Compiladores

2ºano de Engenharia Informática

Trabalho Prático 3

Vitor Alberto de Sá Ribeiro (al78138)
Tiago Silva (al78417)

2 de janeiro de 2024

Soluções Implementadas

O nosso programa pode ser dividido em **4 partes principais**:

Definição de variáveis:

- O **nível de bateria** (inteiro)
- **Localizações possíveis** (array de strings)
- **Localização atua**l (inteiro correspondente ao index do array de strings)
- **Linha de montagem**, utilizada na entrega de materiais (string)
- **Contador de manutenção** (inteiro)
- **Carga total do carrinho** (inteiro)
- **Lista de tuplas**, structs utilizadas para referenciar qualquer material

Funções auxiliares:

Estas funções **já tinham sido utilizadas na última etapa do trabalho**, nesta etapa apenas nos limitamos a transportá-las para o parser YACC/BISON, tendo apenas adicionado uma função usada maioritariamente para debugging.

- **yyerror** que recebe uma mensagem de erro e foi criada para tornar mais fácil o debug do código, uma vez que devolve essa mesma mensagem de erro juntamente com a linha onde o erro aconteceu, o que nos permitiu identificar e resolver os erros muito mais facilmente
- **countOpenParentheses** que recebe uma string e devolve o número de parênteses abertos dentro da string
- **countOpenCommas** que recebe uma string e devolve o número de vírgulas
- **storeTuples** que recebe uma string com os materias a recolher e separa a string em tuplas para depois as guardar na lista de tuplas

- **removeTuples** que recebe uma string com o material a remover, um inteiro que é a quantidade a remover e a posição da lista em que está a tupla a ser alterada
- **print_states** que é a função utilizada para mostrar o estado do veículo, é apresentada sempre depois de qualquer ação feita pelo carro
- **deslocar** que recebe um index da lista de localizações(inteiro) e desloca o veículo até à localização pretendida
- **print_Estado** que é a função específica utilizada na tarefa ESTADO(I) e recebe um array de 0-3 elementos e mostra as informações necessárias dependendo da quantidade de elementos passados

Funções principais:

- MANUTENCAO(V) – indica ao veículo que se deve dirigir para o posto de manutenção, onde V pode assumir valores de 0, 1 ou 2. 0 significa que se deve deslocar imediatamente, 1 significa que antes de se deslocar para o posto de manutenção deve primeiro terminar alguma tarefa que esteja a desempenhar no momento, e 2 significa que antes de se deslocar para o posto de manutenção deve terminar todas as tarefas que tenha pendentes.

```
int fmanutencao(int numero){
    //deve se deslocar ate a manutencao
    if (numero > 2 || numero < 0 ){
        printf("Erro %d fora dos limites da manutencao\n",numero);
        return 1;
    }

    if (localizacao != 1){
        if(Estado_Bateria - 10 - Carga_Total >= 0){ //se bateria for suficiente
            deslocar(1);
            Estado_Bateria = Estado_Bateria - 10 - Carga_Total;
            Vezes_Manutencao ++;
            if(Vezes_Manutencao == 3){
                printf("Veiculo foi 3 vezes a manutencao, contador sera colocado a 0\n");
                Vezes_Manutencao = 0;
            }
        }
        else{
            printf("Nao ha bateria suficiente para esta deslocacao");
        }
    }
    else{
        printf("O veiculo ja se encontra na zona de manutencao\n");
    }

    print_states();
    return 0;
}
```

- CARREGA-BATERIA(V) – indica ao veículo que se deve dirigir para o posto de carregamento, onde V pode assumir valores de 0, 1 ou 2. 0 significa que se deve deslocar imediatamente, 1 significa que antes de se deslocar para o posto de carregamento deve primeiro terminar alguma tarefa que esteja a desempenhar no momento, e 2 significa que antes de se deslocar para o posto de carregamento deve terminar todas as tarefas que tenha pendentes.

```
int fcarrega(int numero){
        //deve se deslocar ao posto de carregamento
    if (numero > 2 || numero < 0 ){
        printf("Erro %d fora dos limites do carregamento",numero);
        return 1;
    }

    if(Estado_Bateria == 100){

        printf("Foi pedido um carregamento com a bateria a 100\n");
    }

    if (localizacao != 0){
        if(Estado_Bateria - 10 - Carga_Total >= 0){ //se bateria for suficiente
            deslocar(0);
            Estado_Bateria = 100;
        }
        else{
            printf("Nao ha bateria suficiente para esta deslocacao\n");
        }
    }
    else{
        printf("O veiculo ja se encontra na zona de carregamento \n");
    }

    print_states();
    return 0;
}
```

