Universidade do Minho
Escola de Engenharia

Mestrado Integrado em Engenharia de Telecomunicações e Informática

MÉTODOS DE PROGRAMAÇÃO I

PROJETO ID CODES
(VERSÃO FINAL)

Beatriz da Ressurreição
Alves - A96003
a96003@alunos.uminho.pt

28 de agosto de 2022

Índice

Índice de Figuras

Índice de Tabelas

Lista de excertos de código

1 Introdução

No âmbito da unidade curricular de Métodos de Programação I, foi pedido a solução de um problema algorítmico, utilizando como ferramentas o que foi leccionado na Unidade Curricular.

Para a resolução deste problema, foi necessário compreender cada passo que o programa teria de fazer recorrendo a um algoritmo bem construído.

Neste fase o relatório contem um algoritmo não refinado, um fluxograma com exemplos e o programa em linguagem C.

Já que não é especificado qual deveria ser o comportamento do programa caso fossem introduzidos dados inválidos pelo utilizador (tais como sinais de pontuação ou caracteres especiais no meio de uma string), consideramos que o programa nesses casos corra normalmente.

1.1. Ferramentas utilizadas

As ferramentas que utilizamos para fazer este relatório final foram:

- **Plataforma Overleaf**, para a escrita do relatório(em Latex) por ambos os membros do grupo simultaneamente ;
- **Plataforma Draw io**, para a construção dos fluxogramas;
- **Plataforma Discord** , para a comuniçao dos membros do grupo;
- **Programa Dev-C ++**, para a escrita do código em C.

2

Descrição do Problema

Enunciado do problema:

O objectivo deste trabalho prático visa a compreensão dos princípios de resolução de problemas com base numa solução algorítmica.Pretende-se que os alunos construam e validem um algoritmo que permita resolver de forma eficiente o problema proposto.

É 2084 e o ano do Big Brother finalmente chegou. Para exercer maior controle sobre os seus cidadãos o governo decide implementar uma medida radical - todos devem ter um nano chip cirurgicamente implantado.

Este computador conterá todas as informações pessoais, bem como um transmissor que permitirá que os movimentos sejam registados e monitorizados.

Cada chip terá um código de identificação único, que consiste em 50 caracteres escolhidos a partir do alfabeto (26 letras minúsculas). Este conjunto de caracteres é escolhido de forma (quase) aleatória. Dado que escrever códigos completamente aleatórios no chip é mais dispendioso, o fabricante produz códigos que são rearranjos de outros códigos, com uma seleção diferente de letras. Assim, assim que um conjunto de letras for escolhido, todos os códigos possíveis derivados dele são usados antes de mudar o conjunto.

Por exemplo,suponha que um código conterá exatamente 3 ocorrências de 'a', 2 de 'b' e 1 de 'c', então 3 dos 60 códigos permitidos nestas condições são:

abaabc

abaacb

ababac

Esses três códigos são listados de cima para baixo em ordem alfabética.

A sua missão, caso aceite, será escrever e refinar um algoritmo que permita resolver o problema supra citado. O algoritmo deverá aceitar uma sequência de não mais do que 50 letras minúsculas (pode conter repetições) e imprimir o código sucessor, ou a mensagem "Sem sucessor" se o código fornecido for

o último da sequência para esse conjunto de caracteres.

Inputs

Uma série de linhas, cada uma contendo uma string representando um código. O ultimo input será o caracter#

Output

Uma linha por cada código colocado no input com o código seguinte (na sequência), ou com as palavras "sem sucessor"

Input Exemplo

abaacb

cbbaa

#

Output Esperado

ababac

Sem sucessor

Observações: Após a leitura do enunciado do problema cheguei à conclusão que o programa que teria que criar tinha como objetivo ler uma sequência de caracteres minúsculos e teria que imprimir no programa a sequência seguinte, após ter feito todas as permutações possíveis para isso recorri a um pensamento matemático de permutações.

Nas páginas que se seguem encontra se a descrição de um algoritmo, com o seu respetivo fluxograma e exemplos de testes para saber se estava tudo a funcionar corretamente.

