

Tema: Introdução à programação

Atividade: Montagem de programas - Karel

- 01.) Editar e salvar um esboço de programa,
o nome do arquivo deverá ser Guia0101.c,
concordando maiúsculas e minúsculas, sem espaços em branco, acentos ou cedilha:

```
/*
  Guia_0101 - v0.0. - __ / __ / ____
  Author: _____

  Para compilar em uma janela de comandos (terminal):

  No Linux   : gcc -o Guia_00 ./Guia_00.c
  No Windows: gcc -o Guia_00.exe Guia_00.c

  OBS.: O arquivo de definições karel.h devese estar disponível
        na pasta include do compilador ou na pasta do programa.

  Para executar em uma janela de comandos (terminal):

  No Linux   : ./Guia0101.c
  No Windows: Guia0101

*/
// lista de dependencias
#include "karel.h" // na pasta do programa

// ----- definicoes de metodos

/**
  decorateWorld - Metodo para preparar o cenario.
  @param fileName - nome do arquivo para guardar a descricao.
*/
void decorateWorld ( char* fileName )
{
  // definir o contexto
  ref_world v_world = world_now;

  // colocar um marcador no mundo
  set_World ( 4, 4, BEEPER );

  // salvar a configuracao atual do mundo
  save_World( fileName );
} // decorateWorld ( )
```

```

/**
 * turnRight - Metodo para virar 'a direita.
 */
void turnRight( )
{
    // o executor deste metodo
    // deve virar tres vezes 'a esquerda
    turnLeft( );
    turnLeft( );
    turnLeft( );
} // end turnRight( )

// ----- acao principal

/**
 * Acao principal: executar a tarefa descrita acima.
 */

int main ( )
{
    // definir o contexto
    world v_world;      ref_world ref_v_world = ref v_world;      world_now = ref_v_world;
    robot v_robot;      ref_robot ref_v_robot = ref v_robot;      robot_now = ref_v_robot;
    box v_box ;         ref_box ref_v_box = ref v_box ;           box_now = ref_v_box ;

    // criar o mundo
    create_World ( "Guia_01_01_v01" );

    // decorar o ambiente com um marcador
    // OBS.: executar pelo menos uma vez,
    //      antes de qualquer outra coisa
    //      (depois de criado, podera' ser comentado)
    decorateWorld( "Guia0101.txt" );

    // comandos para tornar o mundo visivel
    reset_World( );      // limpar configuracoes
    set_Speed ( 1 );      // escolher velocidade
    read_World( "Guia0101.txt" ); // ler configuracao do ambiente

    // colocar o robo com a configuracao inicial
    create_Robot ( 1, 1, EAST, 0, "Karel" );

    // executar acoes
    move( );
    move( );
    turnLeft( );
    move( );
    move( );
    turnLeft( );
    move( );
    move( );
    turnLeft( );
    move( );
    move( );
    turnLeft( );
    turnLeft( );
    turnLeft( );
    turnOff ( );

    // preparar o encerramento
    close_World ( );

```

```

// encerrar o programa
    getchar ( );
    return ( 0 );
} // end main ( )

// ----- testes

/*
----- documentacao complementar

----- notas / observacoes / comentarios

----- previsao de testes

----- historico

Versao      Data      Modificacao
0.1         __/___     esboco

----- testes

Versao      Teste
0.1         01. ( )     teste inicial

*/

```

02.) Compilar o programa.

Se houver erros, identificar individualmente a referência para a linha onde ocorrem.

Consultar atentamente o modelo acima na linha onde ocorreu o erro (e também linhas próximas), editar as modificações necessárias.

Compilar novamente e proceder assim até que todos os erros tenham sido resolvidos.

Se não houver erros, seguir para o próximo passo.

OBS.: O ajuste de velocidade deverá ser feito para compatibilizar-se com o sistema operacional.

Recomenda-se, entretanto, não usar valores que acelerem demais o processo.

DICA: Se precisar de ajuda sobre como proceder a compilação,

consultar os vídeos com as demonstrações sobre algumas formas para fazê-lo.

SUGESTÃO: Para se acostumar ao tratamento de erros,

registrar a mensagem de erro (como comentário)

e o que foi feito para resolvê-lo.

03.) Executar o programa.

Observar as saídas.

Registrar os resultados.

```

// ----- testes
//
// Versao      Teste
// 0.1         01. ( OK )     teste inicial
//

```

04.) Copiar a versão atual do programa para outra (nova) – Guia0102.c.

