Apêndice A

Reuso de programas

Sabemos programar não é uma tarefa fácil. Por isso, sempre que podemos, devemos reutilizar os códigos que já fizemos. Porém, reutilizar código não é copiar e colar um código pronto em uma nova aplicação. A ideia de um padrão de desenvolvimento copy&paste é inconcebível. Imagine se quisermos alterar uma implementação depois desta ser copiada e colada mil vezes! Será que conseguiríamos alterar todas as cópias da implementação? Ou deveríamos adaptar os novos códigos a implementação antiga, tornando-a imutável? Claro que a resposta correta as estas duas perguntas é NÃO. Então, se não podemos copiar e colar um código, como devemos programar para que a alteração de uma implementação ocorra em apenas um lugar?

A.1 Dividindo programas

A linguagem C nos permite dividir nossos programas em mais de um arquivo fonte. Isso nos permite utilizar um trecho de código em mais de um programa. Suponha que precisamos de fazer um programa que calcule as raízes de uma equação do segundo grau. Podemos implementar a fórmula de Báskara diretamente no programa, mas se implementarmos a fórmula de Báskara em um arquivo separado, poderemos fazer vários programas que usam esta implementação.

Algoritmo 27: baskara.c

```
1  #include <math.h>
2  float delta(float a, float b, float c) {
3    return b*b - 4*a*c;
4  }
5  /* Coloca o valor das raizes nas variaveis x1 e x2, se existirem.
6  ** Retorna a quantidade de raizes encontradas. */
7  int baskara(float a, float b, float c, float *x1, float *x2) {
8    float d = delta(a, b, c);
9    /* Se o delta negativo a equacao nao possui raizes */
10   if(d < 0)
11     return 0;
12   /* Se o delta for zero so existe uma raiz */
13   if(d == 0){
14     *x1 = *x2 = ((-1)*b)/(2*a);
15     return 1;
16   }
17   /* Se o delta for positivo existem duas raizes reais */
18   *x1 = ((-1)*b+sqrt(d))/(2*a);
19   *x2 = ((-1)*b-sqrt(d))/(2*a);
20   return 2;
21 }
```

Note que incluímos a biblioteca *math.h*, pois utilizamos a função *sqrt*. Porém, não incluímos a bilbioteca *stdio.h*, já que não utilizamos nenhuma função de entrada e saída padrão. Isso nos permite utilizar esta

implementação da fórmula de Báskara em vários programas, que utilizam diferentes interfaces com o usuário, ou até mesmo em programas que necessitam da implementação desta fórmula e o usuário sequer sabe que esta fórmula esta sendo utilizada. Para utilizarmos as funções implementadas em *baskara.c* basta incluirmos (*include*) este arquivo no código fonte onde ele é necessário.

Algoritmo 28: eq2grau.c

```
1  #include "baskara.c"
2  #include <stdio.h>
3  int main() {
4    float a, b, c, x1, x2;
5    int qraizes;
6    printf("Digite os valores de a, b e c: ");
7    scanf("%f %f %f", &a, &b, &c);
8    qraizes = baskara(a, b, c, &x1, &x2);
9    if(a == 0)
10     printf("Para ser equacao do 2o. grau 'a' deve ser diferente de zero.\n");
11   else
12     if(qraizes == 0)
13       printf("Esta equacao nao possui raizes reais.\n");
14     else
15       if(qraizes == 1)
16         printf("Esta equacao possui apenas uma raiz real: x = %f\n", x1);
17       else
18   printf("Esta equacao possui duas raizes reais: x1 = %f e x2 = %f\n", x1, x2)
       ;
19   return 0;
20 }
```

Note que o arquivo *baskara.c* esta entre aspas e não entre < e >. O arquivo *baskara.c* deve estar no mesmo diretório do arquivo *eq2grau.c*. Agora que precisamos de interação com o usuário, podemos incluir a biblioteca *stdio.h*. Se refizermos este programa utilizando uma interface gráfica, podemos ainda utilizar o arquivo *baskara.c*. O comando do gcc para compilar este programa pode ser: `gcc eq2grau.c -o eq2grau.bin -Wall -lm`.