3

Pressupostos

3.1. Algoritmo não Refinado

1. Pedir ao utilizador que digite # quando não quiseres escrever mais caracteres.

1.1. Ler sequência;

2. Verificar se a sequência lida está em letras minúsculas e não contem mais de 50 caracteres,

2.1. Se se verificar estas condições avançar para o terceiro passo

2.2. Caso isso não aconteça é imprimida uma mensagem de erro no programa;

3. Analisar as últimas duas letras da sequência inserida;

3.1. Posição da penúltima letra = A e posição da última letra = B.

3.2. Se letra de A < letra de B (se a letra da penúltima posição for **menor** que a letra da ultima posição, para auxilio utilizamos a tabela de ASCII);

3.3. Trocar letra de **A** por uma letra maior mas que seja a menor das restantes após estarem por ordem alfabética.

3.4. Escrever a sequência seguinte

4. Se B é igual a 0.(se a posição da ultima letra for 0)

4.1. Escrever "sem sucessor".

5. Letra de A > letra de B (e a letra da penúltima posição for **maior** que a letra da ultima posição, para auxilio utilizamos a tabela de ASCII);

5.1. **A** vai ser igual A-1 e tornar **B** igual a B-1.(ou seja a posição da penúltima letra é igual a posição anterior e isto também acontece com a posição da ultima letra)

5.2. Voltar ao passo 3.2

6. O programa pára quando o utilizador prime o #.

3.2. Algoritmo Refinado

1. Inicio

1.1. ler(string[50]);

2. Chama a funçao "principal"

2.1. principal(char string[50]);

3. se strlen(string[50])>50 escrever "Erro excedeu tamanho máximo";

3.1. Volta a pedir outra string ,le e volta a chamar a funçao principal;

4. se nao strcmp(string,'#')!=0

5. Chama função maiusculas e verifica se a string contem letras maiusculas se sim escrever "ERRO!!!"

5.1. Volta a pedir outra string ,le e volta a chamar a funçao principal;

6. tam $\leftarrow strlen(string); B \leftarrow tam - 1; A \leftarrow tam - 2$

Enquanto string[A]>=string[B]

A$\leftarrow A - 1$;

B$\leftarrow B - 1$;

FimEnquanto

Se string[A]<string[B] && B!=0

strA $\leftarrow string[A]$;

Se strlen(string)==2;

string[A]=string[B];

string[B]=strA;

Escreve string e pede uma nova sequência lê e volta a chamar a função principal

Se não chama função trocas

trocas(char a[ ], char min) ;

novoA← $trocas(\&string[A], string[A])$;

string[A]=novoA;

6.1. Percorre a string toda para colocar em ordem alfabetica as restantes letras da string ;

Escreve string e volta a pedir uma nova sequência lê e volta a chamar a função principal

Se B==0

Escreve "Sem sucessor"e volta a pedi uma sequência, lê e volta a chamar a função principal;

Se nao

exit(0);

FimSeNão

3.3. Fluxogramas

OBSERVAÇÃO:A função troca desempenha a troca da letra de A por uma letra maior que esteja na string verificando se a mesma é a menor das restantes letras para isso recorremos à tabela de ASCII para verificar.

