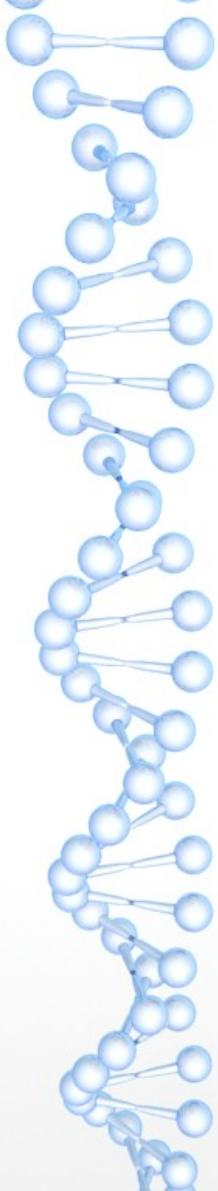
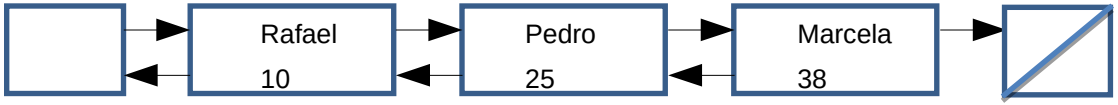


Estrutura de Dados - I

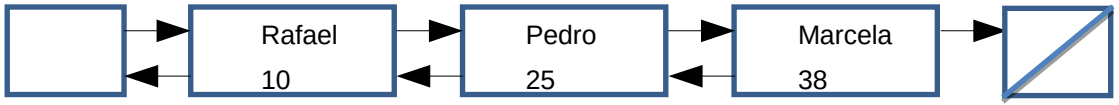
Listas Ligadas

Prof. MSc. Rafael Staiger Bressan
rafael.bressan@unicesumar.edu.br

Lista Duplamente Encadeada

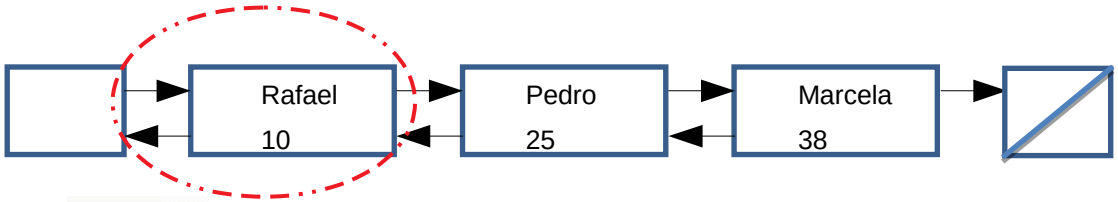


Lista Duplamente Encadeada
Remove (Idade)



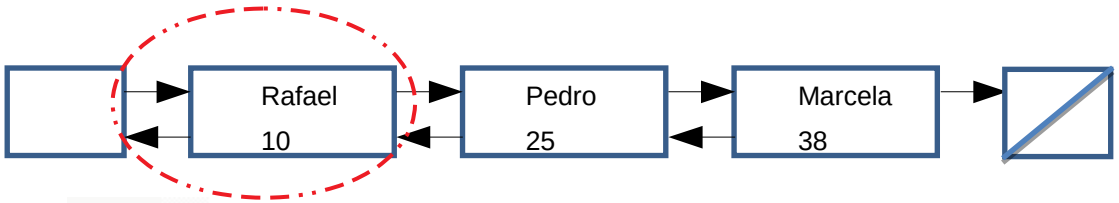

```
68    // Remove Idade
69  void removeIdade(int idade, celula *ini) {
70      if (ini->prox != NULL){
71          if(idade == ini->prox->idade){
72              celula *lixo;
73              lixo = ini->prox;
74              if (ini->prox->prox != NULL){
75                  ini->prox->prox->ant = ini;
76                  ini->prox = ini->prox->prox;
77              }else{
78                  ini->prox = NULL;
79              }
80              free(lixo);
81          }else{
82              removeIdade(idade, ini->prox);
83          }
84      }
85  }
```

Lista Duplamente Encadeada
Remove (Idade)



