UConv

Relatório de desenvolvimento do “UConv”

Instituto Superior Técnico

Mestrado Integrado em Engenharia Aeroespacial

Desempenho

Diogo Sousa

95779

1. Introdução

No âmbito da disciplina de Desempenho do segundo ano de Mestrado Integrado em Engenharia Aeroespacial desenvolvi como trabalho opcional um conversor de unidades de diferentes grandezas físicas, correspondente ao ponto 6 dos trabalhos opcionais ("Ilustração da conversão de algumas unidades").

O trabalho desenvolvido consiste num programa simples e, penso eu, intuitivo que permite fazer conversões rápidas entre diferentes unidades.

2. Desenvolvimento

Todo o programa foi desenvolvido usando linguagem de programação C e recorrendo a "makefile" para construção facilitada do executável do programa. Todo o código fonte se encontra junto do executável e no GitHub.

O UConv utiliza três bibliotecas especialmente desenvolvidas para o programa e as seguintes bibliotecas standard da linguagem C: stdio.h, stdlib.h, math.h, string.h e ctype.h.

As bibliotecas desenvolvidas para o programa têm os seguintes nomes e funções:

- ConversionFunctions.h – Onde se encontram as funções que fazem a conversão entre as unidades.
- Interface.h – Onde se encontram as funções que mostram diálogo e onde se faz o tratamento do input do utilizador.
- Formules.h – Onde se encontram as funções que imprimem para a linha de comandos as fórmulas utilizadas numa conversão.

As secções seguintes referem-se a cada uma das bibliotecas e ficheiros usados.

3. ConversionFunctions

Como é habitual em C, o código fonte tem tanto para esta biblioteca como para as restantes dois ficheiros de igual nome, mas extensão diferente. O ficheiro com extensão .c inclui todo o corpo de todas as funções da biblioteca. O ficheiro com extensão .h contém os protótipos das funções definidas no ficheiro .c, bem como todas as macrodefinições e outras bibliotecas necessárias para a definição das funções.

No caso específico da biblioteca ConversionFunctions, esta consiste em diversas funções que fazem a conversão de ou para a unidade SI da grandeza em questão. Por forma a reduzir o número de funções necessárias foi usado este método de conversão para SI como intermédio entre duas unidades que não a SI.

A sintaxe das funções é idêntica para cada conversão, pelo que incluirei apenas um exemplo explicado de uma destas funções de conversão, sendo as restantes funções iguais com fatores de conversão diferentes.

```
float Unit_Convert_Mass_Lb_to_Kg (float mass){
  float fct = 0.45359237;
  return(mass*fct);
}
```

Esta função converte valores de massa em libras para quilogramas, a unidade SI de massa. A função recebe como argumento o valor de uma massa em libras, que armazena na variável mass, para depois transformar num valor em quilogramas, multiplicando mass por fct, o fator de conversão entre libras e quilogramas. A função devolve, então, o valor da massa inserida em quilogramas. Quanto às funções de transformação da unidade SI numa outra unidade, apenas se altera a expressão de retorno da função, como se pode ver no exemplo seguinte:

```
float Unit_Convert_Mass_Kg_to_Lb (float mass){
  float fct = 0.45359237;
  return(mass/fct);
}
```

Esta função converte um valor de massa em quilogramas num valor de massa em libras, sendo a função inversa da função referida anteriormente.

Todas as funções desta biblioteca são definidas desta forma com exceção das funções para unidades de temperatura, que por vezes envolvem o uso de mais de um fator de conversão e que, ao contrário das restantes grandezas se relacionam através de somas de fatores e não produtos e quocientes. Apresenta-se então um exemplo de uma conversão de unidades de temperatura:

```
float Unit_Convert_Temperature_oC_to_K (float temperature){
  float fct = 273.15;
  return (temperature+fct);
}
```

Está também incluída nesta biblioteca uma função que permite a inversão de unidades, embora não esteja implementada no programa.

4. Formules

À semelhança da biblioteca anterior esta consiste em diversas funções que mostram ao utilizador as fórmulas usadas na conversão entre duas unidades. Um exemplo destas funções é a função que mostra a fórmula de conversão entre libras e quilogramas que é apresentada a seguir:

```
void Unit_Convert_Mass_Lb_to_Kg_Formula (){
  printf("[Lb]*0.45359237 = [Kg]\n");
}
```

Estas funções não necessitam de argumentos e não devolvem qualquer valor, usando apenas a função “printf”, incluída em “stdio.h”.

Todas as funções desta biblioteca são definidas de maneira similar.

5. Interface

Nesta biblioteca estão incluídas funções que fazem o tratamento do input do utilizador e que fazem uso das funções definidas nas restantes bibliotecas. As funções que imprimem no ecrã indicações para o utilizador estão também incluídas nesta biblioteca, como é exemplo a função que mostra a lista de quantidades físicas suportadas pelo programa, apresentada a seguir.

