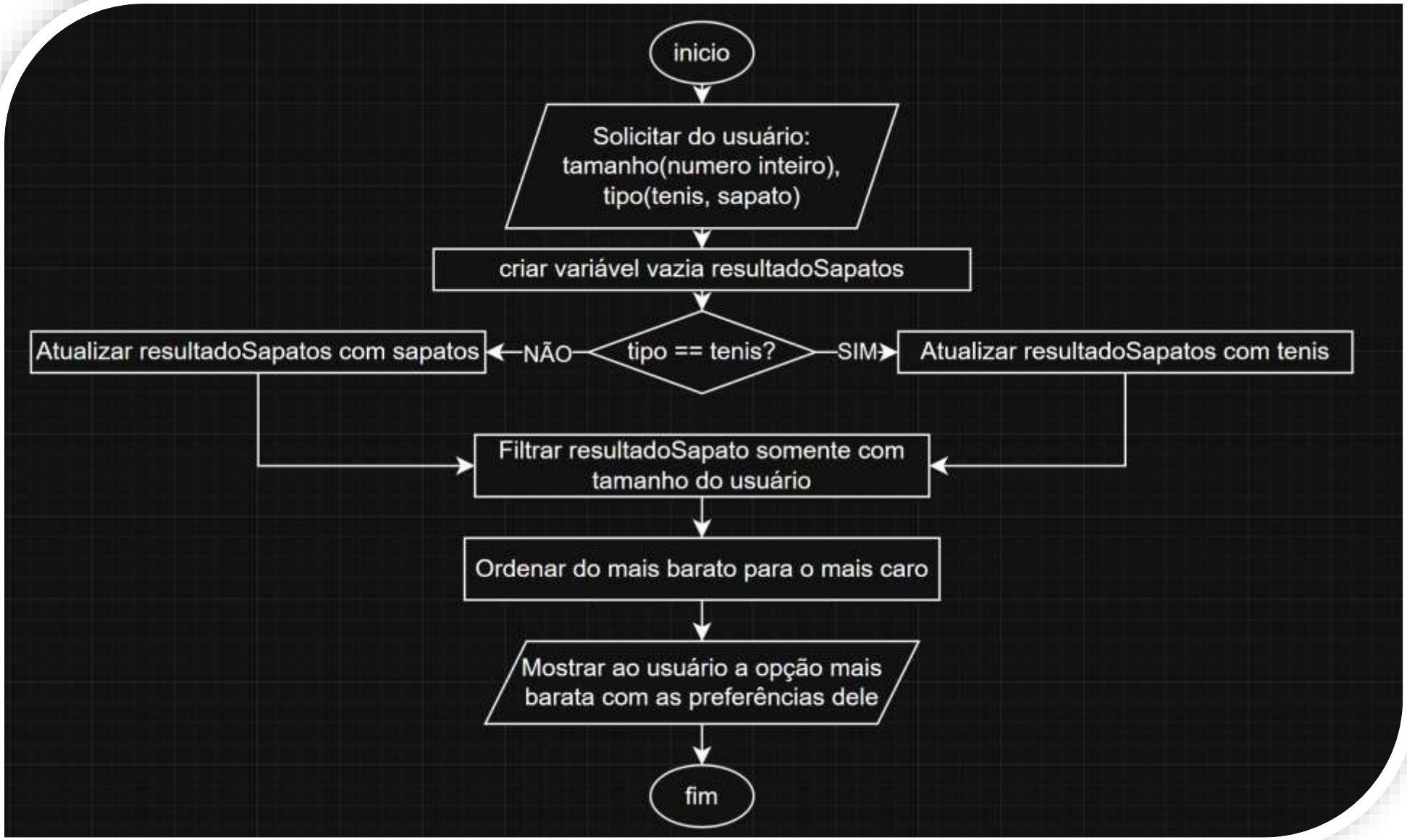
@fpftech.educacional

FPFtech
Escola Tecnológica

FPFtech
Escola Tecnológica

Algoritmo sapato barato



Busca palavra em texto

```
1   programa {
2     inclua biblioteca Texto --> tx
3     funcao inicio() {
4       inteiro palavrachave
5       cadeia buscatexto
6       cadeia textobase = "Lorem ipsum dolor sit amet, consectetur adipiscing elit, s
7
8       escreva("Digite uma palavra a ser procurada\n")
9       leia(buscatexto)
10      palavrachave = tx.posicao_texto(buscatexto, textobase, 0)
11      se (palavrachave==-1) {
12        escreva("Sua palavra não foi encontrada")
13      } senao {
14        escreva("A palavra: ", buscatexto, " começa no índice: ", palavrachave)
15      }
16    }
17  }
18
```

```
Digite uma palavra a ser procurada
sit
A palavra: sit começa no índice: 18
Programa finalizado. Tempo de execução: 16847 milissegundos
```

- A palavra a ser buscada
- O texto onde buscar a palavra, nesse caso, foi o parágrafo do Lorem ipsum.
- O índice do começo, nesse caso, numero 0 significa que começará a procurar desde o início do texto.