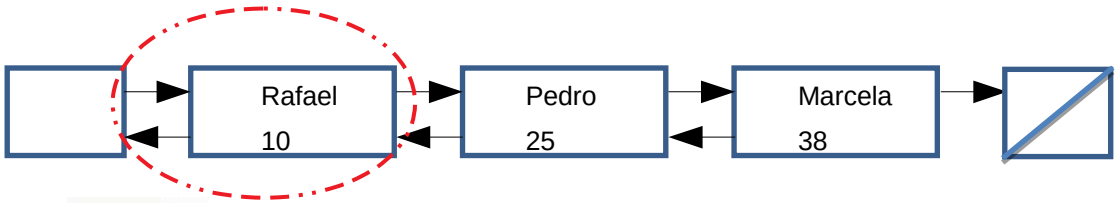

```
68    // Remove Idade
69    void removeIdade(int idade, celula *ini) {
70        if (ini->prox != NULL){
71            if(idade == ini->prox->idade){
72                celula *lixo;
73                lixo = ini->prox;
74                if (ini->prox->prox != NULL){
75                    ini->prox->prox->ant = ini;
76                    ini->prox = ini->prox->prox;
77                }else{
78                    ini->prox = NULL;
79                }
80                free(lixo);
81            }else{
82                removeIdade(idade, ini->prox);
83            }
84        }
85    }
```

Lista Duplamente Encadeada
Remove (Idade)




```
68    // Remove Idade
69  void removeIdade(int idade, celula *ini) {
70      if (ini->prox != NULL){
71          if(idade == ini->prox->idade){
72              celula *lixo;
73              lixo = ini->prox;
74              if (ini->prox->prox != NULL){
75                  ini->prox->prox->ant = ini;
76                  ini->prox = ini->prox->prox;
77              }else{
78                  ini->prox = NULL;
79              }
80              free(lixo);
81          }else{
82              removeIdade(idade, ini->prox);
83          }
84      }
85  }
```

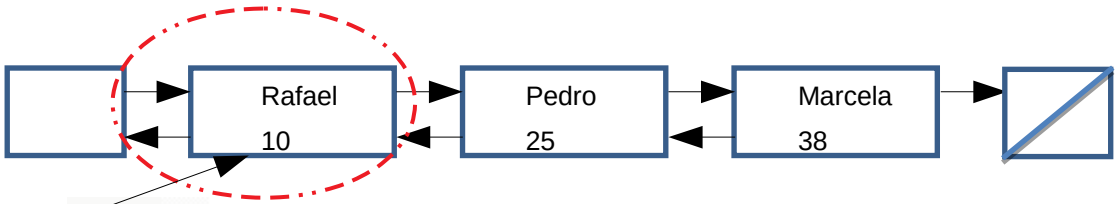
Lista Duplamente Encadeada
Remove (Idade)



LIXO

```
68    // Remove Idade
69  void removeIdade(int idade, celula *ini) {
70      if (ini->prox != NULL){
71          if(idade == ini->prox->idade){
72              celula *lixo;
73              lixo = ini->prox;
74              if (ini->prox->prox != NULL){
75                  ini->prox->prox->ant = ini;
76                  ini->prox = ini->prox->prox;
77              }else{
78                  ini->prox = NULL;
79              }
80              free(lixo);
81          }else{
82              removeIdade(idade, ini->prox);
83          }
84      }
85  }
```

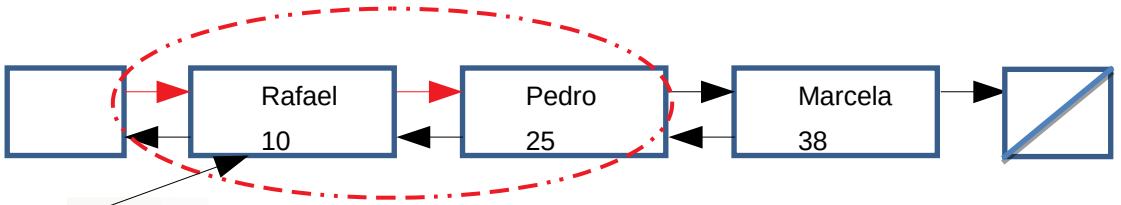
Lista Duplamente Encadeada
Remove (Idade)



LIXO

```
68    // Remove Idade
69    void removeIdade(int idade, celula *ini) {
70        if (ini->prox != NULL){
71            if(idade == ini->prox->idade){
72                celula *lixo;
73                lixo = ini->prox;
74                if (ini->prox->prox != NULL){
75                    ini->prox->prox->ant = ini;
76                    ini->prox = ini->prox->prox;
77                }else{
78                    ini->prox = NULL;
79                }
80                free(lixo);
81            }else{
82                removeIdade(idade, ini->prox);
83            }
84        }
85    }
```