Por conveniência e para evitar repetição foram suprimidas algumas linhas de código. AS supressões encontram-se indicadas com o comentário "/*…*/".

```
void pqlist (){
  printf("\n| Physical Quantity |\n");
  printf("->Mass\n");
  printf("->Distance\n");
  printf("->Volume\n");
  printf("->Force\n");
  printf("->Density\n");
  printf("->Speed\n");
  printf("->Pressure\n");
  printf("->Frequency\n");
  printf("->Energy\n");
  printf("->Power\n");
  printf("->Temperature\n\n");
}
```

A função "select_menu" recebe como argumento uma string com o input do utilizador e caso o input corresponda a um comando válido, executa a função que lhe é respetiva. Transcrevendo apenas uma parte desta função, devido à extensão desta, correspondente à execução do comando **pqlist** que mostra a lista de unidades.

```
void select_menu (char line [MAX]){
  lowercase(line);
  /*…*/

  /*Mostra lista de unidades*/
  if(strcmp(line , "pqlist")==0)
        pqlist();

  /*Mass Menu*/
  if(strcmp(line , "mass")==0)
        mass_interface (line);
  /*…*/
}
```

O argumento da função, a linha de input do utilizador, é denominado line. A função "strcmp" compara o argumento line a uma outra string, neste caso "pqlist" e caso sejam iguais devolve 0, executando a função "pqlist" que mostrará a lista de quantidades físicas.

Para o caso de o argumento se referir a um submenu, a linha é necessária como argumento para a interface deste submenu, sendo então passada para esta função como é possível observar nas linhas que sucedem o comentário "/*Mass Menu*/".

As funções de interface em submenus são semelhantes entre si, alterando-se apenas as funções chamadas e as comparações de comandos.

Nos submenus (e no menu inicial) o programa corre com recurso a um ciclo "do, while" e para apenas quando o comando inserido é **back** ou **exit** (apenas quando o comando é **exit** para o menu inicial). Os submenus são apresentados como um prefixo ao input do utilizador, por forma a distinguir cada submenu e o menu principal.

Para o submenu massa, o prefixo será da forma "UConv/Mass->", por exemplo. Para outras grandezas, a grandeza mostrada a seguir a "UConv/" altera-se. No menu principal, o prefixo é "UConv->".

A função "lowercase", que faz uso da função "tolower" incluída em "ctype.h", permite que o argumento possa ser escrito com maiúsculas e/ou minúsculas.

```
void lowercase (char string [MAX]){

int i;
for (i=0; i<MAX;i++){
        string[i] = tolower(string[i]);
}
}
```

Através de um ciclo "for", todos os caracteres alfabéticos do argumento da função, a string denominada "string", são colocados em minúscula.

As funções de interface de submenus fazem ainda uso de outras funções definidas após a função de interface principal.
Para o caso de interface do submenu massa, a função é a seguinte:

```
void mass_interface (char line[MAX]){
char unit[2][MAX];
float number;
do{
    printf("UConv/Mass->");
    gets(line);
    lowercase(line);
    if((3 == sscanf(line , "%s to %s %f" , unit[1] , unit[2] , &number))
    &&(mass_check(unit[1])) && (mass_check(unit[2]))){

        /*Lê Input e Verifica unidades para conversão*/
        /*Unidade origem*/
        if(strcmp(unit[1] , "kg")==0){}

        if(strcmp(unit[1] , "lb")==0)
```

```
                    number = Unit_Convert_Mass_Lb_to_Kg (number);
                /*..*/

                /*Unidade destino*/
                if(strcmp(unit[2] , "kg")==0){}

                if(strcmp(unit[2] , "lb")==0)
                    number = Unit_Convert_Mass_Kg_to_Lb (number);
                /*…*/
                printf("%f\n" , number);
                }
  if(2 == sscanf(line , "sf %s to %s" , unit[1] , unit[2]) && mass_check(unit[1]) &&
mass_check(unit[2])){
                /*Unidade origem*/
                if(strcmp(unit[1] , "kg")==0){}

                if(strcmp(unit[1] , "lb")==0)
                    Unit_Convert_Mass_Lb_to_Kg_Formula ();

                /*…*/

                /*Unidade destino*/
                if(strcmp(unit[2] , "kg")==0){}

                if(strcmp(unit[2] , "lb")==0)
                    Unit_Convert_Mass_Kg_to_Lb_Formula ();

                /*…*/
        }
                if (strcmp(line , "units")==0){
                    mass_units();
                }
  }while((strcmp(line , "back")!=0)&&(strcmp(line , "exit")!=0));
  }
```

A função “mass_interface” recebe como argumento a string de input do utilizador denominada “line”. Através de comparação da string com os comandos possíveis, com recurso à função “strcmp”, já explicada anteriormente, a função realizará a operação correspondente a cada comando.