- ENTREGA(L,M,Q) – indica ao veículo que deve efetuar uma entrega numa linha de montagem. L identifica a linha de montagem através de um valor numérico entre 1 e 100 antecedido por duas letras maiúsculas. M identifica o material através de um código de 5 caracteres composto por letras e/ou números. Q representa a quantidade através de um valor numérico que pode assumir valores maiores que zero.

```
int fentrega(char* L, char* M, int Q){
    //deve fazer uma entrega na linha de montagem

    if (Q <= 0){
        printf("Quantidade nao pode assumir valores menores ou iguais a zero \n");
        return 1;
    }

    int PosicaoRemover = -1;


    for (int i = 0; i <Posicao_Tupla; i++){

        if(strcmp(TupleList[i].material,M) == 0){
            PosicaoRemover = i;
        }
    }

    if(PosicaoRemover == -1){
        printf("Material nao existe\n");
    }
    else{
        if (localizacao != 3){ //nao se encontra na linha de montagem
            if(Estado_Bateria - 10 - Carga_Total >= 0){ //se bateria for suficiente
                deslocar(3);
                Estado_Bateria = Estado_Bateria - 10 - Carga_Total;
                strcpy(Linha_Montagem,L);
                removeTuples(M,Q,PosicaoRemover); //fazer uma entrega
            }
            else{
                printf("Nao ha bateria suficiente para esta deslocacao");
            }
        }
        else{ //ja se encontra na linha de montagem
            if(strcmp(Linha_Montagem,L) == 0){
                printf("Ja se encontra na linha de montagem %s\n",Linha_Montagem);
                removeTuples(M,Q,PosicaoRemover); //fazer uma entrega
            }
            else{
                if(Estado_Bateria - 5 - Carga_Total >= 0){ //se bateria for suficiente
                    strcpy(Linha_Montagem,L);
                    printf("Ja se encontra numa linha de montagem, moveu-se para %s\n",Linha_Montagem);
                    Estado_Bateria = Estado_Bateria - 5 - Carga_Total;
                    removeTuples(M,Q,PosicaoRemover); //fazer uma entrega
                }
                else{
                    printf("Nao ha bateria suficiente para esta deslocacao");
                }

            }
        }
    }

    print_states();
    return 0;
}
```

- RECOLHE(LISTA)– indica ao veículo que se deve deslocar ao armazém para efetuar a recolha de uma lista de materiais. LISTA representa a lista de materiais, iniciada e terminada por [e] respetivamente, e onde cada elemento da lista é representado por uma tupla no formato (M,Q) onde M identifica o material através de um código de 5 caracteres composto por letras e/ou números e Q representa a quantidade através de um valor numérico que pode assumir valores inteiros maiores que zero. Exemplo: RECOLHE([(A4gt6,300), (cbv45,3), (12345,21)]).

```
int fcarrega(int numero){
        //deve se deslocar ao posto de carregamento
    if (numero > 2 || numero < 0 ){
        printf("Erro %d fora dos limites do carregamento",numero);
        return 1;
    }

    if(Estado_Bateria == 100){

        printf("Foi pedido um carregamento com a bateria a 100\n");
    }

    if (localizacao != 0){
        if(Estado_Bateria - 10 - Carga_Total >= 0){ //se bateria for suficiente
            deslocar(0);
            Estado_Bateria = 100;
        }
        else{
            printf("Nao ha bateria suficiente para esta deslocacao\n");
        }
    }
    else{
        printf("O veiculo ja se encontra na zona de carregamento \n");
    }

    print_states();
    return 0;
}
```

- ESTADO(I) - indica ao veículo que deve comunicar o seu estado atual. I identifica a informação que deve ser comunicada, podendo assumir o valor de B (representa o estado da bateria); M (representa os materiais e quantidades que está a carregar); T (representa as tarefas que tem pendentes); ou qualquer combinação entre estas 3 letras, sendo que se existir mais que uma letra, estas devem ser separadas por uma virgula.

```
void festado(char* string){
    //Indica o estado atual do veiculo onde B - Bateria  M- Lista de materiais t - tarefas pendentes
    //ESTADO(B,M,T) 3
    //ESTADO(B) 1
    //ESTADO(B,M) 2
    char Caracteres[3];

    string[strlen(string) ] = ')';

    int array[3] = {0,0,0}; //bateria,tarefas,materiais e quantidade
    char* token;

    int comas = countOpenCommas(string);


    if(comas == 0){
        token = strtok(string,")");
        Caracteres[0] = *token;

        for(int i=0;i<1;i++){
            if(Caracteres[i] == 'B'){
                array[0] = 1;
            }
            if(Caracteres[i] == 'T'){
                array[1] = 1;
            }
            if(Caracteres[i] == 'M'){
                array[2] = 1;
            }
        }
    }
     else if(comas == 1){
         token = strtok(string,",");
         Caracteres[0] = *token;
         token = strtok(NULL,")");
         Caracteres[1] = *token;

         for(int i=0;i<2;i++){
             if(Caracteres[i] == 'B'){
                 array[0] = 1;
             }
             if(Caracteres[i] == 'T'){
                 array[1] = 1;
             }
             if(Caracteres[i] == 'M'){
                 array[2] = 1;
             }
         }
     }
     else if(comas == 2){
         for(int i = 0;i<3;i++){
             array[i] = 1;
         }
     }

     print_Estado(array);


}
```

