BCC 201 - Introdução à Programação I

Cadeia de caracteres (*strings*)

Guillermo Cámara-Chávez
UFOP

Caracter I

- Um caracter é considerado um tipo de dado primitivo
- Um tipo de dado é primitivo se o computador possui instruções em linguagem de máquina que permite a manipulação deste tipo.
- Desde que uma cadeia é uma sequência ordenada de caracteres, o caracter é a entidade fundamental de manipulação de uma cadeia.

Caracter II

- Um caracter pertence a um conjunto finito de caracteres: um alfabeto
- Um exemplo de alfabeto é o conjunto de letras da língua portuguesa
- Outro alfabeto comum é o conjunto de dígitos decimais
- Ao longo dos anos, vários alfabetos foram desenvolvidos para serem utilizados em computadores

Caracter III

- Caracteres literais são representados por aspas simples, como em 'A' e 'a'
- Variáveis do tipo *char* podem receber valores literais do tipo caracter ou também valores inteiros (que nesse caso representam o caracter correspondente, conforme o sistema de codificação adotado)
- Variáveis do tipo *char* podem também ter o seu valor comparado com inteiros

Caracter IV

- Entre os vários métodos de codificação, os mais populares são
  - código ASCII (7 bits) - *American Standard Code for Information Interchange*
  - código EBCDIC (8 bits) - *Extended Binary Coded Decimal Interchange Code*
  - código UNICODE (8 bits)
  - código UTF-8, nos dias de hoje substitui o sistema ASCII

Caracter V

- Tabela ASCII

	0	1	2	3	4	5	6	7	8	9
30			sp	!	"	#	$	%	&	'
40	(	)	*	+	,	-	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[	\	]	^	_	`	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	S	t	u	v	w
120	x	y	z	{	|	}	~			

Caracter VI

- Os dígitos são codificados sequencialmente na tabela ASCII: '0' (48), '1' (49), etc.
- O código a seguir verifica se um caracter é um digito (de 0 a 9)

```
int digito (char c)
{
    if (c >= '0' && c <= '9')
        return 1;
    else
        return 0;
}
```

Caracter VII

- Para converter de letra minúscula para maiúscula (usando codificação sequencial)

```
char maiuscula (char c)
{
    if (c >= 'a' && c <= 'z')
    {
        c = c -'a' + 'A';
    }
}
```

Caracter VIII

- `#include <cctype>`

Função	Descrição
isalnum(c)	true se *c* é uma letra ou dígito
isalpha(c)	true se *c* é uma letra
isblank(c)	true se *c* é espaço ou tabulação
isdigit(c)	true se *c* é um digito
islower(c)	true se *c* é uma letra minúscula
isupper(c)	true se *c* é uma letra maiúscula
ispunct(c)	true se *c* é um símbolo de puntação
tolower(c)	retorna a versão minúscula do caracter *c*
toupper(c)	retorna a versão maiúscula do caracter *c*

Cadeia de caracteres I

- Definição: são sequências de letras, números ou símbolos onde o último caracter é o caracter nulo (\0)
- Na linguagem C utilizamos vetores de tipo `char` para armazenar cadeias de caracteres.

Cadeia de caracteres II

- Por exemplo, para declarar um espaço de memória que contenha 20 caracteres fazemos:

```
char nome[20];
```


```
cout << nome;
```

- A função `cin`, quando utilizada com *strings* lê todos os caracteres até encontrar um caracter igual a espaço ou fim de linha

Cadeia de caracteres III

- Para manipular este tipo de *string* é preciso ter certo cuidado (se **acessamos um endereço fora do vetor**, invadiremos outras áreas da memória).
- Pergunta: Qual o tamanho do menor vetor que devemos declarar para armazenar uma cadeia de caracteres com 10 letras?

Cadeia de caracteres IV

- Escreva um program que lê uma palavra da entrada e imprime o número de caracteres desta palavra

Cadeia de caracteres V

```
int main()
{
    char vetor[100];
    int i, n;
    cout << "Entre com a palavra: ";
    cin >> vetor;
    i = 0;
    n = 0;
    while(vetor[i++] != '\0')
        n++;
    cout << "O numero de caracteres eh " << n;
    return 0;
}
```

Cadeia de caracteres em C++ I

- Para utilizar *strings* em C++ é necessário incluir o arquivo *header*
  - `#include <string>`
- *Strings* são declaradas da mesma forma que os tipos de dados primitivos, tais como inteiros ou reais, utilizando o tipo string

```
// declaracao de variaveis tipo string
string s1, s2, s3;
```

Cadeia de caracteres em C++ II

- Inicialização e atribuição de *strings*

```
#include<iostream>
#include<string> // necessario para usar strings
using namespace std;
int main(){
    string nome1("Fulano"); // inicializa nome1
    string nome2 = "Beltrano"; // inicializa nome1

    // nao inicializa nome3 e nome4
    string nome3, nome4;

    string nome5(10, 'c'); // s4 eh ccccccccccc
    nome3 = "Ciclano";
    nome4 = nome3;
    ...
}
```

Cadeia de caracteres em C++ III

- Acesso a caracteres individuais de uma *string*
  - Assimo como em C, C++ define o operador [] para permitir o acesso a caracteres individuais
  - Além disso, C++ também define o operado `at` que não permite o acesso a posições inválidas

Cadeia de caracteres em C++ IV

```
int main(){
    string nome1("Fulano");
    string nome2("Beltrano");

    // Troca o primeiro caracter de nome1
    nome1[0] = nome2[0];

    // usando o indice
    cout << "Decimo caracter: " << nome2[10];
    // ERRO! Nao existe o caracter 10

    // usando operador at
    cout << "Decimo caracter: " << nome2.at(10);
    // ERRO! mas aborta o programa
}
```

Cadeia de caracteres em C++ V

- Comparação de *strings*
  - em C++, pode-se usar os operadores $==$, $!=$, $>$, $>=$ ou $<$ para fazer comparações alfabéticas entre duas *strings*
  - Também pode-se empregar a função `compare` que retorna
    - = 0: ambos strings são iguais
    - > 0: o primeiro caracter que não casa tem um valor superior na primeira cadeia
    - < 0: o primeiro caracter que não casa tem um valor inferior na primeira cadeia

Cadeia de caracteres em C++ VI

- Manipulando o tamanho de uma *string*
  - As *strings* em C++ têm tamanho variável, *i.e.*, reserva uma certa quantidade de memoria (capacidade), mas não necessariamente utiliza toda essa memória
  - Caso a *string* cresça mais do que sua capacidade, mais memória é reservada

Cadeia de caracteres em C++ VII

- A seguir apresenta-se as principais funções:
  - `size()` ou `length()`: retorna o tamanho da *string*
  - `capacity()`: retorno a capacidade corrente da *string*, ou seja, quantos elementos ela poderá conter antes de precisar mais memoria
  - `max_size()`: retorna o tamanho máximo possível de uma *string*, geralmente depende da máquina e do compilador

Cadeia de caracteres em C++ VIII

```
int main(){
    string s1, s2("valor inicial"), s3[10];
    cout << "tamanho de s2: " << s2.size();
    return 0;
}
```

Cadeia de caracteres em C++ IX

- Concatenação de *strings*, é utilizado o operador + para "juntar" cadeias

```
string s1 = "Ciencia ", s2 = "da ", s3, s4;
s3 = "Computacao";
s4 = s1 + s2 + s3;
cout << s4;
```

Mostra na tela: "Ciencia da Computacao"

Cadeia de caracteres em C++ X

Quando concatenamos variáveis tipo *string* e cadeias ou caracteres, pelo menos um operando para cada operador + deve ser de tipo *string*

```
string s1 = "ola"; // atribuição

// ok: concatenação de um tipo string e um literal
string s4 = s1 + ", ";

// erro: nao existe um operando tipo string
string s5 = "hello" + ", ";

// ok: cada + tem um operando tipo string
string s6 = s1 + ", " + "world";

// erro: nao pode concatenar cadeias
string s7 = "hello" + ", " + s2;
```