- Quando ele encontra a palavra da primeira vez, ele retorna o índice (tipo inteiro) do primeiro caractere da palavra.
- Se não encontrar, ele retorna o índice -1.

Algoritmo sapato barato

```
programa {
  inclua biblioteca Texto --> tx
  inclua biblioteca Tipos --> tp
  funcao inicio() {
    inteiro tamanho, i
    cadeia tipo, tam_string
    caracter genero
    logico achouSapato = falso
    cadeia estoqueSapatos[5] = {
      "40 - M - Tênis - 200",
      "38 - F - Sapato - 150",
      "40 - M - Sapato - 180",
      "36 - F - Tênis - 120",
      "38 - M - Tênis - 130"
    }
    escreva("Qual o tamanho do calçado?\n")
    leia(tamanho)
    escreva("Qual o gênero?\nM ou F\n")
    leia(genero)
    escreva("Qual o tipo?\nTênis ou Sapato\n")
    leia(tipo)

    tam_string = tp.inteiro_para_cadeia(tamanho, 10)
    limpa()
    para (i=0; i<5; i++) {
      se (tx.posicao_texto(tam_string, estoqueSapatos[i], 0)!=-1 e
      tx.posicao_texto(genero, estoqueSapatos[i], 0)!=-1 e
      tx.posicao_texto(tipo, estoqueSapatos[i], 0)!=-1
      ) {
        escreva("Temos o calçado ", estoqueSapatos[i])
        achouSapato = verdadeiro
        pare
      }
    }
    se (achouSapato == falso) {
      escreva("Não achamos o calçado.")
    }
  }
}
```

- Utilizamos bibliotecas Texto, Tipos

- Da biblioteca Tipo precisamos da função “inteiro_para_cadeia”
- A função inteiro_para_cadeia precisa de 2 argumentos:
- 1 – numero inteiro
- 2 – em que base de numero converter (base10, base2, base16, etc)

- Da biblioteca Textos precisamos da função “posicao_texto”
- Posicao_texto precisa de 3 argumentos:
- 1 – palavra a se procurar
- 2 – o texto origem (onde procurar)
- 3 – posição de onde começar (índice)
- Lembre-se que ela retorna um numero(índice), e checamos se é diferente de -1

Algoritmo sapato barato

```
programa {
  inclua biblioteca Texto --> tx
  inclua biblioteca Tipos --> tp
  funcao inicio() {
    inteiro tamanho, i
    cadeia tipo, tam_string
    caracter genero
    logico achouSapato = falso
    cadeia estoqueSapatos[5] = {
      "40 - M - Tênis - 200",
      "38 - F - Sapato - 150",
      "40 - M - Sapato - 180",
      "36 - F - Tênis - 120",
      "38 - M - Tênis - 130"
    }
    escreva("Qual o tamanho do calçado?\n")
    leia(tamanho)
    escreva("Qual o gênero?\nM ou F\n")
    leia(genero)
    escreva("Qual o tipo?\nTênis ou Sapato\n")
    leia(tipo)

    tam_string = tp.inteiro_para_cadeia(tamanho, 10)
    limpa()
    para (i=0; i<5; i++) {
      se (tx.posicao_texto(tam_string, estoqueSapatos[i], 0)!=-1 e
      tx.posicao_texto(genero, estoqueSapatos[i], 0)!=-1 e
      tx.posicao_texto(tipo, estoqueSapatos[i], 0)!=-1
      ) {
        escreva("Temos o calçado ", estoqueSapatos[i])
        achouSapato = verdadeiro
        pare
      }
    }
    se (achouSapato == falso) {
      escreva("Não achamos o calçado.")
    }
  }
}
```

Se achou sapato, troca o valor do booleano "achouSapato" para verdadeiro.
Depois checamos se é verdadeiro ou falso...
Se for falso, não achou o sapato, escreva "Não achamos o calçado"