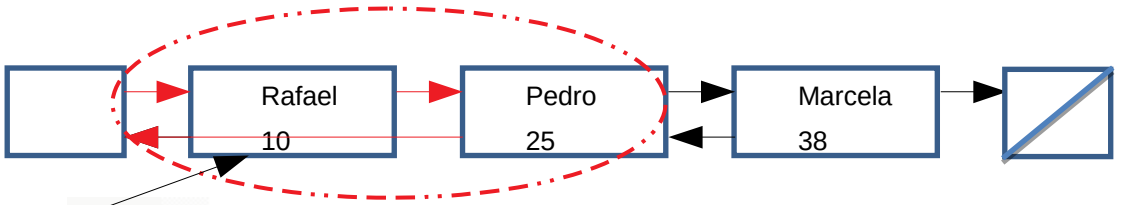
Lista Duplamente Encadeada
Remove (Idade)



LIXO

```
68    // Remove Idade
69    void removeIdade(int idade, celula *ini) {
70        if (ini->prox != NULL){
71            if(idade == ini->prox->idade){
72                celula *lixo;
73                lixo = ini->prox;
74                if (ini->prox->prox != NULL){
75                    ini->prox->prox->ant = ini;
76                    ini->prox = ini->prox->prox;
77                }else{
78                    ini->prox = NULL;
79                }
80                free(lixo);
81            }else{
82                removeIdade(idade, ini->prox);
83            }
84        }
85    }
```

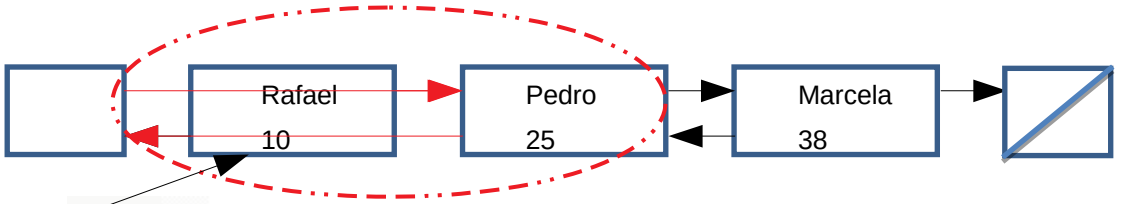
Lista Duplamente Encadeada
Remove (Idade)



LIXO

```
68    // Remove Idade
69    void removeIdade(int idade, celula *ini) {
70        if (ini->prox != NULL){
71            if(idade == ini->prox->idade){
72                celula *lixo;
73                lixo = ini->prox;
74                if (ini->prox->prox != NULL){
75                    ini->prox->prox->ant = ini;
76                    ini->prox = ini->prox->prox;
77                }else{
78                    ini->prox = NULL;
79                }
80                free(lixo);
81            }else{
82                removeIdade(idade, ini->prox);
83            }
84        }
85    }
```

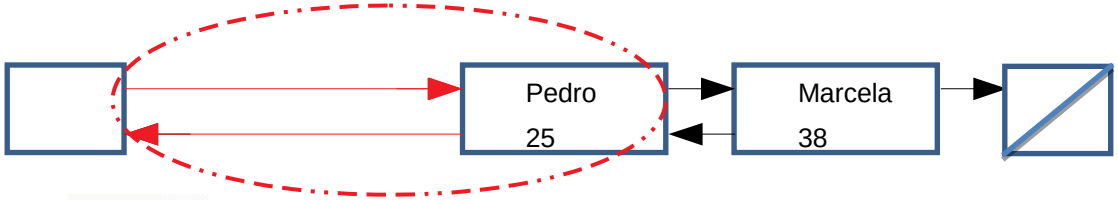
Lista Duplamente Encadeada
Remove (Idade)



LIXO

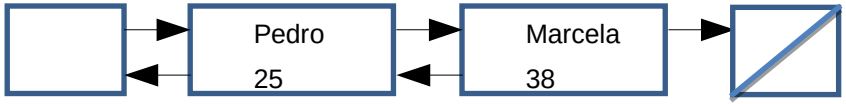
```
68    // Remove Idade
69  void removeIdade(int idade, celula *ini) {
70      if (ini->prox != NULL){
71          if(idade == ini->prox->idade){
72              celula *lixo;
73              lixo = ini->prox;
74              if (ini->prox->prox != NULL){
75                  ini->prox->prox->ant = ini;
76                  ini->prox = ini->prox->prox;
77              }else{
78                  ini->prox = NULL;
79              }
80              free(lixo);
81          }else{
82              removeIdade(idade, ini->prox);
83          }
84      }
85  }
```

Lista Duplamente Encadeada
Remove (Idade)