Cadeia de caracteres em C++ XI

- Outras funções

Comandos	**Descrição**
`clear`	Limpa o texto
`insert`	Insere mais texto
`erase`	Apaga caracteres de texto
`replace`	Substitui parte do texto
`swap`	Troca conteúdo entre textos
`find`	Procura um texto dentro de outro
`empty`	Retorna *true* se a cadeia está vazia

Cadeia de caracteres em C++ XII

```
#include<string>
int main()
{
    string cad("Uva, maca, laranja");
    cout << cad << endl;
    cout << "Tamanho: " << cad.size() << endl;
    // "Tamanho: 18"
    cad.clear();
    cout << "Tamanho: " << cad.size() << endl;
    // "Tamanho: 0"
    return 0;
}
```

Cadeia de caracteres em C++ XIII

```
string& insert (size_t pos, const string& str);
```

pos : posição a partir de onde será inserida a nova cadeia

str : cadeia a ser inserida

```
string& erase (size_t pos = 0, size_t len = npos);
```

pos : posição do primeiro caracter que será apagado

len : número de caracteres que serão apagados

Cadeia de caracteres em C++ XIV

```
#include<string>
int main(){
    string cad("Uva, maca, laranja");
    cout << cad << endl;
    // "Uva, maca, laranja"
    cad.insert(5, "banana, ");
    cout << cad << endl;
    // "Uva, banana, maca, laranja"
    cad.erase(5, 8);
    cout << cad << endl;
    // "Uva, maca, laranja"
    return 0;}
```

Cadeia de caracteres em C++ XV

```
string& replace(size_t pos, size_t len, const string& str);
```

pos : posição do primeiro caracter que será substituído

len : número de caracteres que serão substituídas

str : cadeia que será copiada

Cadeia de caracteres em C++ XVI

```
#include <string>
int main()
{
    string cad("Uva, banana, laranja");
    cout << cad << endl;
    // "Uva, banana, laranja"
    cad.replace(5, 8, "abacaxi, pera, ");
    // "Uva, abacaxi, pera, laranja"
    cout << cad << endl;
    return 0;
}
```

Cadeia de caracteres em C++ XVII

```
void swap (string& str)

int main()
{
    string suco("Uva, banana, laranja");
    string sanduiche("pao, presunto, queijo");

    cout << "Antes da troca " << suco << endl;
    cout << "Antes da troca " << sanduiche << endl;
    // "Antes da troca Uva, banana, laranja"
    // "Antes da troca pao, presunto, queijo"

    suco.swap(sanduiche);

    cout << "Depois da troca " << suco << endl;
    cout << "Depois da troca " << sanduiche << endl;
    // "Depois da troca pao, presunto, queijo"
    // "Depois da troca Uva, banana, laranja"

    return 0;
}
```

Cadeia de caracteres em C++ XVIII

```
size_t find (const string& str, size_t pos = 0)
```

str : cadeia que será procurada

pos : a partir de onde será procurada a cadeia

Cadeia de caracteres em C++ XIX

```
int main(){
    string suco("Uva, banana, laranja, banana");

    int pos = suco.find("banana");
    cout << "Esta na posicao " << pos;
    // Esta na posicao 5

    pos = suco.find("banana", 6);
    cout << "Esta na posicao " << pos << endl;
    // Esta na posicao 23

    return 0;
}
```

Exercicios I

- Implementar uma função que crie uma cópia de um string

Exercicios II

```
string Copia(string cad);
int main()
{
    string cad, cadCopia;
    cout << "Inserir uma cadeia \n";
    cin >> cad;
    cadCopia = Copia(cadCopia);
    cout << "cadeia: << cad
         << "Copia: " << cadCopia;
    return 0;
}
```

Exercicios III

```
string Copia(string cad)
{
    string ncad;

    for (int i = 0; i < cad.size(); i++)
    {
        ncad += cad[i];
    }
    return ncad;
}
```

Exercicios IV

- Criar a função que receba um string e que gere outro *string* com a ordem inversa dos caracteres. Ex “alem” deve ser gerado “mela”.

Exercicios V

```
#include<string.h>
string Inverte(string);
int main()
{
     string cad1, cad2
     cout << "Inserir uma cadeia";
     cin >> cad1;
     cad2 = Inverte(cad1);
     cout << "cadeia1: " << cad1
          << "cadeia2: " << cad2;
     return 0;
}
```

Exercicios VI

```
string Inverte(string cad)
{
    int n = cad.size(), i, j;
    string cadinv(n, ' ');
    for (i = 0, j = n-1; i < n ; i++; j--)
        cadinv[i] = cad[j];
    return cadinv;
}
```

Exercicios VII

outra forma

```
string Inverte(string cad)
{
    string cadinv;
    for (int i = cad.size()-1; i >= 0 ; i--)
        cadinv += cad[i];
    return cadinv;
}
```

Exercicios VIII

- Criar uma função que elimine os espaços em branco de uma cadeia de caracteres

Exercicios IX

```
string Elimina(string cad);
int main()
{
    string cad, cadSem;
    cout << "Inserir uma cadeia \n";
    getline(cin, cad);
    cadSem = Elimina(cad);
    cout << "cadeia:  " << cadSem;
    return 0;
}
```

Exercicios X

```
string Elimina(string cad)
{
string ncad;
    for (int i = 0; i < cad.length(); i++)
        if (cad[i] != ' ')
            ncad += cad[i];
    return ncad;
}
```

Exercicios XI

- Criar as funções, LEFT e RIGHT, que retornem a subcadeia de n elementos a partir da esquerda e da direita respectivamente. Ex cad = "transformar". Mostrar os 3 primeiros caracteres a partir da esquerda: "tra". Mostrar os 5 primeiros caracteres a partir da direita: "ormar".

Exercicios XII

```
string Left(string , int);
string Right(string , int);
int main()
{
    string cad, cadRight, cadLeft;
    cout << "Inserir cadeia: ";
    cin >> cad1;
    cadleft = Left(cad, 4);
    cadRight = Right(cad, 4);
    cout << "Right " << cadRight << endl;
    cout << "Left " << cadLeft << endl;
    return 0;
}
```

Exercicios XIII

```
string Left(string cad, int n)
{
    string ncad;
    int tam = cad.length();

    if (n > tam)
    {
        n = tam;
    }

    for (int i = 0; i < n; i++)
    {
        ncad += cad[i];
    }
    return ncad;
}
```

Exercicios XIV

```
string Right(string cad, int n)
{
    string ncad;
    int tam = cad.length();

    if (n > tam)
    {
        n = tam;
    }
    for (int i = tam - n; i < tam; i++)
    {
        ncad += cad[i];
    }
    return ncad;
}
```

Exercicios XV

- O método `substr(ini_pos, numCaract)` da classe string em C++ extrai uma subcadeia de dimensão *numCaract* a partir da posição `ini_pos`. Por exemplo,

```
string palavra = "a maioria voltara";
cad = palavra.substr(2, 7);
```

  A variável cad contem a cadeia "maioria".

Exercicios XVI

Determinação da primeira ocorrência de uma subcadeia em uma cadeia.

