url=https%3A%2F%2F211c9f77-18c9-42e7-a7d8-b67813cc574d.sequences.api.brightspace.com%2F90086%2Factivity%2F1552204%3FfilterOnDatesAndDepth%3D1 (E os conteúdos aprendidos em aula).

Apêndice: https://github.com/ruinivan/c/tree/main/works/work_n1