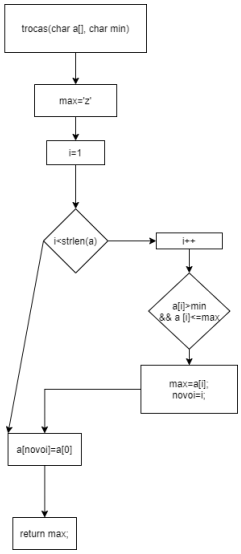

Figura 3.1: Fluxograma função trocas

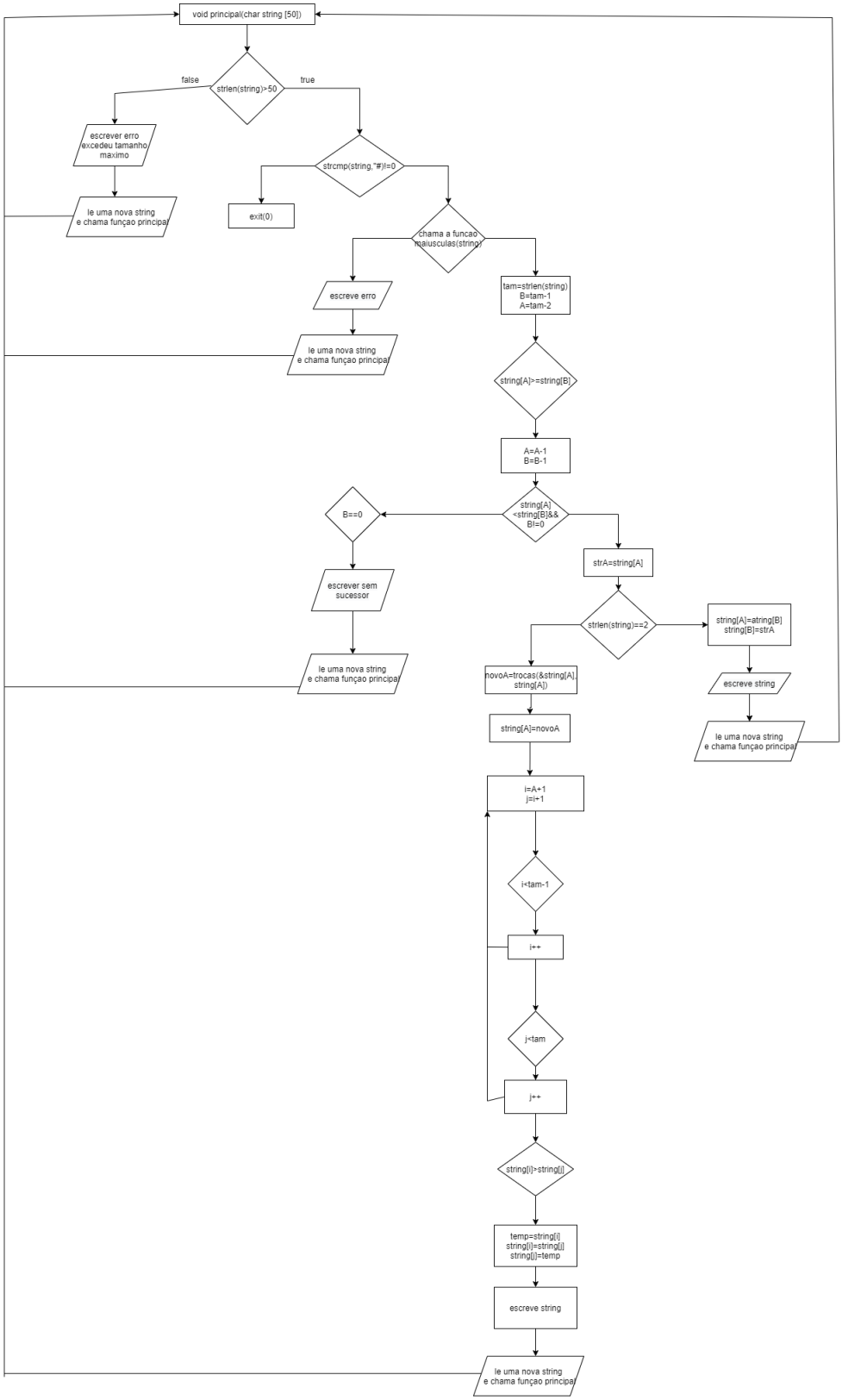

Figura 3.2: Fluxograma função principal

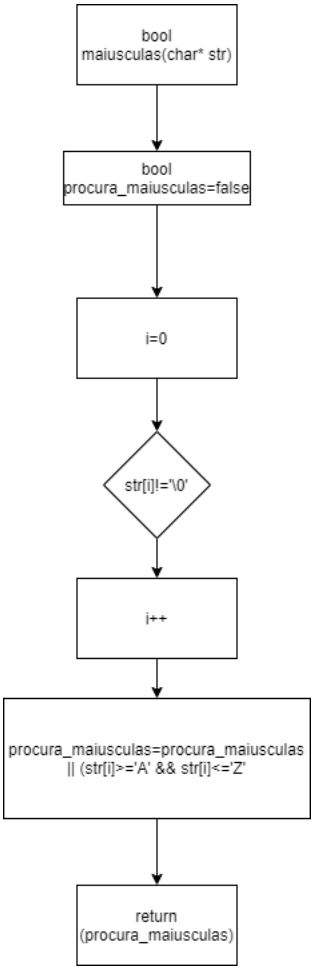

Figura 3.3: Fluxograma função maiusculas

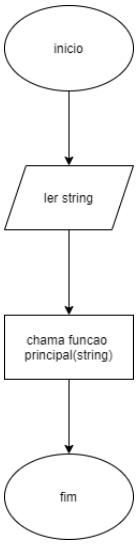

Figura 3.4: Fluxograma função main

4

Descrição Algorítmica

Para uma melhor compreensão do que o programa vai fazer fiz testes para verificar se este algoritmo que propus esta a funcionar corretamente.

EXEMPLO 1:

Supondo que o input é **"a b a a c b"**:

O programa verifica se esta string está em letras minúsculas e procede para o passso 3, neste caso o input esta em minúsculas logo o programa continua.

Tabela 4.1: Posição Inicial das letras da string no vetor

					A	B
Posição	**0**	**1**	**2**	**3**	**4**	**5**
Letra	**a**	**b**	**a**	**a**	**c**	**b**

No passo **3** é atribuída a Posição da penúltima (A) =4 e posição da ultima (B)=5

Depois vamos verificar neste caso se c<b(letra de A < letra de B, esta afirmação é falsa logo vamos saltar agora para o passo **5.1** do algoritmo.

Vamos tornar A=A-1 e B=B-1, neste exemplo o A vai tomar o valor de 3 e o B o valor de 4.

Tabela 4.2: Posição das letras da string no vetor

					A	B	
Posição	**0**	**1**	**2**	**3**	**4**	**5**	
Letra	**a**	**b**	**a**	**a**	**c**	**b**	

E agora vamos voltar a verificar se a<c(Letra de A<Letra de B esta afirmação é verdade logo vamos chamar a função troca