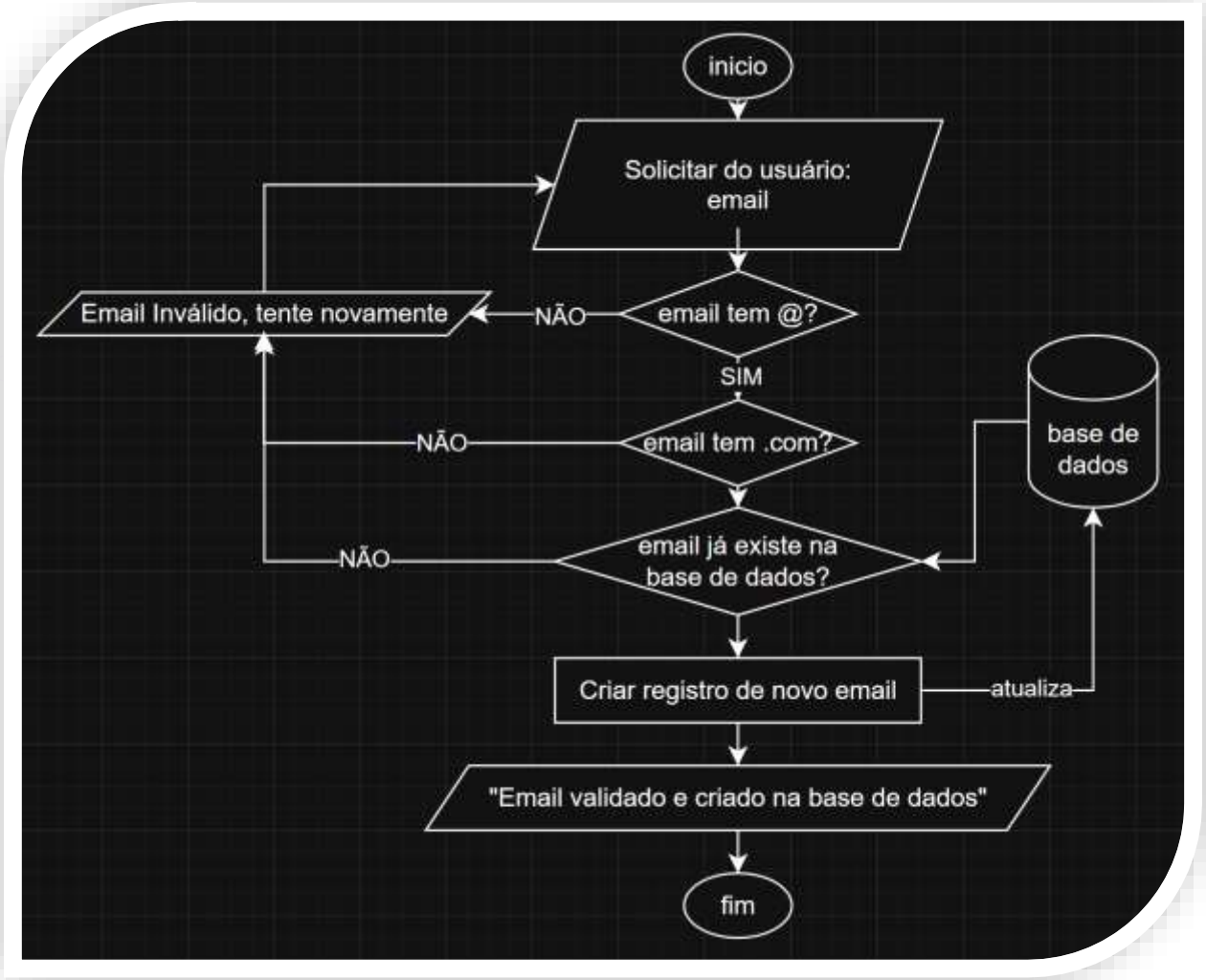
- Juntamente com a função tx.posição_texto() colocamos um operador lógico (e):
- Lembre-se da tabela verdade:

0(ñ passou)**false**	e	0(ñ passou)**false**	=	0(ñ passou)**false**
0(ñ passou)**false**	e	1(passou)**true**	=	0(ñ passou)**false**
1(passou)**true**	e	0(ñ passou)**false**	=	0(ñ passou)**false**
1(passou)**true**	e	1(passou)**true**	=	1(passou)**true**

0(ñ passou)**false**	ou	0(ñ passou)**false**	=	0(ñ passou)**false**
0(ñ passou)**false**	ou	1(passou)**true**	=	1(passou)**true**
1(passou)**true**	ou	0(ñ passou)**false**	=	1(passou)**true**
1(passou)**true**	ou	1(passou)**true**	=	1(passou)**true**

- Ou seja, para cada sapato da lista, ele procura pelo tamanho e pelo gênero e pelo tipo...
- Só vai escrever "Temos o calçado" se todos esses 3 forem encontrados em 1 sapato.