- 05.) Editar mudanças no nome do programa e versão, conforme as indicações a seguir, tomando o cuidado de modificar todas as referências, inclusive as presentes em comentários. Incluir na documentação complementar as alterações feitas, acrescentar indicações de mudança de versão e prever novos testes.

```
/*
  Guia_0102 - v0.0. - __ / __ / ____
  Author: _____

  Para compilar em uma janela de comandos (terminal):

  No Linux   : gcc -o Guia0102 ./Guia0102.c
  No Windows: gcc -o Guia0102.exe Guia0102.c

  Para executar em uma janela de comandos (terminal):

  No Linux   : ./Guia0102.c
  No Windows: Guia0102

*/
// lista de dependencias
#include "karel.h" // na pasta do programa

// ----- definicoes de metodos

/**
  decorateWorld - Metodo para preparar o cenario.
  @param fileName - nome do arquivo para guardar a descricao.
*/
void decorateWorld ( char* fileName )
{
  // definir o contexto
  ref_world v_world = world_now;

  // colocar um marcador no mundo
  set_World ( 4, 4, BEEPER );

  // salvar a configuracao atual do mundo
  save_World( fileName );
} // decorateWorld ( )

/**
  * turnRight para virar 'a direita.
*/
void turnRight( )
{
  // o executor deste metodo
  // deve virar tres vezes 'a esquerda
  turnLeft( );
  turnLeft( );
  turnLeft( );
} // end turnRight( )
```

```

/**
 * doTask - Metodo para especificar parte de uma tarefa.
 */
void doTask( )
{
    move( );
    move( );
    turnLeft( );
    move( );
    move( );
    turnLeft( );
    move( );
    move( );
    turnLeft( );
    move( );
    move( );
    turnLeft( );
    turnLeft( );
    turnOff ( );
} // end doTask( )

// ----- acao principal

/**
 * Acao principal: executar a tarefa descrita acima.
 */

int main ( )
{
    // definir o contexto
    world v_world;      ref_world ref_v_world = ref v_world;      world_now = ref_v_world;
    robot v_robot;      ref_robot ref_v_robot = ref v_robot;      robot_now = ref_v_robot;
    box v_box ;         ref_box ref_v_box = ref v_box ;           box_now = ref_v_box ;

    // criar o mundo
    create_World ( "Guia_01_02_v01" );

    // decorar o ambiente com um marcador
    // OBS.: executar pelo menos uma vez,
    //      antes de qualquer outra coisa
    //      (depois de criado, podera' ser comentado)
    decorateWorld( "Guia0101.txt" );

    // comandos para tornar o mundo visivel
    reset_World( );      // limpar configuracoes
    set_Speed ( 1 );      // escolher velocidade
    read_World( "Guia0101.txt" ); // ler configuracao do ambiente

    // colocar o robo com a configuracao inicial
    create_Robot ( 1, 1, EAST, 0, "Karel" );

    // executar acoes
    doTask( );

    // preparar o encerramento
    close_World ( );

    // encerrar o programa
    getchar ( );
    return ( 0 );
} // end main ( )

```

```
// ----- testes

/*
----- documentacao complementar
----- notas / observacoes / comentarios

----- previsao de testes

----- historico

Versao      Data      Modificacao
0.1         _/_      esboco

----- testes

Versao      Teste
0.1         01. ( OK )      teste inicial
0.2         01. (   )      teste da tarefa

*/
```

- 06.) Compilar o programa novamente.
 Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
 Se não houver erros, seguir para o próximo passo.
- 07.) Executar o programa.
 Observar as saídas.
 Registrar os resultados.

```
// ----- testes
//
// Versao      Teste
// 0.1         01. ( OK )      teste inicial
// 0.2         01. ( OK )      teste da tarefa
//
```

- 08.) Copiar a versão atual do programa para outra (nova) – Guia0103.c.