Trocar letra de A por uma letra maior mas ao mesmo tempo que seja menor das restantes e organizar as restantes por ordem alfabética

.

Tabela 4.3: Posição das letras da string no vetor após a função troca

Posição	**0**	**1**	**2**	**3**	**4**	**5**
Letra	**a**	**b**	**a**	**b**	**a**	**c**

Após esta etapa é imprimido no programa a string **"a b a b a c"** depois disto o programa pára quando o input do utilizador for "#".

EXEMPLO 2:

Supondo que o input é **"c b b a a"**:

O programa verifica se esta string está em letras minúsculas e procede para o passso 3, neste caso o input esta em minúsculas logo o programa continua.

Tabela 4.4: Ex.2-Posição Inicial das letras da string no vetor

				A	B
Posiçao	**0**	**1**	**2**	**3**	**4**
Letra	**c**	**b**	**b**	**a**	**a**

No passo **3** é atribuída a Posição da penúltima (A) =3 e posição da ultima (B)=4

Depois vamos verificar neste caso se a<a(letra de A < letra de B, esta afirmação é falsa logo vamos saltar agora para o passo **5.1** do algoritmo.

Vamos tornar A=A-1 e B=B-1, neste exemplo o A vai tomar o valor de 2 e o B o valor de 3.

Tabela 4.5: Ex.2-Posição das letras da string no vetor

			A	B	
Posiçao	**0**	**1**	**2**	**3**	**4**
Letra	**c**	**b**	**b**	**a**	**a**

E agora vamos voltar a verificar se b<a(Letra de A<Letra de B esta afirmação é falso logo vamos voltar ao passo **5.1** do algoritmo.

Vamos tornar novamente A=A-1 e B=B-1, neste caso agora o A vai tomar o valor de 1 e o B o valor de 2.