Exercicios XVII

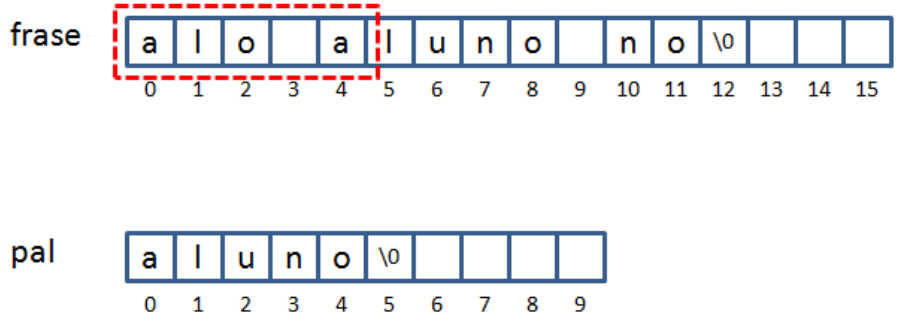
frase

a	l	o		a	l	u	n	o		n	o	\0			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

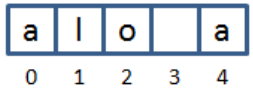
pal

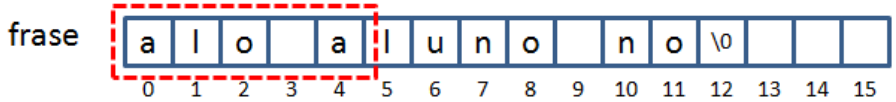
a	l	u	n	o	\0				
0	1	2	3	4	5	6	7	8	9

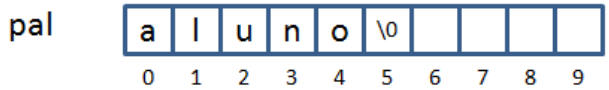
Exercicios XVIII



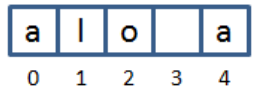
Exercicios XIX

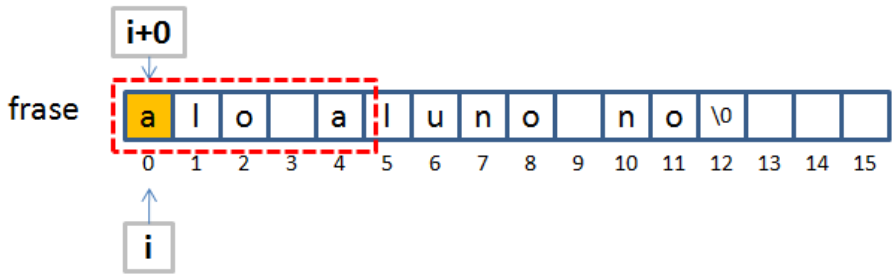


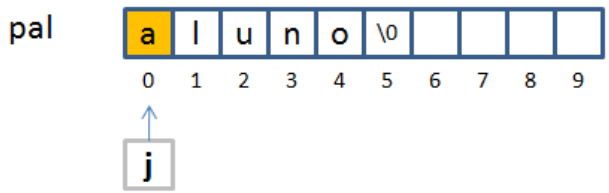




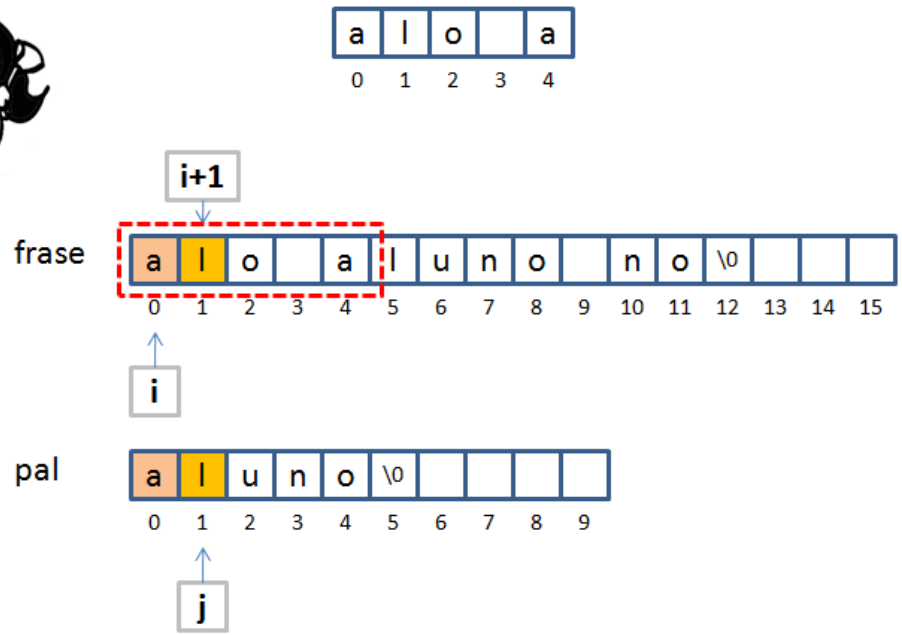
Exercicios XX



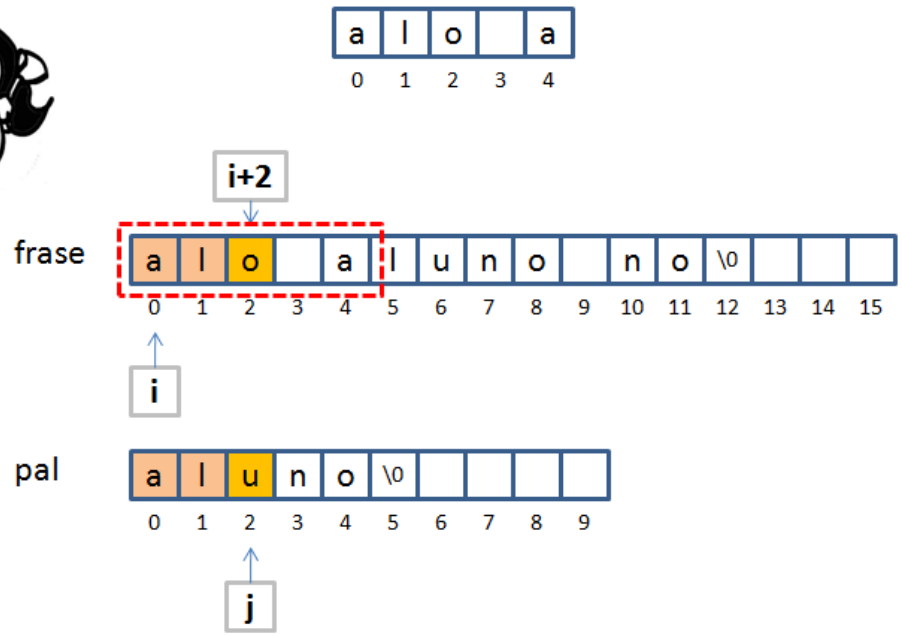




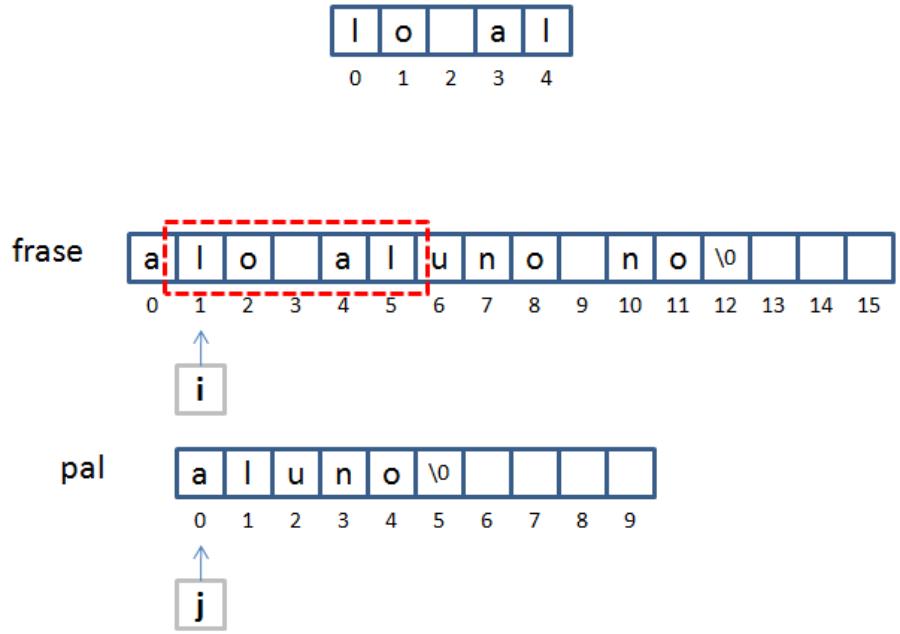
Exercicios XXI



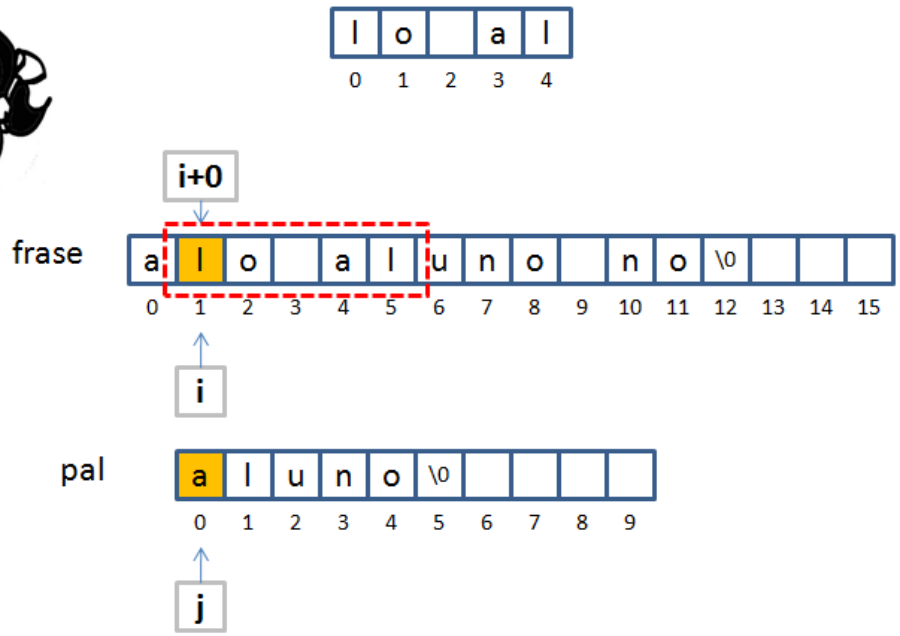