09.) Acrescentar ao programa as modificações indicadas abaixo:

```
/*
Guia0103 - v0.0. - __ / __ / ____
Author: _____

Para compilar em uma janela de comandos (terminal):

No Linux   : gcc -o Guia0103 ./Guia0103.c
No Windows: gcc -o Guia0103.exe Guia0103.c

Para executar em uma janela de comandos (terminal):

No Linux   : ./Guia0103.c
No Windows: Guia0103

*/
// lista de dependencias
#include "karel.h"      // na pasta do programa

// ----- definicoes de metodos

/**
  decorateWorld - Metodo para preparar o cenario.
  @param fileName - nome do arquivo para guardar a descricao.
*/
void decorateWorld ( char* fileName )
{
  // colocar um marcador no mundo
  set_World ( 4, 4, BEEPER );

  // salvar a configuracao atual do mundo
  save_World( fileName );
} // decorateWorld ( )

/**
  * turnRight - Metodo para virar 'a direita.
*/
void turnRight( )
{
  // o executor deste metodo
  // deve virar tres vezes 'a esquerda
  turnLeft( );
  turnLeft( );
  turnLeft( );
} // end turnRight( )

/**
  * doPartialTask - Metodo para especificar parte de uma tarefa.
*/
void doPartialTask( )
{
  // especificar acoes dessa parte da tarefa
  move( );
  move( );
  move( );
  turnLeft( );
} // end doPartialTask( )
```

```

/**
 * doTask - Metodo para especificar toda a tarefa.
 */
void doTask( )
{
    // especificar acoes da tarefa
    doPartialTask( );
    doPartialTask( );
    doPartialTask( );
    doPartialTask( );
    turnLeft( );
    turnOff( );
} // end doTask( )

// ----- acao principal

/**
 * Acao principal: executar a tarefa descrita acima.
 */

int main ( )
{
    // definir o contexto
    world v_world;      ref_world ref_v_world = ref v_world;      world_now = ref_v_world;
    robot v_robot;      ref_robot ref_v_robot = ref v_robot;      robot_now = ref_v_robot;
    box v_box ;         ref_box ref_v_box = ref v_box ;           box_now = ref_v_box ;

    // criar o mundo
    create_World ( "Guia_01_03_v01" );

    // criar o ambiente com um marcador
    // OBS.: executar pelo menos uma vez,
    //        antes de qualquer outra coisa
    //        (depois de criado, podera' ser comentado)
    decorateWorld( "Guia0103.txt" );

    // comandos para tornar o mundo visivel
    reset_World( );      // limpar configuracoes
    set_Speed ( 1 );      // escolher velocidade
    read_World( "Guia0103.txt" ); // ler configuracao do ambiente

    // colocar o robo no necessario
    create_Robot ( 1, 1, EAST, 0, "Karel" );

    // executar acoes
    doTask ( );

    // preparar o encerramento
    close_World ( );

    // encerrar o programa
    getchar ( );
    return ( 0 );
} // end main ( )

```



```
// ----- testes

/*
----- documentacao complementar
----- notas / observacoes / comentarios

----- previsao de testes

----- historico

Versao      Data      Modificacao
0.1         _/_      esboco

----- testes

Versao      Teste
0.1         0.1 ( OK )      teste inicial
0.2         0.1 ( OK )      teste da tarefa
0.3         0.1 (   )      teste da tarefa parcial
*/
```

10.) Compilar o programa novamente.

Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.

Se não houver erros, seguir para o próximo passo.

11.) Executar o programa.

Observar as saídas.

Registrar os resultados.

```
// ----- testes
//
// Versao      Teste
// 0.1         01. ( OK )      teste inicial
// 0.2         01. ( OK )      teste da tarefa
// 0.3         01. ( OK )      teste da tarefa parcial
//
```

12.) Copiar a versão atual do programa para outra (nova) – Guia0104.c.

13.) Alterar as identificações de programa e acrescentar as modificações indicadas abaixo:

```
/**
/*
  Guia0104 - v0.0. - __ / __ / ____
  Author: _____

  Para compilar em uma janela de comandos (terminal):

  No Linux   : gcc -o Guia0104   ./Guia0104.c
  No Windows: gcc -o Guia0104.exe Guia0104.c

  Para executar em uma janela de comandos (terminal):

  No Linux   : ./Guia0104.c
  No Windows: Guia0104

*/
// lista de dependencias
#include "karel.h"    // na pasta do programa

// ----- definicoes de metodos

/**
  decorateWorld - Metodo para preparar o cenario.
  @param fileName - nome do arquivo para guardar a descricao.
*/
void decorateWorld ( char* fileName )
{
  // colocar um marcador no mundo
  set_World ( 4, 4, BEEPER );

  // salvar a configuracao atual do mundo
  save_World( fileName );

} // decorateWorld ( )

/**
  * turnRight - Metodo para virar 'a direita.
  */
void turnRight( )
{
  // o executor deste metodo
  // deve virar tres vezes 'a esquerda
  turnLeft( );
  turnLeft( );
  turnLeft( );
} // end turnRight( )
```

```

/**
 * doPartialTask - Metodo para especificar parte de uma tarefa.
 */
void doPartialTask( )
{
    // especificar acoes dessa parte da tarefa
    move( );
    move( );
    move( );
    turnLeft( );
} // end doPartialTask( )

/**
 * doTask - Metodo para especificar toda a tarefa.
 */
void doTask( )
{
    // especificar acoes da tarefa
    doPartialTask( );
    doPartialTask( );
    pickBeeper( ); // apanhar marcador
    doPartialTask( );
    doPartialTask( );
    turnLeft( );
    turnOff( );
} // end doTask( )

// ----- acao principal

/**
 * Acao principal: executar a tarefa descrita acima.
 */

int main ( )
{
    // definir o contexto
    world v_world;      ref_world ref_v_world = ref v_world;      world_now = ref_v_world;
    robot v_robot;      ref_robot ref_v_robot = ref v_robot;      robot_now = ref_v_robot;
    box v_box ;         ref_box ref_v_box = ref v_box ;           box_now = ref_v_box ;

    // criar o mundo
    create_World ( "Guia_01_04_v01" );

    // criar o ambiente com um marcador
    // OBS.: executar pelo menos uma vez,
    //        antes de qualquer outra coisa
    //        (depois de criado, podera' ser comentado)
    decorateWorld( "Guia0104.txt" );

    // comandos para tornar o mundo visivel
    reset_World( );      // limpar configuracoes
    set_Speed ( 1 );      // escolher velocidade
    read_World( "Guia0104.txt" ); // ler configuracao do ambiente

    // colocar o robo no necessario
    create_Robot ( 1, 1, EAST, 0, "Karel" );

    // executar acoes
    doTask ( );