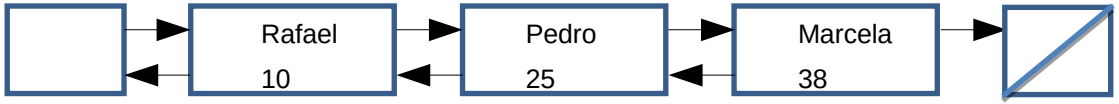

```
68    // Remove Idade
69  void removeIdade(int idade, celula *ini) {
70      if (ini->prox != NULL){
71          if(idade == ini->prox->idade){
72              celula *lixo;
73              lixo = ini->prox;
74              if (ini->prox->prox != NULL){
75                  ini->prox->prox->ant = ini;
76                  ini->prox = ini->prox->prox;
77              }else{
78                  ini->prox = NULL;
79              }
80              free(lixo);
81          }else{
82              removeIdade(idade, ini->prox);
83          }
84      }
85  }
```

Lista Duplamente Encadeada
Remove (Idade)



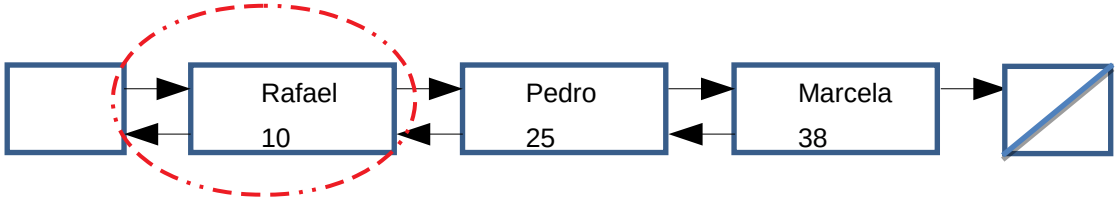

```
68    // Remove Idade
69    void removeIdade(int idade, celula *ini) {
70        if (ini->prox != NULL){
71            if(idade == ini->prox->idade){
72                celula *lixo;
73                lixo = ini->prox;
74                if (ini->prox->prox != NULL){
75                    ini->prox->prox->ant = ini;
76                    ini->prox = ini->prox->prox;
77                }else{
78                    ini->prox = NULL;
79                }
80                free(lixo);
81            }else{
82                removeIdade(idade, ini->prox);
83            }
84        }
85    }
```

Lista Duplamente Encadeada
Remove (Nome)



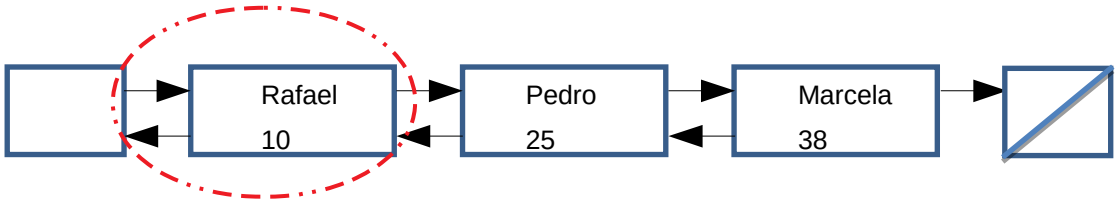

```
87    // Remove nome
88    void removeNome(char nome[], celula *ini) {
89        if (ini->prox != NULL){
90            if (strcmp(nome, ini->prox->nome) == 0){
91                celula *lixo;
92                lixo = ini->prox;
93                if (ini->prox->prox != NULL){
94                    ini->prox->prox->ant = ini;
95                    ini->prox = ini->prox->prox;
96                }else{
97                    ini->prox = NULL;
98                }
99                free(lixo);
100           }else{
101               removeNome(nome, ini->prox);
102           }
103       }
104   }
```

Lista Duplamente Encadeada
Remove (Nome)



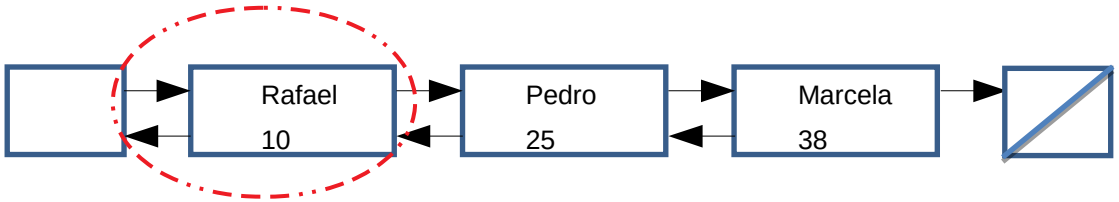

```
87    // Remove nome
88    void removeNome(char nome[], celula *ini) {
89        if (ini->prox != NULL){
90            if (strcmp(nome, ini->prox->nome) == 0){
91                celula *lixo;
92                lixo = ini->prox;
93                if (ini->prox->prox != NULL){
94                    ini->prox->prox->ant = ini;
95                    ini->prox = ini->prox->prox;
96                }else{
97                    ini->prox = NULL;
98                }
99                free(lixo);
100           }else{
101               removeNome(nome, ini->prox);
102           }
103       }
104   }
```