Algoritmo validar email



Algoritmo validar email

```
programa {
  inclua biblioteca Texto --> tx
  inclua biblioteca Util --> ut
  funcao inicio() {
    cadeia email
    inteiro i
    cadeia database[] = {
      "julian123@gmail.com",
      "aluno20@gmail.com",
      "alguem@hotmail.com",
      "estudante@fpftech.com",
      "portugol@yahoo.com",
      ""
    }
    enquanto (email != "sair") {
      escreva("Digite seu email OU digite a palavra: 'sair'\n")
      leia(email)

      se (tx.posicao_texto("@", email, 5)!=-1 e tx.posicao_texto(".com", email, 9)!=-1) {
        para (i=0; i<ut.numero_elementos(database); i++) {
          se (database[i]!=email e database[i]=="") {
            database[i] = email
            escreva("email salvo\n")
            email = "sair"
            pare
          } senao se (database[i]==email) {
            escreva("Email já existe!\nTente novamente.\n")
            pare
          }
        }
      } senao se (email!="sair") {
        escreva("Email inválido. Tente novamente\n")
      }
    }
  }
}
```

Lembre-se: separado por vírgulas, que estão fora das aspas

- Utilizamos bibliotecas Texto, Util

- Criamos uma variável do tipo cadeia(string) chamada “email”
- Criamos uma variável do tipo inteiro(numero) chamada “i” (esse será nosso índice para o (for loop)
- Criamos um vetor (lista) de elementos do tipo cadeia (string) e chamamos essa lista de database, já colocamos valores iniciais na nossa database.

Algoritmo validar email

```
programa {
  inclua biblioteca Texto --> tx
  inclua biblioteca Util --> ut
  funcao inicio() {
    cadeia email
    inteiro i
    cadeia database[] = {
      "julian123@gmail.com",
      "aluno20@gmail.com",
      "alguem@hotmail.com",
      "estudante@fpftech.com",
      "portugol@yahoo.com",
      ""
    }
    enquanto (email != "sair") {
      escreva("Digite seu email OU digite a palavra: 'sair'\n")
      leia(email)

      se (tx.posicao_texto("@", email, 5)!=-1 e tx.posicao_texto(".com", email, 9)!=-1) {
        para (i=0; i<ut.numero_elementos(database); i++) {
          se (database[i]!=email e database[i]=="") {
            database[i] = email
            escreva("email salvo\n")
            email = "sair"
            pare
          } senao se (database[i]==email) {
            escreva("Email já existe!\nTente novamente.\n")
            pare
          }
        }
      } senao se (email!="sair") {
        escreva("Email inválido. Tente novamente\n")
      }
    }
  }
}
```

- Enquanto (while loop) é uma repetição que até que a condição não seja contrariada...
- Ele vai ficar preso no loop (dentro do quadrado laranja).
- No nosso caso, a condição é: email diferente de “sair”.
- Ou seja, o programa vai ficar preso no loop rodando a interação até que o valor da variável email vire “sair”. (email==“sair”)

Algoritmo validar email

```
programa {
  inclua biblioteca Texto --> tx
  inclua biblioteca Util --> ut
  funcao inicio() {
    cadeia email
    inteiro i
    cadeia database[] = {
      "julian123@gmail.com",
      "aluno20@gmail.com",
      "alguem@hotmail.com",
      "estudante@fpftech.com",
      "portugol@yahoo.com",
      ""
    }
    enquanto (email != "sair") {
      escreva("Digite seu email OU digite a palavra: 'sair'\n")
      leia(email)

      se (tx.posicao_texto("@", email, 5)!=-1 e tx.posicao_texto(".com", email, 9)!=-1) {
        para (i=0; i<ut.numero_elementos(database); i++) {
          se (database[i]!=email e database[i]=="") {
            database[i] = email
            escreva("email salvo\n")
            email = "sair"
            pare
          } senao se (database[i]==email) {
            escreva("Email já existe!\nTente novamente.\n")
            pare
          }
        }
      } senao se (email!="sair") {
        escreva("Email inválido. Tente novamente\n")
      }
    }
  }
}
```

- Checa se:
- tem “@” no email a partir do índice 5 ?
- E (tabela verdade)
- tem “.com” no email a partir do índice 9 ?