```

```

// preparar o encerramento
close_World ( );

// encerrar o programa
getchar ( );
return ( 0 );
} // end main ( )

// ----- testes

/*
----- documentacao complementar

----- notas / observacoes / comentarios

----- previsao de testes

----- historico

Versao      Data      Modificacao
0.1         _/_/     esboco

----- testes

Versao      Teste
0.1         0.1 ( OK )   teste inicial
0.2         0.1 ( OK )   teste da tarefa
0.3         0.1 ( OK )   teste da tarefa parcial
0.4         0.1 (   )   teste do apanhar marcador

*/

```

- 14.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.
- 15.) Executar o programa.
Observar as saídas.
Registrar os resultados.

```

// ----- testes
//
// Versao      Teste
// 0.1         01. ( OK )   teste inicial
// 0.2         01. ( OK )   teste da tarefa
// 0.3         01. ( OK )   teste da tarefa parcial
// 0.4         01. ( OK )   teste do apanhar marcador
//

```

- 16.) Copiar a versão atual do programa para outra (nova) – Guia0105.c.

17.) Alterar as identificações de programa e acrescentar as modificações indicadas abaixo:

```
/**
/*
  Guia0105 - v0.0. - __ / __ / ____
  Author: _____

  Para compilar em uma janela de comandos (terminal):

  No Linux   : gcc -o Guia0105   ./Guia0105.c
  No Windows: gcc -o Guia0105.exe Guia0105.c

  Para executar em uma janela de comandos (terminal):

  No Linux   : ./Guia0105.c
  No Windows: Guia0105

*/
// lista de dependencias
#include "karel.h"    // na pasta do programa

// ----- definicoes de metodos

/**
  decorateWorld - Metodo para preparar o cenario.
  @param fileName - nome do arquivo para guardar a descricao.
*/
void decorateWorld ( char* fileName )
{
  // colocar um marcador no mundo
  set_World ( 4, 4, BEEPER );

  // salvar a configuracao atual do mundo
  save_World( fileName );

} // decorateWorld ( )

/**
  * turnRight - Metodo para virar 'a direita.
*/
void turnRight( )
{
  // o executor deste metodo
  // deve virar tres vezes 'a esquerda
  turnLeft( );
  turnLeft( );
  turnLeft( );
} // end turnRight( )
```

```

/**
 * doPartialTask - Metodo para especificar parte de uma tarefa.
 */
void doPartialTask( )
{
    // especificar acoes dessa parte da tarefa
    move( );
    move( );
    move( );
    turnLeft( );
} // end doPartialTask( )

/**
 * doTask - Metodo para especificar toda a tarefa.
 */
void doTask( )
{
    // especificar acoes da tarefa
    doPartialTask( );
    doPartialTask( );
    pickBeeper( ); // apanhar marcador
    doPartialTask( );
    putBeeper( ); // colocar marcador
    doPartialTask( );
    turnLeft( );
    turnOff( );
} // end doTask( )

// ----- acao principal

/**
 * Acao principal: executar a tarefa descrita acima.
 */

int main ( )
{
    // definir o contexto
    world v_world;    ref_world ref_v_world = ref v_world;    world_now = ref_v_world;
    robot v_robot;    ref_robot ref_v_robot = ref v_robot;    robot_now = ref_v_robot;
    box v_box ;    ref_box ref_v_box = ref v_box ;    box_now = ref_v_box ;

    // criar o mundo
    create_World ( "Guia_01_05_v01" );

    // criar o ambiente com um marcador
    // OBS.: executar pelo menos uma vez,
    // antes de qualquer outra coisa
    // (depois de criado, podera' ser comentado)
    decorateWorld( "Guia0105.txt" );

    // comandos para tornar o mundo visivel
    reset_World( ); // limpar configuracoes
    set_Speed ( 1 ); // escolher velocidade
    read_World( "Guia0105.txt" ); // ler configuracao do ambiente

    // colocar o robo no necessario
    create_Robot ( 1, 1, EAST, 0, "Karel" );

    // executar acoes