Lista Duplamente Encadeada
Remove (Nome)



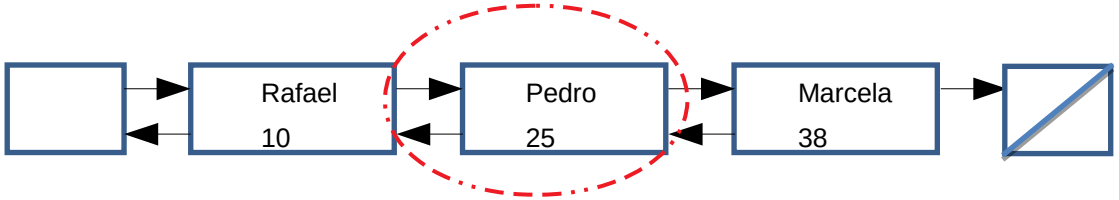

```
87    // Remove nome
88  void removeNome(char nome[], celula *ini) {
89      if (ini->prox != NULL){
90          if (strcmp(nome, ini->prox->nome) == 0){
91              celula *lixo;
92              lixo = ini->prox;
93              if (ini->prox->prox != NULL){
94                  ini->prox->prox->ant = ini;
95                  ini->prox = ini->prox->prox;
96              }else{
97                  ini->prox = NULL;
98              }
99              free(lixo);
100         }else{
101             removeNome(nome, ini->prox);
102         }
103     }
104 }
```

Lista Duplamente Encadeada
Remove (Nome)



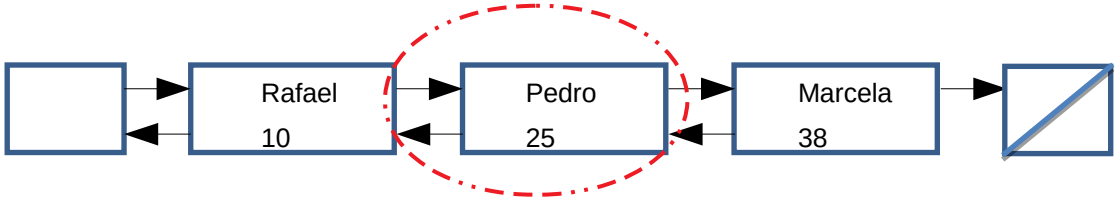

```
87    // Remove nome
88    void removeNome(char nome[], celula *ini) {
89        if (ini->prox != NULL){
90            if (strcmp(nome, ini->prox->nome) == 0){
91                celula *lixo;
92                lixo = ini->prox;
93                if (ini->prox->prox != NULL){
94                    ini->prox->prox->ant = ini;
95                    ini->prox = ini->prox->prox;
96                }else{
97                    ini->prox = NULL;
98                }
99                free(lixo);
100           }else{
101               removeNome(nome, ini->prox);
102           }
103       }
104   }
```

Lista Duplamente Encadeada
Remove (Nome)



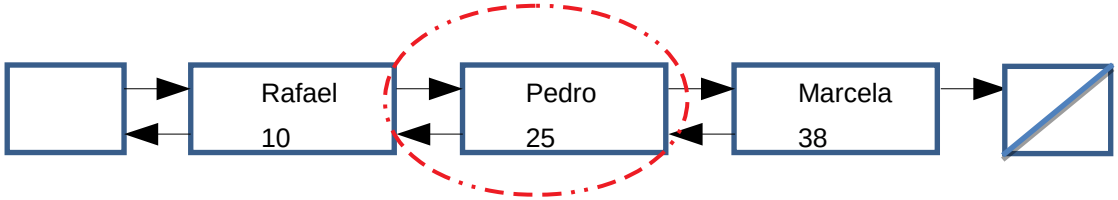

```
87    // Remove nome
88    void removeNome(char nome[], celula *ini) {
89        if (ini->prox != NULL){
90            if (strcmp(nome, ini->prox->nome) == 0){
91                celula *lixo;
92                lixo = ini->prox;
93                if (ini->prox->prox != NULL){
94                    ini->prox->prox->ant = ini;
95                    ini->prox = ini->prox->prox;
96                }else{
97                    ini->prox = NULL;
98                }
99                free(lixo);
100           }else{
101               removeNome(nome, ini->prox);
102           }
103       }
104   }
```

Lista Duplamente Encadeada
Remove (Nome)




```
87    // Remove nome
88    void removeNome(char nome[], celula *ini) {
89        if (ini->prox != NULL){
90            if (strcmp(nome, ini->prox->nome) == 0){
91                celula *lixo;
92                lixo = ini->prox;
93                if (ini->prox->prox != NULL){
94                    ini->prox->prox->ant = ini;
95                    ini->prox = ini->prox->prox;
96                }else{
97                    ini->prox = NULL;
98                }
99                free(lixo);
100           }else{
101               removeNome(nome, ini->prox);
102           }
103       }
104   }
```