Exercicios XXII



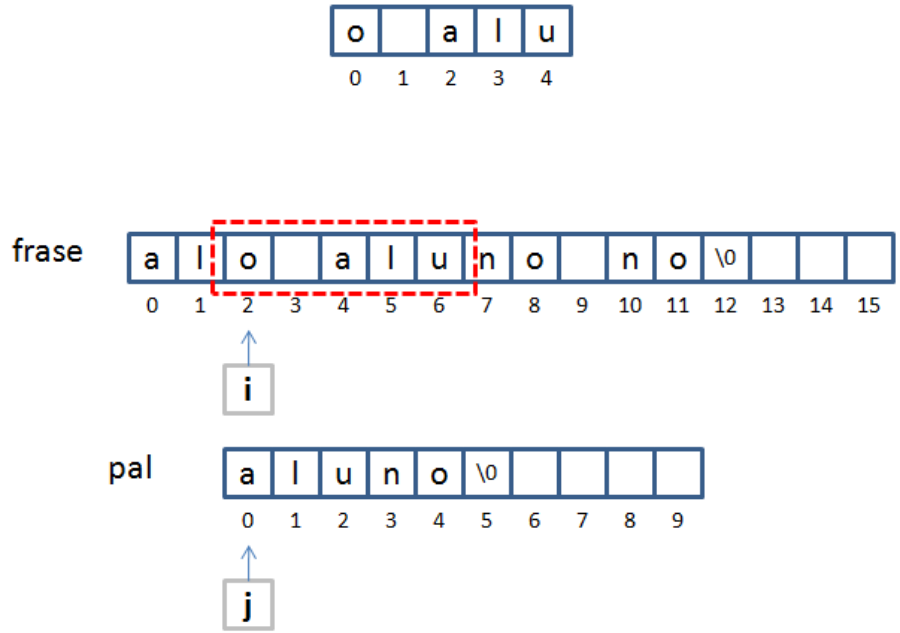
Exercicios XXIII



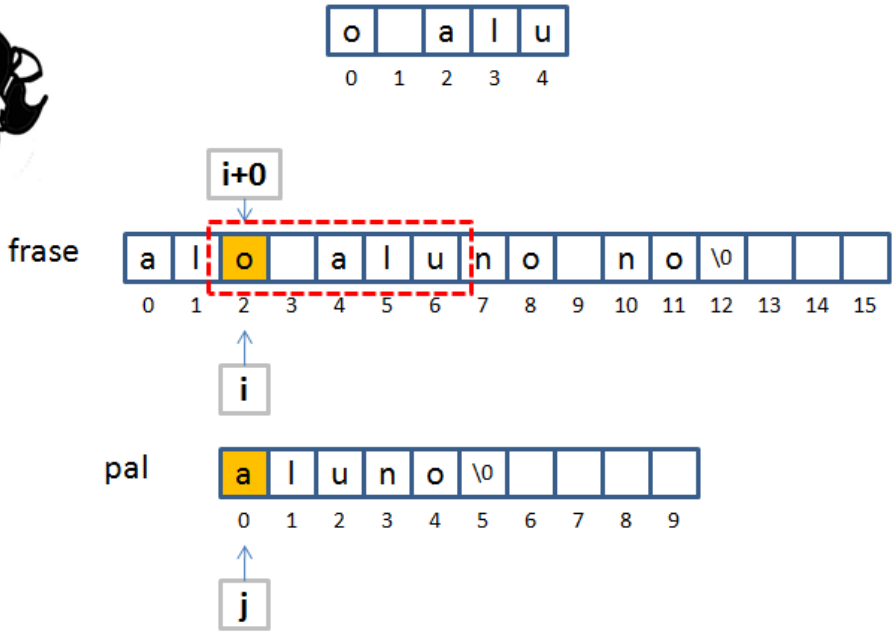
Exercicios XXIV



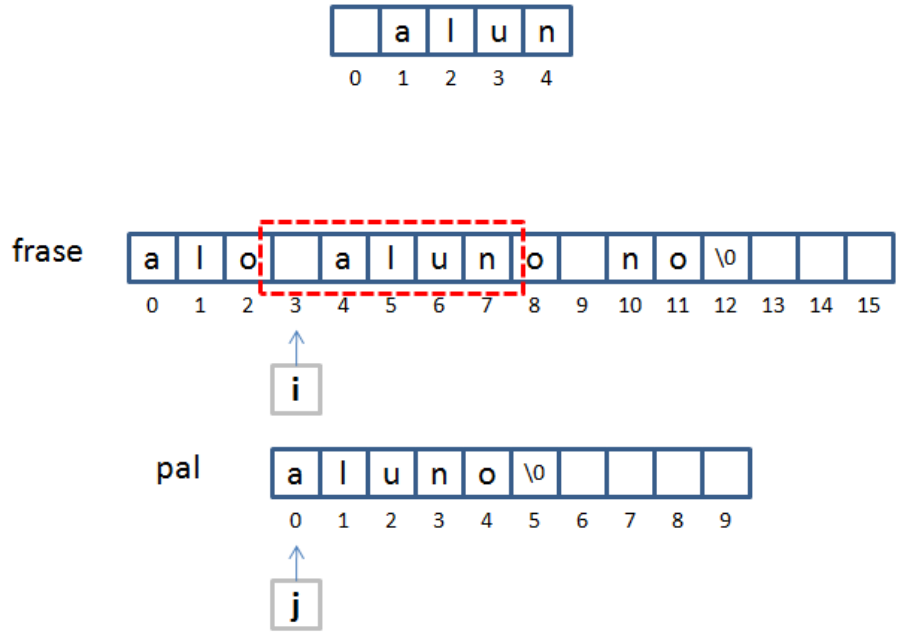
Exercicios XXV



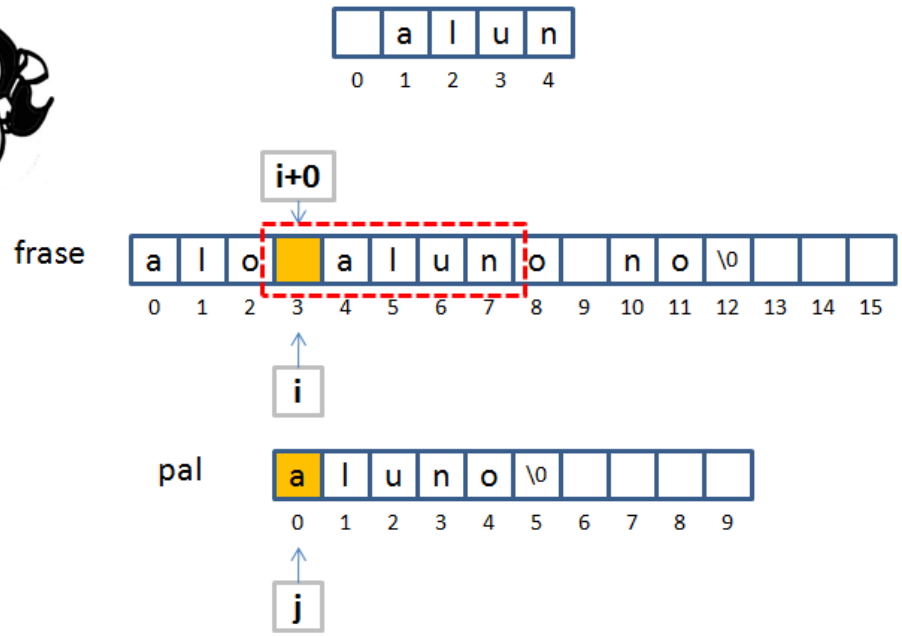
Exercicios XXVI



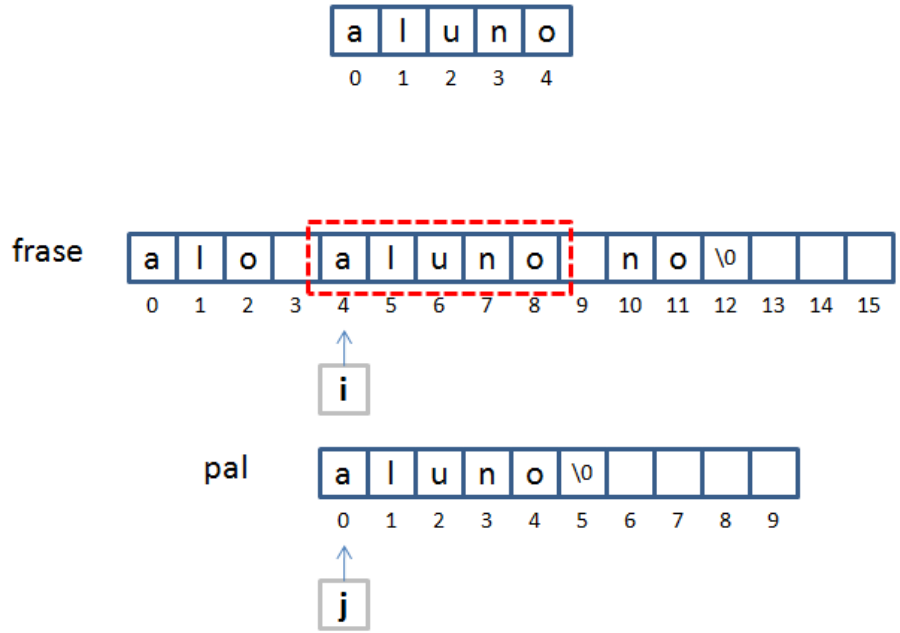
Exercicios XXVII



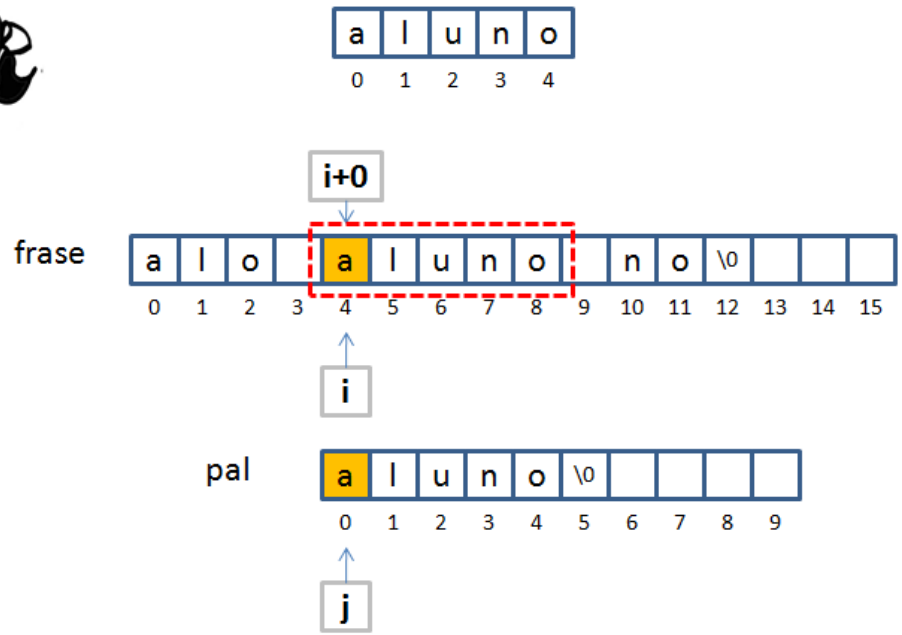
Exercicios XXVIII



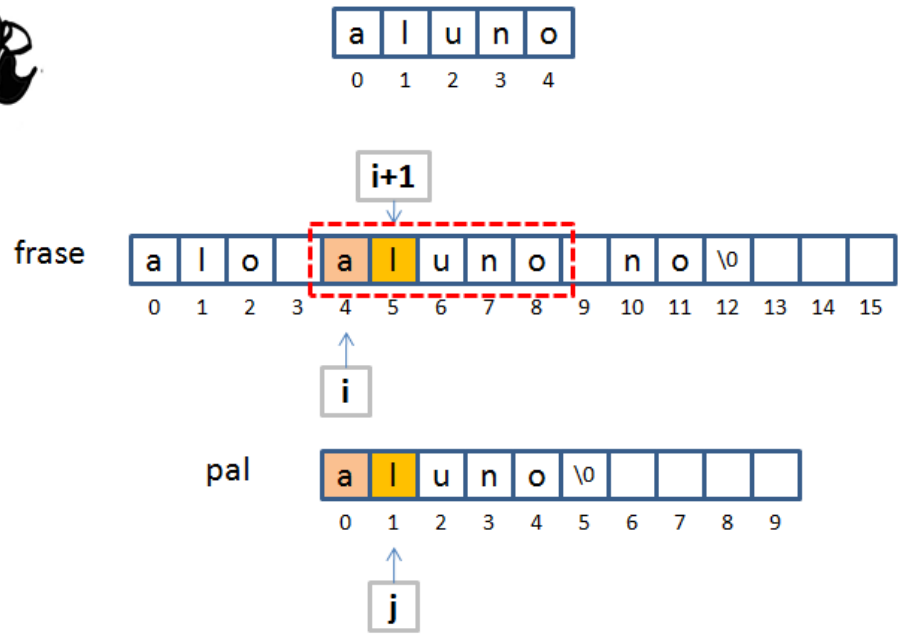
Exercicios XXIX



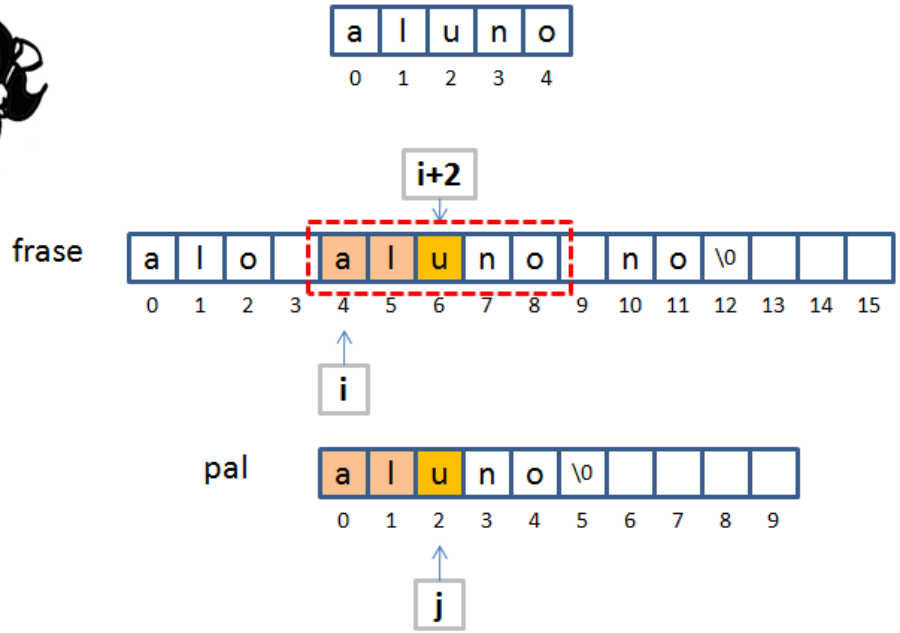
Exercicios XXX



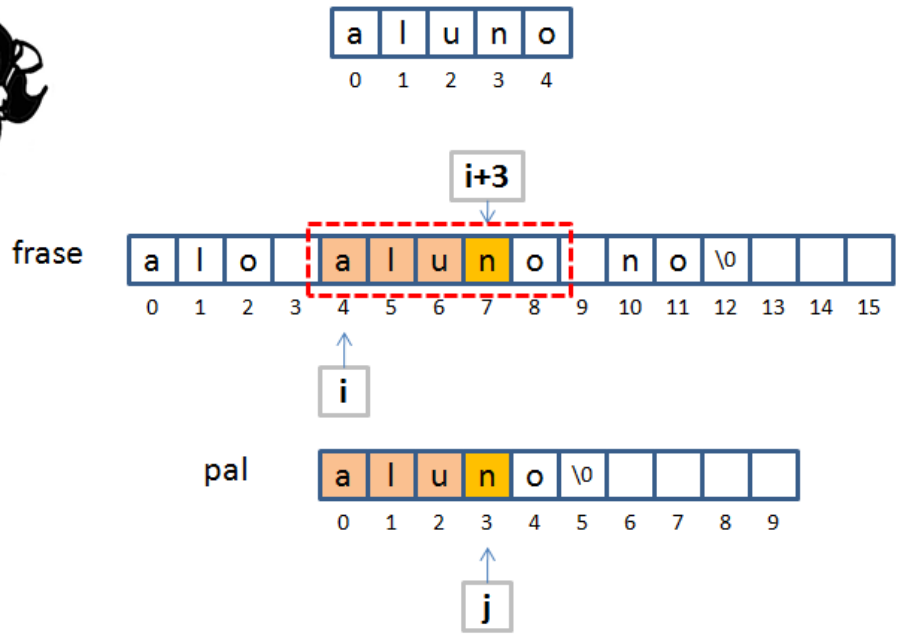
Exercicios XXXI



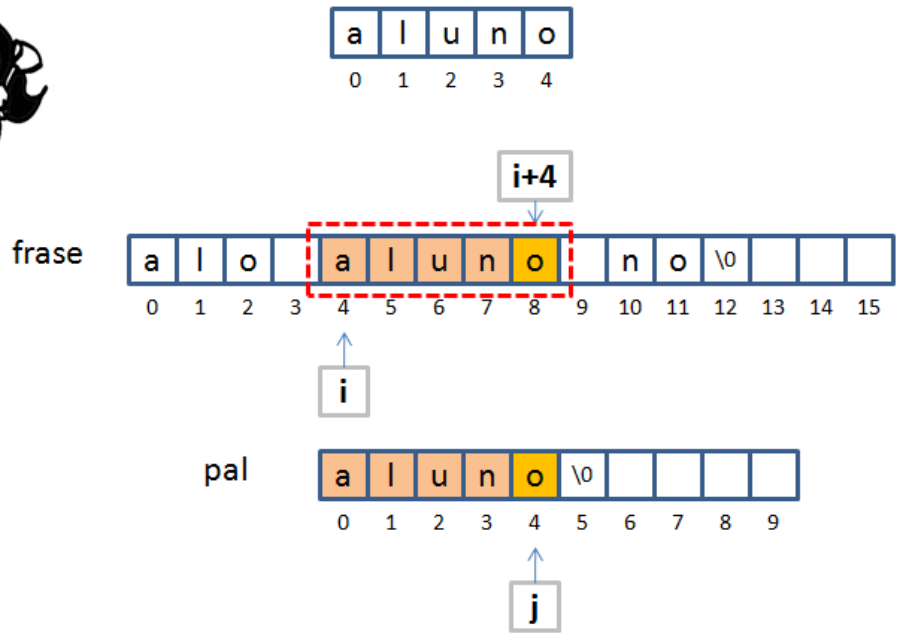
Exercicios XXXII



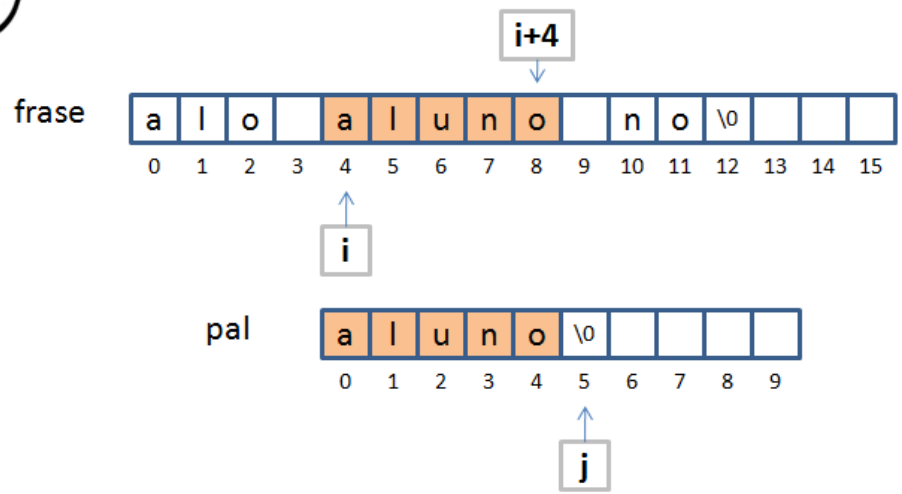
