ESTGOH
Escola Superior de Tecnologia e Gestão de Oliveira do Hospital
Instituto Politécnico de Coimbra

Licenciatura Tecnologias e Sistemas de Informação

Programação II

2019/20

Relatório

Ficha 6

Jorge Miguel dos Santos Martins Nº a2019100813

02/06/2020

Índice

1 Introdução

Introduzindo um conceito de uso de funções / subprogramas ao qual nós já uma vez ou outra teríamos usado, mas sem o conhecimento prévio teórico de tal funcionamento, nesta ficha de trabalho é nos dado a conhecer a chamada recursiva de subfunções no algoritmo por nós escrito. Habitualmente para correr uma lista de dados usaríamos um ciclo com condição de chegar ao seu final, nesta outra visão de funcionamento a função é chamada várias vezes até que uma condição seja efetuada. Podemos dizer que desta forma em alguns casos o código fica mais limpo, em outros não se encontram muitas vantagens, mas de grosso modo para grandes chamadas (repetidas vezes chamada de uma função) esta forma recursiva ocupa mais memoria do que simplesmente um ciclo a decorrer dentro de um subprograma.

2 Trabalho realizado

2.1 Pergunta 1

Neste exercício é pedido para dar o resultado do fatorial de um número, iniciando com um subprograma interativo é feito um ciclo for ao qual é feita a multiplicação do número em forma fatorial, aqui iniciei logo pela multiplicação com “fat=1” e fazendo “1*n” para que “fat” tome o valor logo do inicio e faça as multiplicações sucessivamente até 2, porque a multiplicação de 1 não realiza qualquer efeito no resultado.
Na chamada recursiva enquanto que o “n” que vai sendo chamado numa cadeia de funções de si mesma até chegar a 1, este vai multiplicando pelo número a seguir a si por ordem decrescente, o que faz uma substituição de um ciclo que tradicionalmente conhecemos, e assim criar um encadeamento de chamadas recursivas da mesma função (dando quase que como um ciclo feito manualmente).

2.2 Pergunta 2

Para este exercício ao qual nos é pedido a procura do elemento numa sequencia de Fibonacci, no ciclo interativo criei um ciclo for que começa com i=3, isto pois tenho logo duas variáveis que são os dois primeiros elementos da sequencia fb1=1 e fb=1, ora se o utilizador pedir ou o primeiro ou o segundo elemento este apenas envia o valor 1. Seguidamente são feitas as somas dos valores, dando depois ao fb1 o valor do elemento seguinte que encontra à sua frente na sequência e a fib2 damos o resultado da soma que é o resultado à sua direita original na sequência e assim sucessivamente até encontrarmos o elemento que o utilizador deseja ver.
Na forma recursiva enviamos os valores de n-1 e n-2 sendo as somas dos valores anteriores para o n que desejamos, assim encadeando a função ela irá retornar 1 quando ou o n é igual a 2 ou a 1 fazendo assim várias somas entre si até chegar ao resultado.
Deixo uma observação que para este esquema usei um desenho tal como costumo usar para a arquitetura de ponteiros.

2.3 Pergunta 3

Neste exercício através de recomendação da Docente da disciplina usei a logica que me foi demonstrada para a realização deste exercício. Confusão minha numa mistura de pontos da matéria tentando usar algo mais complexo quando o intuito desta ficha é usar algo simples e simplesmente perceber o fundamento da recursividade em funções.

Então demonstrando na função interativa realizei o tradicional ciclo, mas que aqui o seu limite é metade do tamanho do vetor, para que quando chegue a meio já todas as mudanças tenham sido realizadas, assim fazendo uma troca triangular para a inversão sendo o valor do meio o ponto que não muda neste vetor, mudo simplesmente o primeiro para ultimo e o segundo para quarto, através de uma triangulação de troca de valores sendo suportado pelo ciclo para o acerto dos índices a trocar.
Na sua face recursiva usei um subprograma que inicia com uma condição de que se a com o valor a é igual a b é então retornada a função caso contrário é feita a troca entre índices, usando a mesma logica anterior, mas com uma chamada do próprio subprograma (assim sendo recursivo) em que enviamos a+1 (a é inicialmente 0) e b-1 (que inicialmente é o tamanho do vetor menos 1 sendo o seu limite do índice).