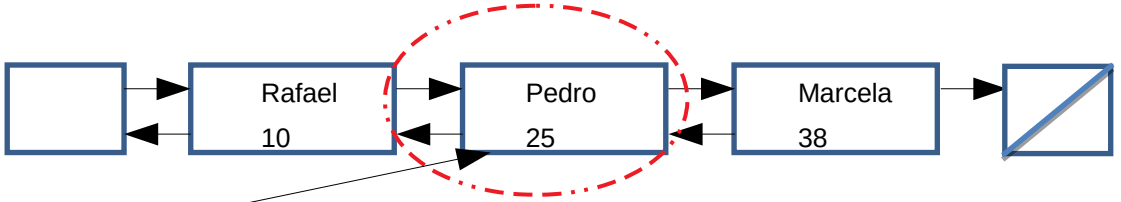
Lista Duplamente Encadeada
Remove (Nome)



LIXO

```
87    // Remove nome
88    void removeNome(char nome[], celula *ini) {
89        if (ini->prox != NULL){
90            if (strcmp(nome, ini->prox->nome) == 0){
91                celula *lixo;
92                lixo = ini->prox;
93                if (ini->prox->prox != NULL){
94                    ini->prox->prox->ant = ini;
95                    ini->prox = ini->prox->prox;
96                }else{
97                    ini->prox = NULL;
98                }
99                free(lixo);
100           }else{
101               removeNome(nome, ini->prox);
102           }
103       }
104   }
```

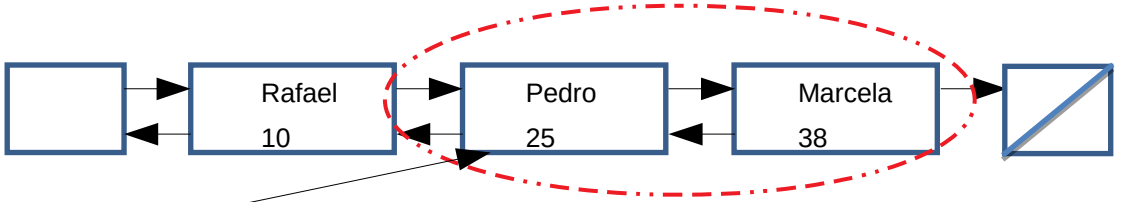
Lista Duplamente Encadeada
Remove (Nome)



LIXO

```
87    // Remove nome
88    void removeNome(char nome[], celula *ini) {
89        if (ini->prox != NULL){
90            if (strcmp(nome, ini->prox->nome) == 0){
91                celula *lixo;
92                lixo = ini->prox;
93                if (ini->prox->prox != NULL){
94                    ini->prox->prox->ant = ini;
95                    ini->prox = ini->prox->prox;
96                }else{
97                    ini->prox = NULL;
98                }
99                free(lixo);
100           }else{
101               removeNome(nome, ini->prox);
102           }
103       }
104   }
```

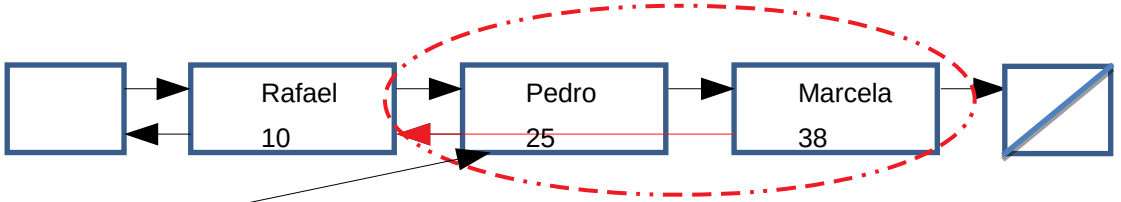
Lista Duplamente Encadeada
Remove (Nome)



LIXO

```
87    // Remove nome
88    void removeNome(char nome[], celula *ini) {
89        if (ini->prox != NULL){
90            if (strcmp(nome, ini->prox->nome) == 0){
91                celula *lixo;
92                lixo = ini->prox;
93                if (ini->prox->prox != NULL){
94                    ini->prox->prox->ant = ini;
95                    ini->prox = ini->prox->prox;
96                }else{
97                    ini->prox = NULL;
98                }
99                free(lixo);
100           }else{
101               removeNome(nome, ini->prox);
102           }
103       }
104   }
```