Exercicios XXXIII



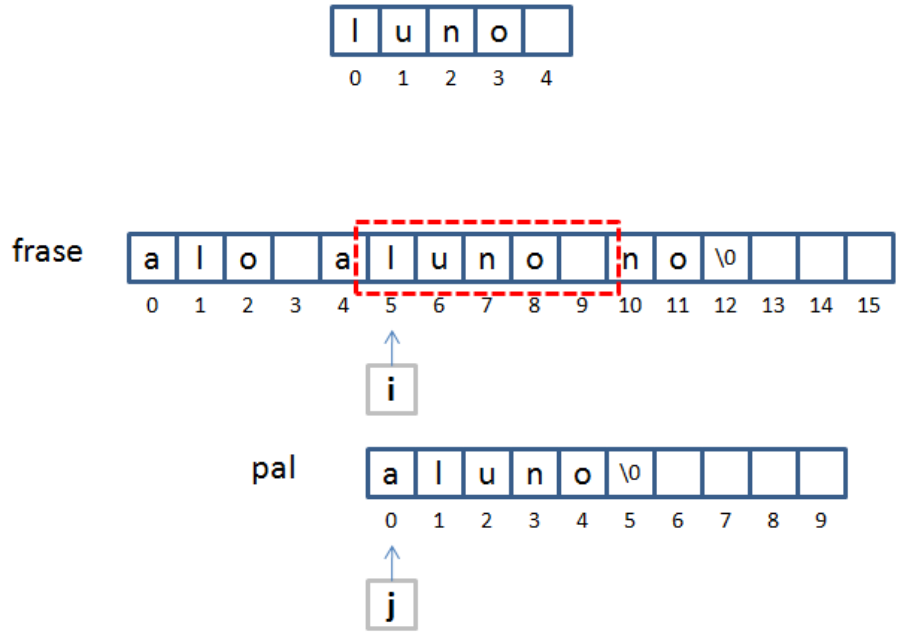
Exercicios XXXIV



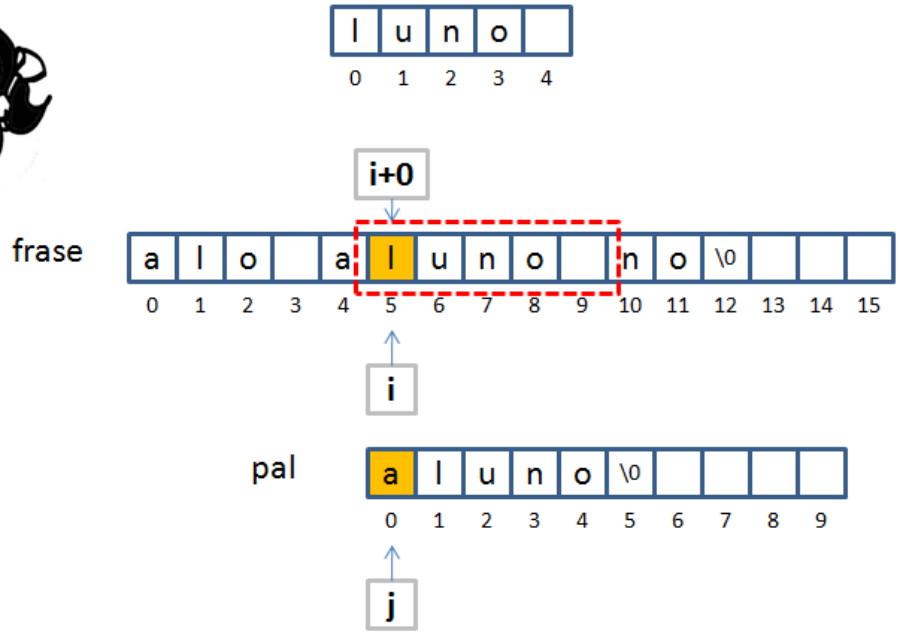
Exercicios XXXV



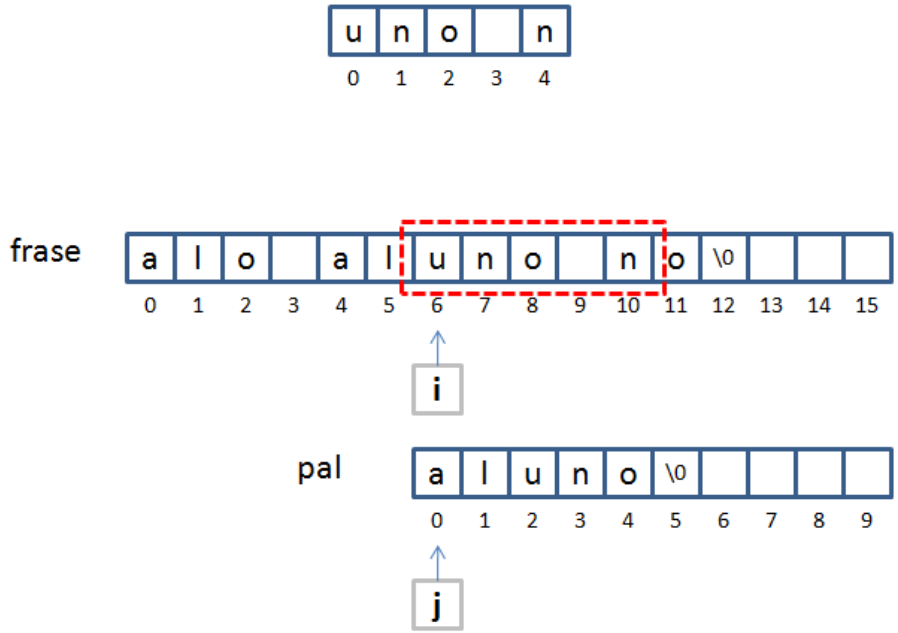
Exercicios XXXVI



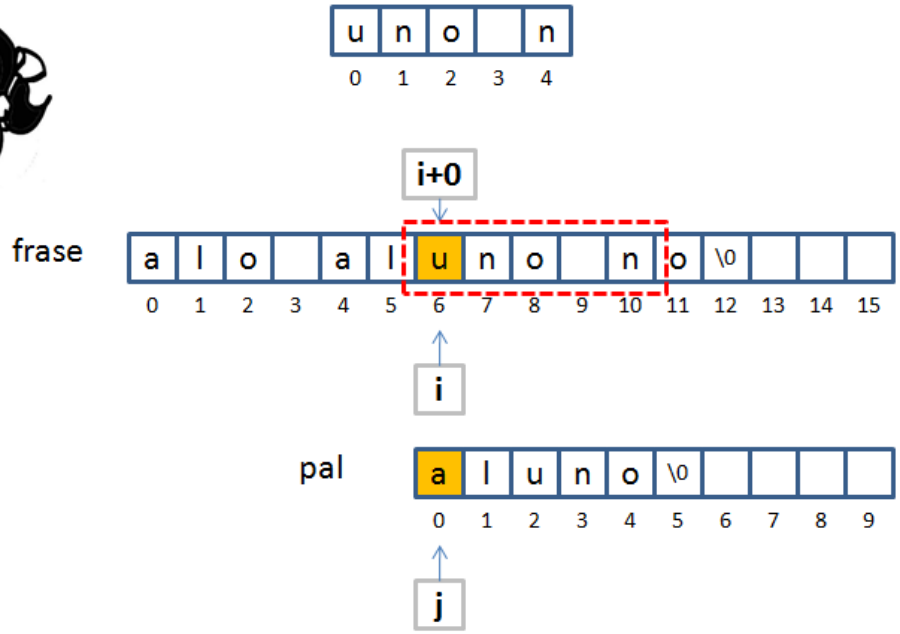
Exercicios XXXVII



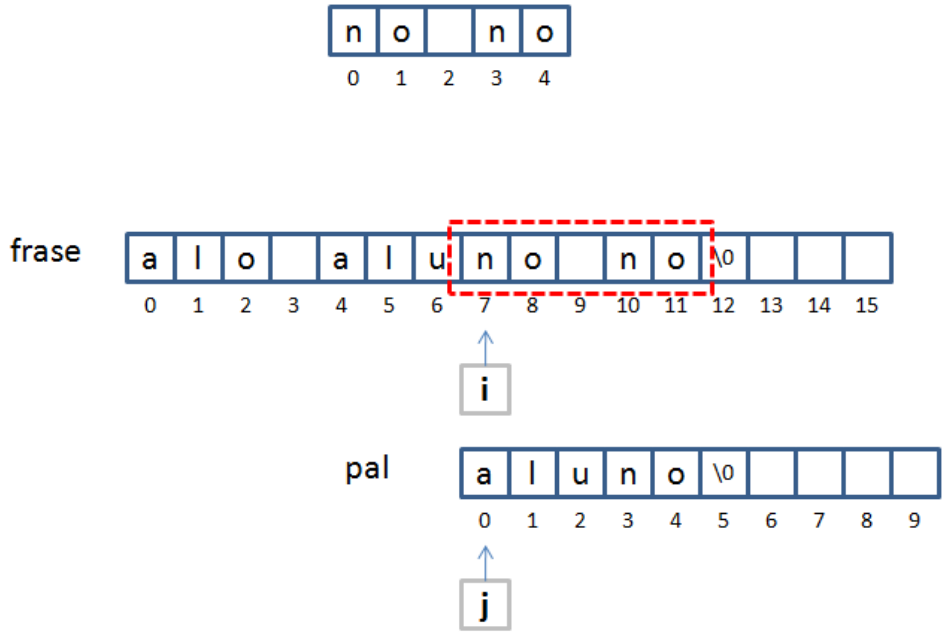
Exercicios XXXVIII



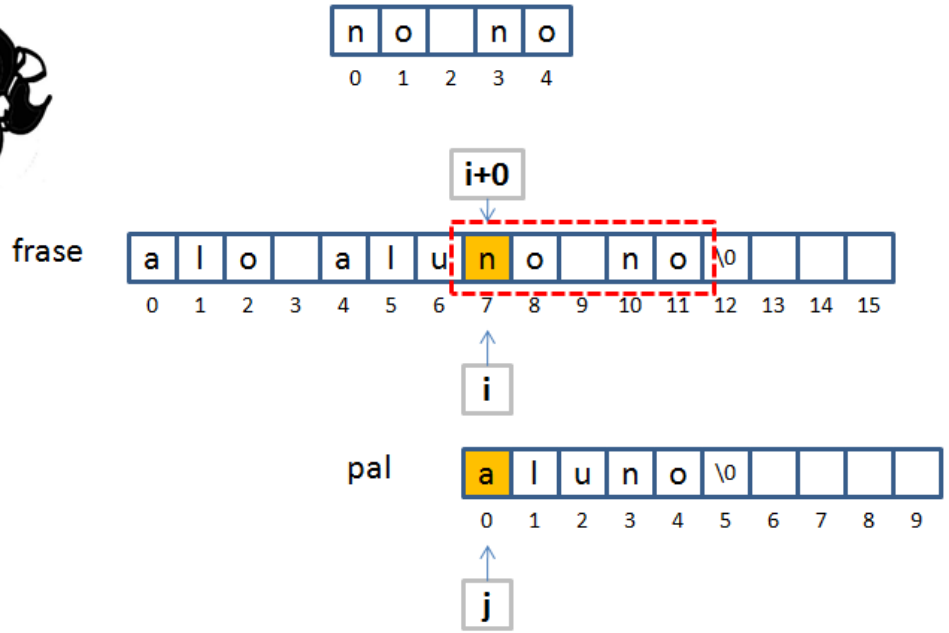
Exercicios XXXIX



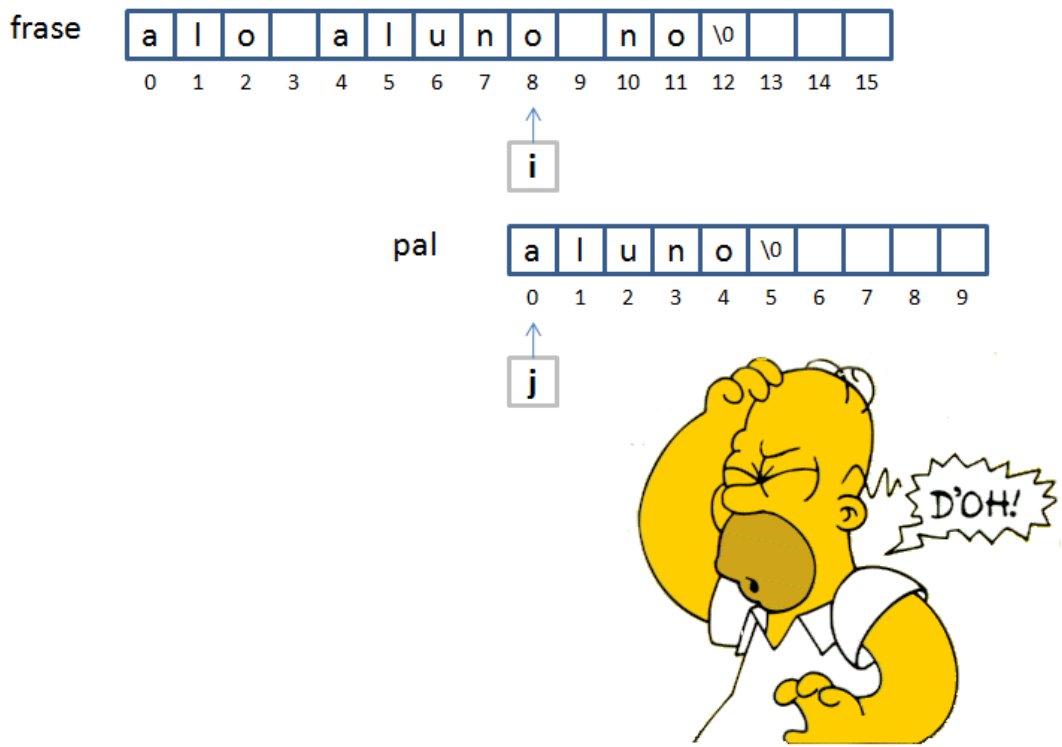
Exercicios XL



Exercicios XLI



Exercicios XLII



Exercicios XLIII

```
bool EstaContida(string frase, string pal);
int main()
{
    string frase, pal;
    cout << "Inserir cadeia e subcadeia: ";
    cin >> cad1 >> cad2;