- Se passou dos checks acima, fazer (For loop)
- começa do i=0;
- i<tamanho_vetor_database;
- i++(de 1_em_1)

- Para saber a quantidade de elementos do vetor “database”, utilizamos uma biblioteca chamada Utils e a função numero_elementos
- Essa função recebe apenas 1 argumento que é o próprio vetor.
- Ele conta e retorna o numero de elementos como tipo inteiro. (=6)
- Utilizamos isso para colocar no nosso (for loop) como condição de término.

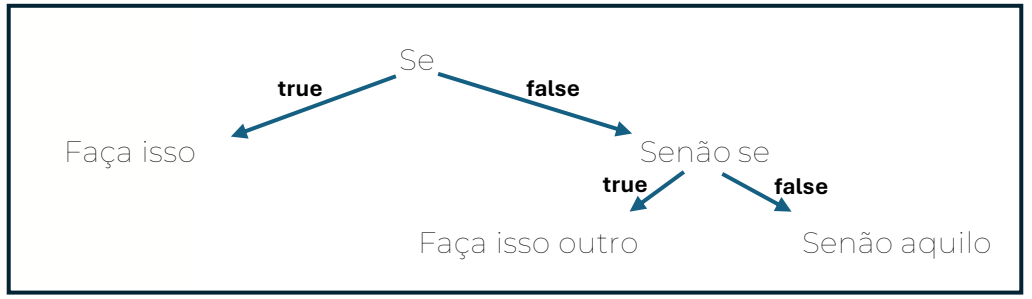
Algoritmo validar email

```
programa {
  inclua biblioteca Texto --> tx
  inclua biblioteca Util --> ut
  funcao inicio() {
    cadeia email
    inteiro i
    cadeia database[] = {
      "julian123@gmail.com",
      "aluno20@gmail.com",
      "alguem@hotmail.com",
      "estudante@fpftech.com",
      "portugol@yahoo.com",
      ""
    }
    enquanto (email != "sair") {
      escreva("Digite seu email OU digite a palavra: 'sair'\n")
      leia(email)

      se (tx.posicao_texto("@", email, 5)!=-1 e tx.posicao_texto(".com", email, 9)!=-1) {
        para (i=0; i<ut.numero_elementos(database); i++) {
          se (database[i]!=email e database[i]=="") {
            database[i] = email
            escreva("email salvo\n")
            email = "sair"
            pare
          } senao se (database[i]==email) {
            escreva("Email já existe!\nTente novamente.\n")
            pare
          }
        }
      } senao se (email!="sair") {
        escreva("Email inválido. Tente novamente\n")
      }
    }
  }
}
```

- Para cada um email da database (database[i]) ele verifica se o email novo é diferente e se tem um espaço vago na base de dados (==“”).
- Se essas duas condições passarem, então salve o novo email na base de dados
- Depois de salvar o novo email, o sistema muda o valor da variável email para “sair”.
- Pare faz com que se saia do (for loop).

- Senão se já tem o novo email digitado na database:
- O sistema escreve “Email já existente”
- Pare faz com que se saia do (for loop).



Algoritmo validar email

```
programa {
  inclua biblioteca Texto --> tx
  inclua biblioteca Util --> ut
  funcao inicio() {
    cadeia email
    inteiro i
    cadeia database[] = {
      "julian123@gmail.com",
      "aluno20@gmail.com",
      "alguem@hotmail.com",
      "estudante@fpftech.com",
      "portugol@yahoo.com",
      ""
    }
    enquanto (email != "sair") {
      escreva("Digite seu email OU digite a palavra: 'sair'\n")
      leia(email)

      se (tx.posicao_texto("@", email, 5)!=-1 e tx.posicao_texto(".com", email, 9)!=-1) {
        para (i=0; i<ut.numero_elementos(database); i++) {
          se (database[i]!=email e database[i]=="") {
            database[i] = email
            escreva("email salvo\n")
            email = "sair"
            pare
          } senao se (database[i]==email) {
            escreva("Email já existe!\nTente novamente.\n")
            pare
          }
        }
      } senao se (email!="sair") {
        escreva("Email inválido. Tente novamente\n")
      }
    }
  }
}
```

- Ao passar essas checagens, ele faz um último check antes de voltar pro início do enquanto.
- Checa se email for diferente de “sair”
- Se for diferente, o sistema escreve “Email inválido, tente novamente”
- E retorna o laço do enquanto (While loop).

- Fica preso no (While loop), até que o email vire “sair” o programa sempre vai estar perguntando “Digite seu email ou a palavra ‘sair’, sempre vai ficar lendo o input do usuário e guardando na variável email. E sempre fazendo as checagens e processamentos que vimos anteriormente.

FPFtech
Escola Tecnológica

Obrigado!

FPFtech
Escola Tecnológica

@fpftech.educacional