2.4 Pergunta 4

Para este exercício é pedido para que ao utilizador insira um número e seguidamente se faça uma sequência decrescente até um e em seguida uma sequência crescente até ao número inicialmente dado pelo utilizador.
Para a resolução interativa usei um ciclo com condição, fazendo uma virada da sequencia quando a variável “aux” toma o valor 1, entrando assim na segunda fase da sequência fazendo uma ordem crescente numérica até que quando a variável “aux” se iguala ao número original inserido pelo utilizador é dado um break(numa condição if após o while da segunda sequencia) assim o primeiro ciclo não reinicia uma nova sequencia decrescente e o programa entre em um loop infinito.
Para um subprograma com uso de recursividade usei a logica de modo contrário inicialmente faço a primeira condição de uma sequência crescente, isto porque C é estrutural e a condição de entrada para esta condição só será possível após a sequência decrescente seja realizada no “else”, assim após a primeira fase estar completa sei que o subprograma não irá cair em loop infinito sendo “preso” para que apenas funcione na primeira condição (usada em segunda fase da sequência). A grande condicionante é a variável ancora que recebe do main o valor 1 que é o valor de v após o final da sequencia decrescente e a cada chamada em recursividade a ancora é incrementada para ao ser enviada esteja igual ao valor de v que é também incrementado.

2.5 Pergunta 5

Nesta pergunta é pedido para que seja adaptado o subprograma “limpa lista” para um modus operanti em recursividade, visto que à partida eu usei uma forma interativa de limpar a minha lista.

```
plista limparLista(plista lst){ //interativa
  plista ptr;
    while (lst != NULL) {
        ptr = lst;
        lst = lst->prox;
        free (ptr); // a cada vez que o lst para para o proximo o anterior é libertado...
    }
    puts("Lista eliminada");
    return NULL; //retorna null para o lst que é o ponteiro ao qual aponta para o inicio da lista.
}

plista limparListaRecursiva(plista lst){
  plista ptr;
  ptr=lst;
  lst=lst->prox;
  free(ptr);
  if(lst != NULL){
    lst=limparListaRecursiva(lst);
  }
  else{
    return lst;
  }

}
```

Figura 1 – LimpaLista

Para mais facilmente se notar as diferenças ao observar a figura 1, posso descrever que a logica tanto na interativa como na recursiva é a mesma, mas na primeira uso um ciclo que verifica se o ponteiro “lst” é nulo, enquanto diferente de nulo ele corre a lista ligada e vai libertando o ponteiro que para aí estaria a apontar.
Enquanto que numa forma recursiva uso apenas uma condição “if” ao qual enquanto não for retornado um ponteiro “NULL” é feita uma chamada recursiva encadeada do próprio subprograma a libertar cada espaço de memoria.

3 Conclusão

Esta ficha elucidou uma forma diferente de uso dos subprogramas, mesmo já tendo existido algumas tentativas de me habituar enquanto estudante de programação na reutilização de subprogramas dentro do meu próprio código ainda caio no erro de principiante de duplicar código desnecessariamente.

Assim este foi uma ficha que demonstrou não só a importância do uso da recursividade como outro meio de uso em vez de simplesmente usar vários ciclos para uma determinada função que se deseja, mas também na possibilidade da chamada de vários subprogramas dentro de eles próprios para a realização de várias tarefas.

Será futuramente um ponto a ter em conta na realização do meu código.

4 Referências

Listar consultas usadas

- Luis Damas Liguagem C

5 Anexos

Exer1.c

Exer2.c

Exer3.c

Exer4.c

Exer5.c