Caso o comando seja da forma **[unidade] to [unidade] [valor]** e caso as unidades correspondam às unidades possíveis para, neste caso, massa, a função usará as funções de conversão para apresentar o valor inserido na unidade precedente ao “to” na unidade sucessora do “to”.

Caso uma destas unidades seja a unidade SI (para a massa caso uma destas unidades seja quilogramas) a função não irá realizar nenhuma ação. Para executar esta conversão é em primeiro lugar analisada a primeira unidade. Não sendo esta SI o valor inserido é convertido desta primeira unidade para a unidade SI. Depois é analisada a segunda

unidade. Se esta não for a unidade SI, o valor resultante da conversão anterior é novamente convertido para a unidade final. O valor é depois apresentado.

Para um comando da forma **sf [unidade] to [unidade]** e caso as unidades correspondam às unidades possíveis para, neste caso, massa, a função usará as funções de apresentação de fórmulas para mostrar as fórmulas usadas numa conversão entre a unidade precedente ao "to" e a unidade sucessora ao "to". A apresentação é realizada de forma semelhante à conversão.

Caso a unidade precedente a "to" não seja SI, a fórmula de conversão desta unidade para a unidade SI é apresentada. Caso a unidade sucessora a "to" não seja SI, a fórmula de conversão da unidade SI para esta unidade é apresentada.

O comando **units** apresenta a lista de unidades entre as quais é possível, atualmente converter valores, usando a função "mass_units" para mostrar ao utilizador estas unidades. A função "mass_units" é semelhante à função "pqlist", alterando-se apenas os valores exibidos.

Tanto os comandos **back** como **exit** abandonam este submenu, sendo que **exit** termina o programa.

Outros pontos importantes são também a impressão do prefixo referente ao submenu no início de cada ciclo "do, while", a função "gets" que guarda o input do utilizador na string line, a função "lowercase" que coloca em letras minúsculas toda a string line e a função "sscanf", que analisa a string line para executar cada comando.

As unidades referentes a cada grandeza física são verificadas usando funções como a função "mass_check", que se apresenta a seguir:

```
int mass_check (char unit[MAX]){
  if((strcmp(unit , "kg")==0) || (strcmp(unit , "lb")==0) || (strcmp(unit , "slug")==0) ||
(strcmp(unit , "oz")==0) || (strcmp(unit , "c")==0))
        return TRUE;
  else
        return FALSE;
}
```

Esta função recebe como argumento uma string, denominada unit, e compara-a com todas as unidades suportadas para uma grandeza física.

Caso corresponda a uma destas, a função devolve "TRUE", uma macrodefinição para um valor diferente de zero correspondente em C a um valor lógico verdadeiro.

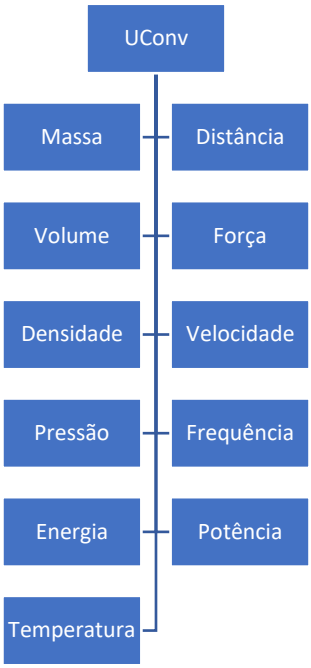
Caso não corresponda a nenhuma das unidades suportadas, a função devolve "FALSE", uma macrodefinição para o valor zero, correspondente em C a um valor lógico falso.

6. Main

A função main inclui um ciclo “do, while” simples que imprime o prefixo do menu principal e analisa o input do utilizador (com recurso à função “select_menu” referida anteriormente), bem como a definição inicial da string onde é armazenado o input e mensagens de boas-vindas e de despedida.

```
int main (void){
    /*string for input*/
    char line [MAX];
    /*Welcome message*/
    printf("Welcome!\n");
    printf("Use help for a list of instructions\n");
    do{
            printf("UConv->");
            gets(line);
            select_menu (line);
    }while(strcmp(line , "exit")!=0);
    /*Good Bye message*/
    printf("Goodbye!");
    return 0;
}
```

Segue-se um fluxograma simples com identificação de todas as unidades suportadas pelo UConv.



Devido à grande quantidade de unidades suportadas pelo programa, a verificação de cada uma destas é realizada em vídeo, usando a calculadora de conversão de unidades do Google como referência.

7. Conclusão

Embora não contenha nenhuma funcionalidade específica que torne este programa inovador, creio que este poderá ser útil para uso integrado com outros programas por forma a facilitar a conversão entre unidades. A biblioteca de funções de conversão desenvolvida poderá ser especialmente útil para uso incluído num outro programa.

Concluindo, acho que o trabalho foi realizado com sucesso e que poderá ser um programa útil e intuitivo que facilite a conversão de unidades.