- INIT-ESTADO(L,B,M,N) – onde L indica a localização inicial do veículo, B o estado (%) inicial da bateria, M representa os materiais e respetivas quantidades que está a carregar, e N o número de vezes que o veículo foi à manutenção.

```
int finitestado(int local, int bateria, char* lista, int manu){
    if(bateria < 0 || bateria > 100){
        printf("Bateria fora dos limites 0-100");
        return 1 ;
    }

    if(manu < 0){
        printf("Quantidade de vezes da manutencao fora dos limites >= 0");
        return 1;
    }

    localizacao = local;
    Estado_Bateria = bateria;
    Vezes_Manutencao = manu;

    if (strcmp(lista, "NULL") != 0){
        storeTuples(lista);
    }

    print_states();

    return 0;
}
```

Definição de tokens:

```
%union{
    int numero;
    char* mystring;
    char caracter;
}

%token START END
%token<mystring> CARREGA ESTADO ENTREGA RECOLHE MANUTENCAO INITESTADO
%token<caracter> PONTOEVIRGULA CHAVETAE CHAVETAD PARENTE PARENTD RETOE RETOD VIRGULA
%token<mystring> L M I LISTA
%token<numero> Q LOCALIZACAO



%%

programa: START CHAVETAE initestado conjunto_instrucoes CHAVETAD END {printf("Conjunto processado corretamente\n");}
        | error END {printf("Erro de formatação no conjunto de instruções.\n");}
        ;

conjunto_instrucoes: lista_instrucoes instrucao PONTOEVIRGULA
                   | instrucao PONTOEVIRGULA
                   | lista_instrucoes instrucao
                   | instrucao
                   ;

lista_instrucoes: lista_instrucoes instrucao PONTOEVIRGULA
                | instrucao PONTOEVIRGULA
                ;

initestado :|INITESTADO PARENTE LOCALIZACAO VIRGULA Q VIRGULA RETOE LISTA RETOD VIRGULA Q PARENTD PONTOEVIRGULA {finitestado($3,$5,$8,$11);}
           |INITESTADO PARENTE LOCALIZACAO VIRGULA Q VIRGULA VIRGULA Q PARENTD PONTOEVIRGULA {finitestado($3,$5,"NULL",$8);}
           |
           ;



instrucao: carrega
         |   estado
         |   entrega
         |   recolhe
         |   manutencao
         ;
carrega : CARREGA PARENTE Q PARENTD {fcarrega($3);}

estado : ESTADO PARENTE I PARENTD {festado($3);}

entrega: ENTREGA PARENTE L VIRGULA M VIRGULA Q PARENTD {fentrega($3, $5, $7);}
       |ENTREGA PARENTE L VIRGULA Q VIRGULA Q PARENTD {
           if($5 >= 10000 && $5 <= 99999 ){
               char auxiliar[6];
               sprintf(auxiliar, "%d", $5); //converter de int para string
               fentrega($3, auxiliar, $7);
           }
           }

recolhe: RECOLHE PARENTE RETOE LISTA RETOD PARENTD {frecolhe($4);}

manutencao: MANUTENCAO PARENTE Q PARENTD {fmanutencao($3);}
```

Ficheiro:

INICIO-DAS-INSTRUCOES {

INIT-ESTADO(Posto de manutencao,70, ,0);
CARREGA-BATERIA(1);
RECOLHE([(A4gt6,40)]);
ENTREGA(LM035,A4gt6,20);
ENTREGA(RV002,A4gt6,20);
CARREGA-BATERIA(1);
RECOLHE([(12dF3,10)]);
ENTREGA(IU100,12dF3,10);

} FINAL-DAS-INSTRUCOES

- Indica o início das soluções com a expressão “INICIO-DAS-INSTRUCOES”
- Inicializa o veículo no posto de manutenção, com 70% de bateria, sem materiais e com o contador de manutenções a 0. **(70%)**
- Gasta 10% de bateria e desloca-se até ao posto de carregamento, carrega a bateria. **(100%)**
- Desloca-se até ao armazém e recolhe 40 unidades do material “A4gt6”. Gasta 10% de bateria . **(90%)**
- Entrega na linha “LM035” 20 do material “A4gt6”, o veículo fica com 20 materiais apenas. Gasta 10%(deslocação) + 40%(carga) de bateria. **(40%)**
- Entrega na linha “RV002” o que resta do material “A4gt6”, o veículo fica vazio. Gasta 10%(deslocação) + 20%(carga) de bateria. **(10%)**
- Gasta 10% de bateria e desloca-se até ao posto de carregamento, carrega a bateria. **(100%)**

- Desloca-se até ao armazém e recolhe 10 unidades do material “12dF3”. Gasta 10% de bateria . **(90%)**
- Entrega na linha “IU100” 10 unidades do material “12dF3”, o veículo fica vazio. Gasta 10%(deslocação) + 10%(carga) de bateria. **(70%)**
- Indica o final das soluções com a expressão “FINAL-DAS-INSTRUCOES”