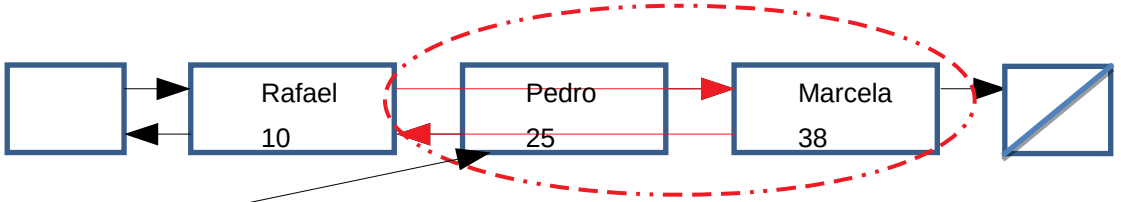
Lista Duplamente Encadeada
Remove (Nome)



LIXO

```
87    // Remove nome
88    void removeNome(char nome[], celula *ini) {
89        if (ini->prox != NULL){
90            if (strcmp(nome, ini->prox->nome) == 0){
91                celula *lixo;
92                lixo = ini->prox;
93                if (ini->prox->prox != NULL){
94                    ini->prox->prox->ant = ini;
95                    ini->prox = ini->prox->prox;
96                }else{
97                    ini->prox = NULL;
98                }
99                free(lixo);
100           }else{
101               removeNome(nome, ini->prox);
102           }
103       }
104   }
```

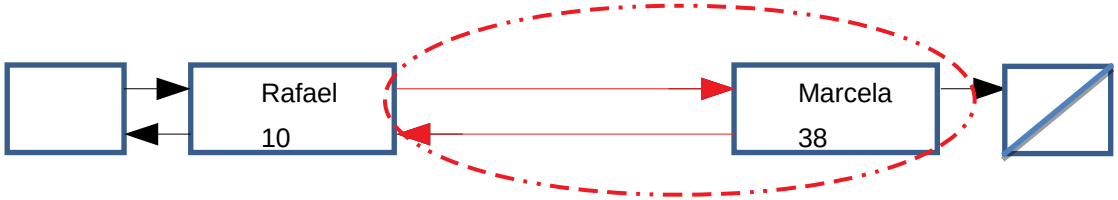
Lista Duplamente Encadeada
Remove (Nome)



LIXO

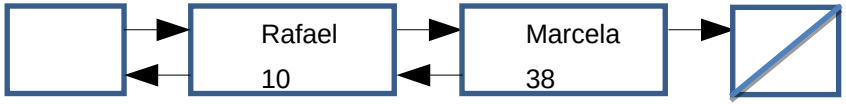
```
87    // Remove nome
88    void removeNome(char nome[], celula *ini) {
89        if (ini->prox != NULL){
90            if (strcmp(nome, ini->prox->nome) == 0){
91                celula *lixo;
92                lixo = ini->prox;
93                if (ini->prox->prox != NULL){
94                    ini->prox->prox->ant = ini;
95                    ini->prox = ini->prox->prox;
96                }else{
97                    ini->prox = NULL;
98                }
99                free(lixo);
100           }else{
101               removeNome(nome, ini->prox);
102           }
103       }
104   }
```

Lista Duplamente Encadeada
Remove (Nome)

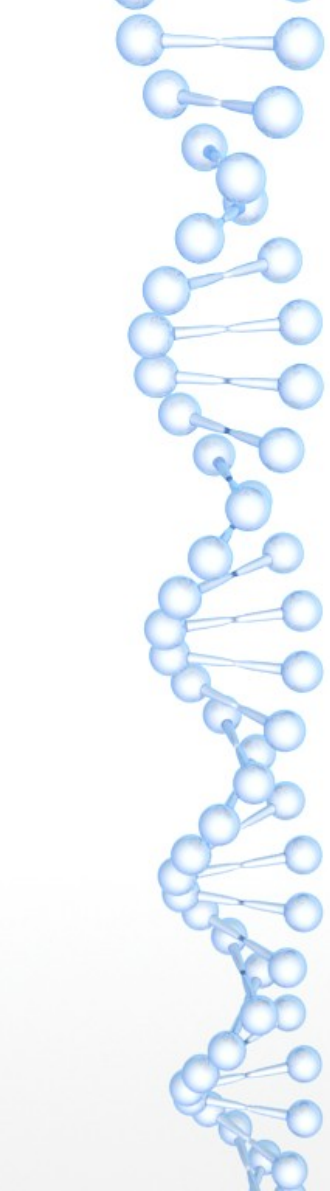



```
87    // Remove nome
88    void removeNome(char nome[], celula *ini) {
89        if (ini->prox != NULL){
90            if (strcmp(nome, ini->prox->nome) == 0){
91                celula *lixo;
92                lixo = ini->prox;
93                if (ini->prox->prox != NULL){
94                    ini->prox->prox->ant = ini;
95                    ini->prox = ini->prox->prox;
96                }else{
97                    ini->prox = NULL;
98                }
99                free(lixo);
100           }else{
101               removeNome(nome, ini->prox);
102           }
103       }
104   }
```