    if (EstaContida(cad1, cad2))
        cout << "subcadeia contida";
    return 0;
}
```

Exercicios XLIV

```
bool estaContida(string frase, string pal)
{
    int i, j, cont;
    for (i = 0; i < frase.size() - pal.size(); i++)
    {
        for (j = 0; j < pal.size(); j++)
        {
            if (frase[i + j] != pal[j]) break;
        }
        if (j == pal.size())
            return true;
    }
    return false;
}
```

Instrução for com base em intervalo I

Para realizar operações com cada um dos caracteres em um *string*, a forma mais segura de realizar é através do uso de um comando introduzido no novo padrão: comando for com base em intervalo

```
for (declaracao : expressao)
    comando
```

onde `expressao` é um objeto que representa uma sequência e `declaracao` define uma variável que será usada para acessar a sequência

Instrução for com base em intervalo II

Imprimir cada caracter usando o comando for com base em intervalo

```
string str("alguma cadeia");

for (auto c : str) // para cada char em str
    cout << c << endl; // imprime o char corrente
```

Instrução for com base em intervalo III

Contar o número de caracteres de pontuação em uma *string*

```
#include<iostream>
#include<cctype>
#include<string>
using namespace std;
int main(){
    string s("Bom dia!!!!");
    int count = 0;
    for (auto ch : s)
    {
        if (ispunct(ch))
            count++;
    }
    cout << count;
    return 0;
}
```

Instrução for com base em intervalo IV

Usando for com base em intervalo para modificar os valores de uma *string*. Para pode alterar os valores do caracteres, deve-se definir uma variável que acesse os dados por referencia (&).

Instrução for com base em intervalo V

Converter os caracteres de uma *string* em maiúsculos

```
#include<iostream>
#include<cctype>
#include<string>
using namespace std;
int main(){
    string s("Bom dia!!!!");
    int count = 0;
    for (auto &ch : s)
    {
        ch = toupper(ch);
    }
    cout << s;
    return 0;
}
```

FIM