A.2 Escondendo o código fonte

Na maioria das vezes precisamos de saber apenas para que serve uma função, quais são seus parâmetros e seus possíveis valores de retorno. Por outro lado, existem situações em que queremos compartilhar a utilização dos nossos códigos mas queremos esconder a nossa implementação. Para isso podemos dividir nossas implementações em dois arquivos: Um com a interface das funções (quais seus parâmetros e possíveis valores de retorno) e outro com a implementação propriamente dita. Os arquivos com interfaces de programação de aplicações (API - Application Programming Interface) possuem a extensão '.h', que significa Header, ou cabeçalho.

Algoritmo 29: baskara.h

```
1  /* Coloca o valor das raizes nas variaveis x1 e x2, se existirem.
2  ** Retorna a quantidade de raizes encontradas. */
3  int baskara(float a, float b, float c, float *x1, float *x2);
```

Note que só colocamos a assinatura da função baskara. Quem utilizar nossa implementação não precisa saber que usamos uma função auxiliar delta para calcular as raízes da equação, como mostra o algoritmo 30.

A função delta foi implementada mas não foi declarada no arquivo de cabeçalho. Isto significa que podemos utilizar função delta apenas no arquivo baskara.c. Nós incluímos o arquivo baskara.h no arquivo baskara.c. Isto permite que utilizemos, dentro do arquivo .c, uma função declarada no arquivo .h, acima de

Algoritmo 30: baskara.c

```
1  #include "baskara.h"
2  #include <math.h>
3  float delta(float a, float b, float c) {
4    return b*b - 4*a*c;
5  }
6  int baskara(float a, float b, float c, float *x1, float *x2) {
7    float d = delta(a, b, c);
8    if(d < 0)
9      return 0;
10   if(d == 0){
11     *x1 = *x2 = ((-1)*b)/(2*a);
12     return 1;
13   }
14   *x1 = ((-1)*b+sqrt(d))/(2*a);
15   *x2 = ((-1)*b-sqrt(d))/(2*a);
16   return 2;
17 }
```

sua implementação. Além disso, a inclusão do arquivo .h no seu respectivo .c, garante que as assinaturas e os tipos de retorno das funções estão iguais. Podemos agora criar um módulo compilado de baskara. Este modulo possui a extenção .o (object) e é utilizado em conjunto com o arquivo .h. O comando do gcc para compilar um arquivo baskara.c em baskara.o é: `gcc baskara.c -c`. Este comando gera o arquivo baskara.o no mesmo diretório que baskara.c. Depois da compilação em .o não precisamos mais do arquivo .c. Para utilizarmos esta biblioteca que acabamos de criar, podemos implementar nosso programa da conforme algoritmo 31.

Algoritmo 31: eq2grau.c

```
1  #include "baskara.h"
2  #include <stdio.h>
3  int main(){
4    float a, b, c, x1, x2;
5    int qraizes;
6    printf("Digite os valores de a, b e c: ");
7    scanf("%f %f %f", &a, &b, &c);
8    qraizes = baskara(a, b, c, &x1, &x2);
9    if(a == 0)
10     printf("Para ser equacao do 2o. grau 'a' deve ser diferente de zero.\n");
11   else if(qraizes == 0)
12       printf("Esta equacao nao possui raizes reais.\n");
13        else if(qraizes == 1)
14          printf("Esta equacao possui apenas uma raiz real: x = %f\n", x1);
15             else
16     printf("Esta equacao possui duas raizes reais: x1 = %f e x2 = %f\n", x1,
           x2);
17   return 0;
18 }
```

Note que incluímos o arquivo *baskara.h* e não *baskara.c*. O comando *gcc* para compilar este programa agora seria: `gcc eq2grau.c -o eq2grau.bin "baskara.o" -lm`. Da mesma forma que precisamos de *-lm* para carregar o módulo da *math.h*, também precisamos no *baskara.o* para utilizar o *baskara.h*. Os módulos locais, como baskara.o, devem ser carregados entre aspas.