Tabela 4.6: Ex.2-Posição das letras da string no vetor

		A	B		
Posiçao	**0**	**1**	**2**	**3**	**4**
Letra	**c**	**b**	**b**	**a**	**a**

Voltamos a verificar se b<b(Letra de A<Letra de B esta afirmação é falso logo vamos voltar ao passo **5.1** do algoritmo.

Vamos tornar novamente A=A-1 e B=B-1, neste caso agora o A vai tomar o valor de 0 e o B o valor de 1.

Tabela 4.7: Ex.2-Posição das letras da string no vetor

	A	B			
Posiçao	**0**	**1**	**2**	**3**	**4**
Letra	**c**	**b**	**b**	**a**	**a**

Voltamos a verificar se c<b(Letra de A<Letra de B esta afirmação é falso logo vamos voltar ao passo **5.1** do algoritmo.

Tabela 4.8: Ex.2-Posição das letras da string no vetor

	B				
Posiçao	**0**	**1**	**2**	**3**	**4**
Letra	**c**	**b**	**b**	**a**	**a**

Daqui verificamos que a letra de B ==0 Logo o output do programa será a mensagem "Sem Sucessor".

Tabela 4.9: Ex.2-Posição das letras da string no vetor após todas as trocas

Posição	**0**	**1**	**2**	**3**	**4**
Letra	**c**	**b**	**b**	**a**	**a**

Após esta etapa é imprimido no programa a string **"c b b a a "** depois disto o programa pára quando o input do utilizador for "#".

EXEMPLO 3:

Supondo que o input é **"a b"**:

O programa verifica se esta string está em letras minúsculas e procede para o passso 3, neste caso o input esta em minúsculas logo o programa continua.

Tabela 4.10: Ex.3-Posição inicial das letras da string no vetor

	A	B
Posiçao	**0**	**1**
Letra	**a**	**b**

No passo **3** é atribuída a Posição da penúltima (A) =0 e posição da ultima (B)=1

Depois vamos verificar neste caso se a<b(letra de A < letra de B, esta afirmação é verdadeira logo vamos chamar a função troca

Trocar letra de A pela letra maior que A mas a menor das restantes e organizar as restantes por ordem alfabética

.

Tabela 4.11: Ex.3-Posição das letras da string no vetor após a função troca

Posição	**0**	**1**
Letra	**b**	**a**

Após esta etapa é imprimido no programa a string **"b a "** depois disto o programa pára quando o input do utilizador for "#".

EXEMPLO 4:

Supondo que o input é **"a c b"**:

O programa verifica se esta string está em letras minúsculas e procede para o passso 3, neste caso o input esta em minúsculas logo o programa continua.

Tabela 4.12: Ex.4-Posição inicial das letras da string no vetor

		A	B
Posiçao	**0**	**1**	**2**
Letra	**a**	**c**	**b**

No passo **3** é atribuída a Posição da penúltima (A) =1 e posição da ultima (B)=2

Depois vamos verificar neste caso se a<a(letra de A < letra de B, esta afirmação é falsa logo vamos saltar agora para o passo **5.1** do algoritmo.

Tabela 4.13: Ex.4-Posição das letras da string no vetor

	A	B	
Posiçao	**0**	**1**	**2**
Letra	**a**	**c**	**b**

Vamos tornar A=A-1 e B=B-1, neste exemplo o A vai tomar o valor de 0 e o B o valor de 1.

E agora vamos voltar a verificar se a<c(Letra de A<Letra de B esta afirmação é verdadeira logo vamos trocar letra de A pela letra maior que A mas que seja a menor das restantes e organizar as restantes por ordem alfabética.

Tabela 4.14: Ex.4-Posiçao final das letras da string no vetor

Posiçao	**0**	**1**	**2**
Letra	**b**	**a**	**c**

Após esta etapa é imprimido no programa a string **"b a c "** depois disto o programa pára quando o input do utilizador for "#".

EXEMPLO 5:

Supondo que o input é **"A B"**:

O programa verifica se esta string está em letras minúsculas e procede para o passso 3, neste caso o input esta em maiusculas logo o programa imprime a mensagem "ERROO!!!".