```

```

doTask ( );

// preparar o encerramento
close_World ( );

// encerrar o programa
getchar ( );
return ( 0 );
} // end main ( )

// ----- testes

/*
----- documentacao complementar

----- notas / observacoes / comentarios

----- previsao de testes

----- historico

Versao      Data      Modificacao
0.1         _/_/___    esboco

----- testes

Versao      Teste
0.1         0.1 ( OK )    teste inicial
0.2         0.1 ( OK )    teste da tarefa
0.3         0.1 ( OK )    teste da tarefa parcial
0.4         0.1 ( OK )    teste do apanhar marcador
0.5         0.1 (   )    teste do colocar marcador

*/

```

18.) Compilar o programa novamente.

Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.

Se não houver erros, seguir para o próximo passo.

19.) Executar o programa.

Observar as saídas.

Registrar os resultados.

```

// ----- testes
//
// Versao      Teste
// 0.1         01. ( OK )    teste inicial
// 0.2         01. ( OK )    teste da tarefa
// 0.3         01. ( OK )    teste da tarefa parcial
// 0.4         01. ( OK )    teste do apanhar marcador
// 0.5         01. ( OK )    teste do colocar marcador
//

```

20.) Copiar a versão atual do programa para outra (nova) – Guia0106.c.

21.) Alterar as identificações de programa e acrescentar as modificações indicadas abaixo:

```
/**
/*
Guia0106 - v0.0. - __/ __/ ____
Author: _____

Para compilar em uma janela de comandos (terminal):

No Linux   : gcc -o Guia0106   ./Guia0106.c
No Windows: gcc -o Guia0106.exe Guia0106.c

Para executar em uma janela de comandos (terminal):

No Linux   : ./Guia0106.c
No Windows:  Guia0106

*/
// lista de dependencias
#include "karel.h"      // na pasta do programa

// ----- definicoes de metodos

/**
    decorateWorld - Metodo para preparar o cenario.
    @param fileName - nome do arquivo para guardar a descricao.
*/
void decorateWorld ( char* fileName )
{
    // colocar um marcador no mundo
    set_World ( 4, 4, BEEPER );

    // salvar a configuracao atual do mundo
    save_World( fileName );

} // decorateWorld ( )

/**
    * turnRight - Metodo para virar 'a direita.
*/
void turnRight( )
{
    // o executor deste metodo
    // deve virar tres vezes 'a esquerda
    turnLeft( );
    turnLeft( );
    turnLeft( );
} // end turnRight( )
```



```

/**
 * moveN - Metodo para mover certa quantidade de passos.
 * @param steps - passos a serem dados.
 */
void moveN( int steps )
{
    // testar se a quantidade de passos e' maior que zero
    if ( steps > 0 )
    {
        // dar um passo
        move( );
        // tentar fazer de novo, com menos um passo
        moveN ( steps - 1 );
    } // end if
} // end moveN( )

/**
 * doPartialTask - Metodo para especificar parte de uma tarefa.
 */
void doPartialTask( )
{
    // especificar acoes dessa parte da tarefa
    moveN( 3 );
    turnLeft( );
} // end doPartialTask( )

/**
 * doTask - Metodo para especificar toda a tarefa.
 */
void doTask( )
{
    // especificar acoes da tarefa
    doPartialTask( );
    doPartialTask( );
    pickBeeper( ); // apanhar marcador
    doPartialTask( );
    putBeeper( ); // colocar marcador
    doPartialTask( );
    turnLeft( );
    turnOff ( );
} // end doTask( )