Lista Duplamente Encadeada
Remove (Nome)




```
87    // Remove nome
88    void removeNome(char nome[], celula *ini) {
89        if (ini->prox != NULL){
90            if (strcmp(nome, ini->prox->nome) == 0){
91                celula *lixo;
92                lixo = ini->prox;
93                if (ini->prox->prox != NULL){
94                    ini->prox->prox->ant = ini;
95                    ini->prox = ini->prox->prox;
96                }else{
97                    ini->prox = NULL;
98                }
99                free(lixo);
100           }else{
101               removeNome(nome, ini->prox);
102           }
103       }
104   }
```



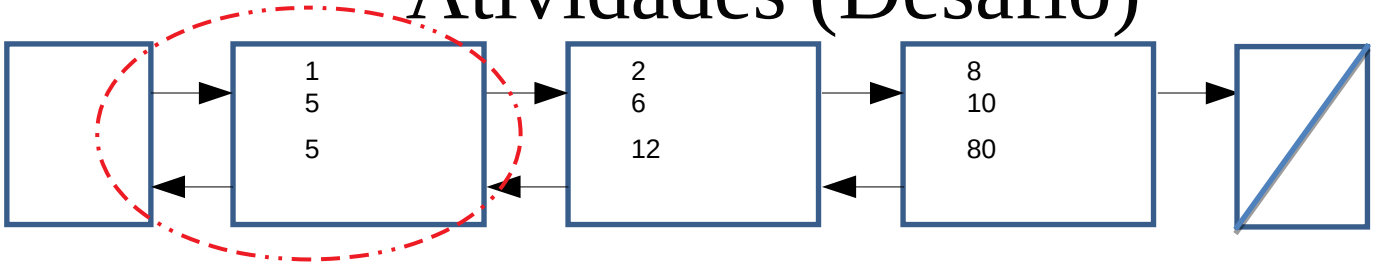
Lista Duplamente Encadeada
Atividades

- Desenvolva uma função que insira de maneira ordenada (menor para maior) a idade em uma lista duplamente encadeada.
- Desenvolva uma função que localiza um nome passado por parâmetro e atualiza os dados da célula.

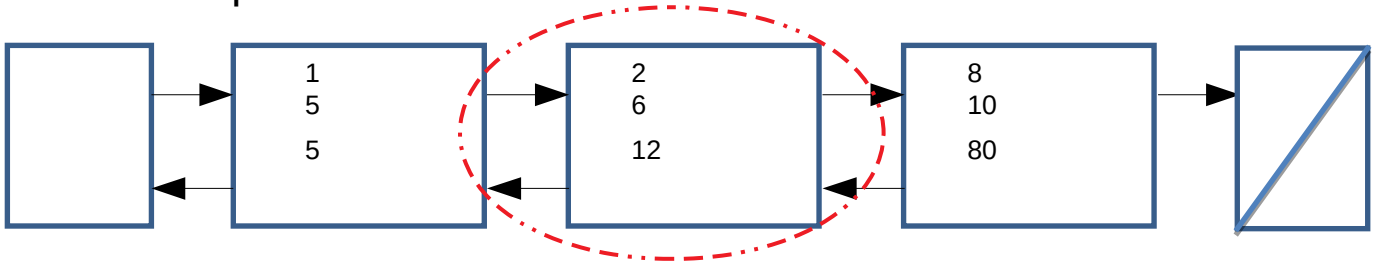
Lista Duplamente Encadeada
Atividades (Desafio)

- Jogo da tabuada;
  - O jogo consiste na criação de uma lista duplamente encadeada que contém em cada célula dois números inteiros (“aleatórios”) e a multiplicação dos números. São iniciadas N células no início do jogo.
  - O jogador inicia a lista e tenta acertar o resultado da multiplicação dos dois números, se ele acertar, a próxima jogada é iniciada, ou seja, passa para a próxima célula. Caso ele erre, ele volta para jogada anterior e uma nova célula (jogada) é criada no final da lista. Cada jogada representa uma célula da lista. O jogador ganha quando chegar em “NULL”;

Lista Duplamente Encadeada
Atividades (Desafio)



1 x 5 = ? | 1 x 5 = 5 –> Acertou!!!



2 x 6 = ? | 2 x 6 = 8 –> Errou!!!