5

Código em Linguagem C

Excerto de código 5.1: Código em Linguagem C

```
//Project MIETI 2020/2021 MPI
//Made by:Beatriz Alves
//Final Version
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include <stdbool.h>//library that uses bool function
//function that check uppercase
bool maiusculas(char* str) {
    int  i;
    bool found_upper = false;
    for (int i = 0; str[i] != '\0'; i++) {//this for used to scroll through the entire string
        and verify that the string characters are uppercase
        found_upper = found_upper || (str[i] >= 'A' && str[i] <= 'Z');
    }
    return (found_upper);
}
//function swap
char trocas(char a[], char min)
{
   char max = 'z';
   int novoi;
    for (int i=1; i<strlen(a); i++){
        if (a[i] > min && a[i]<= max){
         max=a[i];
         novoi=i;
      }
   }
   a[novoi] = a[0];
    return max;
}
void principal(char string[50]){
   int A,B,tam;
   char temp;
//this if checks that the string does not contain more than 50 characters
   if(strlen(string)>50){//if string is longer than 50 characters the error message is printed
       on the screen and the user is asked to enter a valid sequence
      printf("Erro excedeu tamanho maximo.(50 caracteres )\n\n");
      printf("Insira uma sequencia:\n");
      scanf("%s",string);
      principal(string);
   }
//this if checks if the string and equals #
   if (strcmp(string,"#")!=0){ //if the string is equal to # the program closes
   //if the string is different from # the program continues to the next if it will check if
       the string is only in minuscule letters
      if(maiusculas(string) ){//this if calls the function maiusculas which is a function that
          checks whether in the string there are uppercase letters,
      //if the string has uppercase characters and the error message is suppressed and asked
          the user to enter a new string.
       printf("ERRO!!!\n\n");
       printf("Insira uma sequencia:\n");
      scanf("%s",string);
      principal(string);
       }
```

```
//strlen of the string is the length of the string
//subtracts 1 to tam to go to the last position once the positions start at zero dai that
    B=tam-1
      //define the variables
     tam=strlen(string);
     B=tam-1;// position of the last letter of the string
     A=tam-2;//position of the penultimate letter of the string
     while(string[A]>=string[B]){//while letter of A is greater than or equal to the letter
         of position B
     //  A and B will take the value of the position prior to them respectively
        A=A-1;
        B=B-1;
        //printf("A:%c::::B:%c",string[A],string[B]);------THIS COMMENT WAS USED THROUGHOUT
            THE PREPARATION OF THE CODE TO GO CHECKING IF ERRORS OCCURRED IN THE PROGRAM
     }
      if(string[A]<string[B] && B!=0) {//If the letter of position A is less than the letter
          of position B and at the same time position B is different from 0
        //shortcut to two-letter strings
        char strA=string[A];
        if(strlen(string)==2){
           string[A]=string[B];
           string[B]=strA;
           printf("%s\n\n",string);
           printf("Insira uma sequencia:\n");
           scanf("%s",string);
           principal(string);
         }
         //strings with more than two letters
        else{
           char novoA;
           //printf("A->%d,,,B->%d",A, B);------THIS COMMENT WAS USED THROUGHOUT THE
               PREPARATION OF THE CODE TO GO CHECKING IF ERRORS OCCURRED IN THE PROGRAM
           novoA=trocas(&string[A], string[A]);// & is used for function only to use the
               letters that are to the right of the letter of position A
           string[A]=novoA;
           //puts in alphabetic order
           for (int i = A+1; i < tam-1; i++) { //this for scrolling through the entire string
               and checks whether the Position of B<B
              for (int j = i+1; j < tam; j++) {//this for scrolling through the entire string
                  and checks whether the Position of B+1<tam
                 if (string[i] > string[j]) {//this if goes checking if the letter of B > B+1
                     letter
                    temp = string[i];
                    string[i] = string[j];
                    string[j] = temp;
                 }
              }
           }
           printf("%s\n\n",string);
           printf("Insira uma sequencia:\n");
           scanf("%s",string);
           principal(string);
         }
     }
     if(B==0){//if the position of B is equal to 0 the program prints the message "no
         successor" and asks the user to enter a new sequence
        printf("Sem sucessoooooor\n\n");
        printf("Insira uma sequencia:\n");
        scanf("%s",string);
        principal(string);
     }
   }
   else {
      exit (0);
   }
}
int main(){
   char string[50];
   //header
    printf(".....................................................................\n" );
   printf("| d888888b d888b.     .o88b. .d88b. d8888b. d88888b .d8888. |\n");
   printf("| `88'   88  `8D     d8P  Y8 .8P Y8. 88 `8D 88'     88'  YP  |\n");
   printf("|  88    88   88     8P      88    88 88  88 88ooooo `8bo.   |\n");
   printf("|  88    88   88     8b      88    88 88  88 88~~~~~  `Y8b.  |\n");
   printf("| .88.   88  .8D     Y8b  d8 `8b  d8' 88 .8D 88.    db   8D  |\n");
   printf("| Y888888P Y8888D'    `Y88P'  `Y88P'  Y8888D' Y88888P `8888Y' |\n");
   printf("|.....................................................|\n" );
   printf("\n");
   printf("Insira uma sequencia:\n");
```

```
    scanf("%s",string);
    principal(string);//calls the function "principal"
    return 0;
}
```