```

```

// ----- acao principal

/**
 * Acao principal: executar a tarefa descrita acima.
 */

int main ( )
{
    // definir o contexto
    world v_world;      ref_world ref_v_world = ref v_world;      world_now = ref_v_world;
    robot v_robot;      ref_robot ref_v_robot = ref v_robot;      robot_now = ref_v_robot;
    box v_box ;         ref_box ref_v_box = ref v_box ;           box_now = ref_v_box ;

    // criar o mundo
    create_World ( "Guia_01_06_v01" );

    // criar o ambiente com um marcador
    // OBS.: executar pelo menos uma vez,
    //      antes de qualquer outra coisa
    //      (depois de criado, podera' ser comentado)
    decorateWorld( "Guia0106.txt" );

    // comandos para tornar o mundo visivel
    reset_World();      // limpar configuracoes
    set_Speed ( 1 );     // escolher velocidade
    read_World( "Guia0106.txt" ); // ler configuracao do ambiente

    // colocar o robo no necessario
    create_Robot ( 1, 1, EAST, 0, "Karel" );

    // executar acoes
    doTask ( );

    // preparar o encerramento
    close_World ( );

    // encerrar o programa
    getchar ( );
    return ( 0 );
} // end main ( )

```

```
// ----- testes

/*
----- documentacao complementar
----- notas / observacoes / comentarios

----- previsao de testes

----- historico

Versao      Data      Modificacao
0.1         _/_      esboco

----- testes

Versao      Teste
0.1         0.1 ( OK )      teste inicial
0.2         0.1 ( OK )      teste da tarefa
0.3         0.1 ( OK )      teste da tarefa parcial
0.4         0.1 ( OK )      teste do apanhar marcador
0.5         0.1 ( OK )      teste do colocar marcador
0.6         01. ( )      teste da repeticao do movimento
*/
```

- 22.) Compilar o programa novamente.
 Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
 Se não houver erros, seguir para o próximo passo.
- 23.) Executar o programa.
 Observar as saídas.
 Registrar os resultados.

```
// ----- testes
//
// Versao      Teste
// 0.1         01. ( OK )      teste inicial
// 0.2         01. ( OK )      teste da tarefa
// 0.3         01. ( OK )      teste da tarefa parcial
// 0.4         01. ( OK )      teste do apanhar marcador
// 0.5         01. ( OK )      teste do colocar marcador
// 0.6         01. ( OK )      teste da repeticao do movimento
//
```

- 24.) Copiar a versão atual do programa para outra (nova) – Guia0107.c.

25.) Alterar as identificações de programa e acrescentar as modificações indicadas abaixo:

```
/**
/*
Guia0107 - v0.0. - __ / __ / ____
Author: _____

Para compilar em uma janela de comandos (terminal):

No Linux   : gcc -o Guia0107   ./Guia0107.c
No Windows: gcc -o Guia0107.exe Guia0107.c

Para executar em uma janela de comandos (terminal):

No Linux   : ./Guia0107.c
No Windows: Guia0107

*/
// lista de dependencias
#include "karel.h"    // na pasta do programa

// ----- definicoes de metodos

/**
    decorateWorld - Metodo para preparar o cenario.
    @param fileName - nome do arquivo para guardar a descricao.
*/
void decorateWorld ( char* fileName )
{
    // colocar um marcador no mundo
    set_World ( 4, 4, BEEPER );

    // salvar a configuracao atual do mundo
    save_World( fileName );

} // decorateWorld ( )

/**
    * turnRight - Metodo para virar 'a direita.
*/
void turnRight( )
{
    // o executor deste metodo
    // deve virar tres vezes 'a esquerda
    turnLeft( );
    turnLeft( );
    turnLeft( );
} // end turnRight( )
```

```

/**
 * moveN - Metodo para mover certa quantidade de passos.
 * @param steps - passos a serem dados.
 */
void moveN( int steps )
{
    // testar se a quantidade de passos e' maior que zero
    if ( steps > 0 )
    {
        // dar um passo
        move( );
        // tentar fazer de novo, com menos um passo
        moveN ( steps - 1 );
    } // end if
} // end moveN( )

/**
 * doPartialTask - Metodo para especificar parte de uma tarefa.
 */
void doPartialTask( )
{
    // especificar acoes dessa parte da tarefa
    moveN( 3 );
    turnLeft( );
} // end doPartialTask( )

/**
 * doTask - Metodo para especificar toda a tarefa.
 */
void doTask( )
{
    // especificar acoes da tarefa
    doPartialTask( );
    doPartialTask( );
    // testar se ha' marcador antes ...
    if ( nextToABeeper( ) )
    {
        // ... de tentar carrega-lo
        pickBeeper( );
    } // end if
    doPartialTask( );
    // testar se carrega marcador antes ...
    if ( beepersInBag( ) )
    {
        // ... de tentar descarrega-lo
        putBeeper( );
    } // end if
    doPartialTask( );
    turnLeft( );
    turnOff( );
} // end doTask( )