6

Testes e Análises

Neste capitulo vamos testar o programa em C para isso vamos utilizar os mesmos exemplos já mostrados no capitulo descrição algoritmica e verificar se esta tudo a funcionar corretamente.

Exemplo 1:

INPUT :abaabc

OUTPUT:abaacb

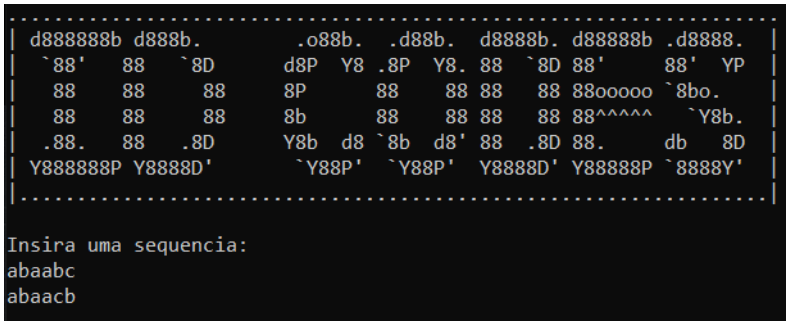

Figura 6.1: Exemplo 1

O programa esta a funcionar corretamente para este primeiro exemplo agora falta testar os outros.

Exemplo 2:

INPUT :cbbaa

OUTPUT:sem sucessor

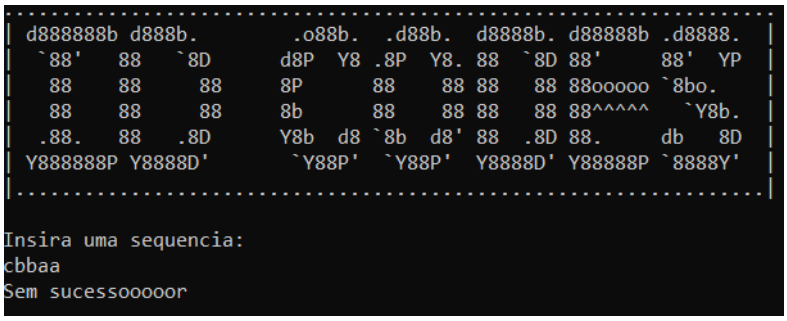

Figura 6.2: Exemplo 2

Exemplo 3:

INPUT :ab

OUTPUT:ba

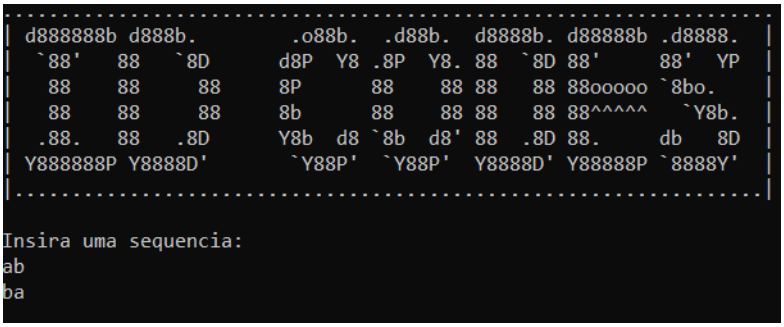

Figura 6.3: Exemplo 3

Exemplo 4:

INPUT :acb

OUTPUT:bac

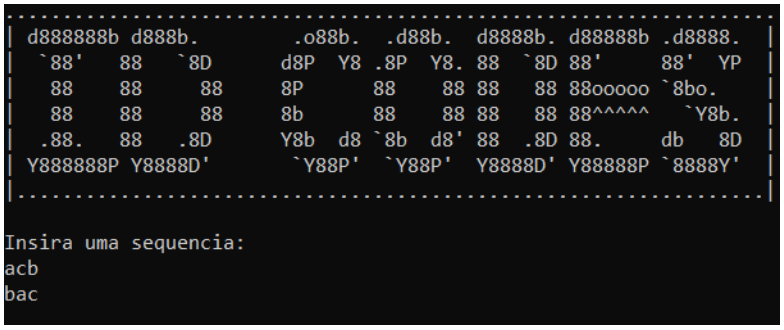

Figura 6.4: Exemplo 4

Exemplo 5:

INPUT :AB

OUTPUT:ERRO!!

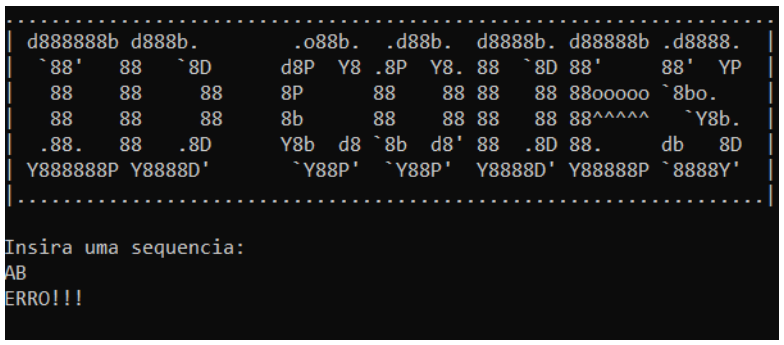

Figura 6.5: Exemplo 5

Após a execução de todos os exemplos no programa em C verificamos que está tudo a funcionar corretamente sem nenhum tipo de erro e chegando assim a conclusão do projeto.

OBS: Uma vez que fiquei curiosa para testar o meu nome (sendo o nomes a dita sequência) no programa decidi colocar aqui também esses exemplo.

Exemplo nome:

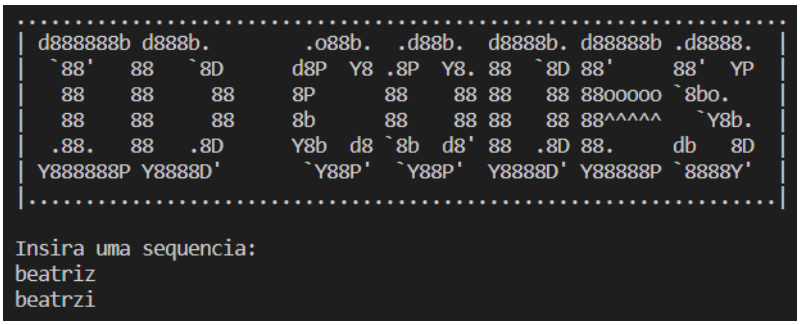

Figura 6.6: Exemplo nomes

7 Conclusão

Após a conclusão desta ultima fase, posso afirmar que fiquei a compreender melhor os conceitos apreendidos, as dificuldades que foram surgindo ao longo desta e da primeira fase rapidamente foram ultrapassadas, apesar da minha inexperiência inicial em algumas matérias.

O problema algorítmico que nos foi colocado era um problema que deu que pensar, contudo acho que o resolvi da melhor forma sendo esta foi a minha proposta e para obter esta solução do problema foi crucial gerir o tempo em cada tarefa para assim concluir este projeto com êxito .