```

```

// ----- acao principal

/**
 * Acao principal: executar a tarefa descrita acima.
 */

int main ( )
{
    // definir o contexto
    world v_world;      ref_world ref_v_world = ref v_world;      world_now = ref_v_world;
    robot v_robot;      ref_robot ref_v_robot = ref v_robot;      robot_now = ref_v_robot;
    box v_box ;         ref_box ref_v_box = ref v_box ;           box_now = ref_v_box ;

    // criar o mundo
    create_World ( "Guia_01_07_v01" );

    // criar o ambiente com um marcador
    // OBS.: executar pelo menos uma vez,
    //      antes de qualquer outra coisa
    //      (depois de criado, podera' ser comentado)
    decorateWorld( "Guia0107.txt" );

    // comandos para tornar o mundo visivel
    reset_World();      // limpar configuracoes
    set_Speed ( 1 );     // escolher velocidade
    read_World( "Guia0107.txt" ); // ler configuracao do ambiente

    // colocar o robo no necessario
    create_Robot ( 1, 1, EAST, 0, "Karel" );

    // executar acoes
    doTask ( );

    // preparar o encerramento
    close_World ( );

    // encerrar o programa
    getchar ( );
    return ( 0 );
} // end main ( )

```

```
// ----- testes

/*
----- documentacao complementar
----- notas / observacoes / comentarios

----- previsao de testes

----- historico

Versao      Data      Modificacao
0.1         _/_      esboco

----- testes

Versao      Teste
0.1         0.1 ( OK )      teste inicial
0.2         0.1 ( OK )      teste da tarefa
0.3         0.1 ( OK )      teste da tarefa parcial
0.4         0.1 ( OK )      teste do apanhar marcador
0.5         0.1 ( OK )      teste do colocar marcador
0.6         01. ( OK )      teste da repeticao do movimento
0.7         01. (   )      teste com marcador na posicao (4,4)
0.7         02. (   )      teste sem marcador na posicao (4,4)

*/
```

26.) Compilar o programa novamente.
 Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
 Se não houver erros, seguir para o próximo passo.

27.) Executar o programa.
 Observar as saídas.
 Registrar os resultados.

```
// ----- testes
//
// Versao      Teste
// 0.1         01. ( OK )      teste inicial
// 0.2         01. ( OK )      teste da tarefa
// 0.3         01. ( OK )      teste da tarefa parcial
// 0.4         01. ( OK )      teste do apanhar marcador
// 0.5         01. ( OK )      teste do colocar marcador
// 0.6         01. ( OK )      teste da repeticao do movimento
// 0.7         01. ( OK )      teste com marcador na posicao (4,4)
//             02. ( OK )      teste com marcador na posicao (4,4)
//
```

28.) Copiar a versão atual do programa para outra (nova) – Guia0008.c.

29.) Alterar as identificações de programa e acrescentar as modificações indicadas abaixo:

```

/*
Guia0108 - v0.0. - __ / __ / ____
Author: _____

Para compilar em uma janela de comandos (terminal):

No Linux   : gcc -o Guia0108   ./Guia0108.c
No Windows: gcc -o Guia0108.exe Guia0108.c

Para executar em uma janela de comandos (terminal):

No Linux   : ./Guia0108.c
No Windows: Guia0108

*/
// lista de dependencias
#include "karel.h"    // na pasta do programa

// ----- definicoes de metodos

/**
  decorateWorld - Metodo para preparar o cenario.
  @param fileName - nome do arquivo para guardar a descricao.
*/
void decorateWorld ( char* fileName )
{
  // colocar um marcador no mundo
  set_World ( 4, 4, BEEPER );

  // salvar a configuracao atual do mundo
  save_World( fileName );
} // decorateWorld ( )

/**
  * turnRight - Metodo para virar 'a direita.
*/
void turnRight( )
{
  // o executor deste metodo
  // deve virar tres vezes 'a esquerda
  turnLeft( );
  turnLeft( );
  turnLeft( );
} // end turnRight( )

```



```

/**
 * moveN - Metodo para mover certa quantidade de passos.
 * @param steps - passos a serem dados.
 */
void moveN( int steps )
{
    // testar se a quantidade de passos e' maior que zero
    if ( steps > 0 )
    {
        // dar um passo
        move( );
        // tentar fazer de novo, com menos um passo
        moveN ( steps - 1 );
    } // end if
} // end moveN( )

/**
 * doPartialTask - Metodo para especificar parte de uma tarefa.
 */
void doPartialTask( )
{
    // especificar acoes dessa parte da tarefa
    moveN( 3 );
    turnLeft( );
} // end doPartialTask( )

/**
 * doTask - Metodo para especificar toda a tarefa.
 */
void doTask( )
{
    // especificar acoes da tarefa
    doPartialTask( );
    doPartialTask( );
    // testar se ha' marcador antes ...
    if ( nextToABeeper( ) )
    {
        // ... de tentar carrega-lo
        pickBeeper( );
    } // end if
    doPartialTask( );
    // testar se carrega marcador antes ...
    // (forma alternativa)
    if ( beepers( ) > 0 )
    {
        // ... de tentar descarrega-lo
        putBeeper( );
    } // end if
    doPartialTask( );
    turnLeft( );
    turnOff( );
} // end doTask( )

```

```

// ----- acao principal

/**
 * Acao principal: executar a tarefa descrita acima.
 */

int main ( )
{
    // definir o contexto
    world v_world;      ref_world ref_v_world = ref v_world;      world_now = ref_v_world;
    robot v_robot;      ref_robot ref_v_robot = ref v_robot;      robot_now = ref_v_robot;
    box v_box ;         ref_box ref_v_box = ref v_box ;           box_now = ref_v_box ;

    // criar o mundo
    create_World ( "Guia_01_08_v01" );

    // criar o ambiente com um marcador
    // OBS.: executar pelo menos uma vez,
    //        antes de qualquer outra coisa
    //        (depois de criado, podera' ser comentado)
    decorateWorld( "Guia0108.txt" );

    // comandos para tornar o mundo visivel
    reset_World( );      // limpar configuracoes
    set_Speed ( 1 );      // escolher velocidade
    read_World( "Guia0108.txt" ); // ler configuracao do ambiente

    // colocar o robo no necessario
    create_Robot ( 1, 1, EAST, 0, "Karel" );

    // executar acoes
    doTask ( );

    // preparar o encerramento
    close_World ( );

    // encerrar o programa
    getchar ( );
    return ( 0 );
} // end main ( )

```

```
// ----- testes

/*
----- documentacao complementar
----- notas / observacoes / comentarios
```

```
----- previsao de testes
```

```
----- historico
```

Versao	Data	Modificacao
0.1	__/__/__	esboco

```
----- testes
```

Versao	Teste	
0.1	0.1 (OK)	teste inicial
0.2	0.1 (OK)	teste da tarefa
0.3	0.1 (OK)	teste da tarefa parcial
0.4	0.1 (OK)	teste do apanhar marcador
0.5	0.1 (OK)	teste do colocar marcador
0.6	01. (OK)	teste da repeticao do movimento
0.7	01. (OK)	teste com marcador na posicao (4,4)
	02. (OK)	teste sem marcador na posicao (4,4)
0.8	01. ()	teste com a quantidade de marcadores

```
*/
```

30.) Compilar o programa novamente.

Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.

Se não houver erros, seguir para o próximo passo.

31.) Executar o programa.

Observar as saídas.

Registrar os resultados.

```
// ----- testes
//
// Versao    Teste
// 0.1       01. ( OK )      teste inicial
// 0.2       01. ( OK )      teste da tarefa
// 0.3       01. ( OK )      teste da tarefa parcial
// 0.4       01. ( OK )      teste do apanhar marcador
// 0.5       01. ( OK )      teste do colocar marcador
// 0.6       01. ( OK )      teste da repeticao do movimento
// 0.7       01. ( OK )      teste com marcador na posicao (4,4)
//           02. ( OK )      teste com marcador na posicao (4,4)
// 0.8       01. ( OK )      teste com a quantidade de marcadores
//
```

32.) Copiar a versão atual do programa para outra (nova) – Guia0109.c.

33.) Alterar as identificações de programa e acrescentar as modificações indicadas abaixo:

```
/*
Guia0109 - v0.0. - __ / __ / ____
Author: _____
```

Para compilar em uma janela de comandos (terminal):

```
No Linux   : gcc -o Guia0109   ./Guia0109.c
No Windows: gcc -o Guia0109.exe Guia0109.c
```

Para executar em uma janela de comandos (terminal):

```
No Linux   : ./Guia0109.c
No Windows:  Guia0109
```

```
*/
// lista de dependencias
#include "karel.h"    // na pasta do programa

// ----- definicoes de metodos

/**
  decorateWorld - Metodo para preparar o cenario.
  @param fileName - nome do arquivo para guardar a descricao.
*/
void decorateWorld ( char* fileName )
{
  // colocar um marcador no mundo
  set_World ( 4, 4, BEEPER );

  // salvar a configuracao atual do mundo
  save_World( fileName );
} // decorateWorld ( )

/**
  * turnRight - Metodo para virar 'a direita.
  */
void turnRight( )
{
  // o executor deste metodo
  // deve virar tres vezes 'a esquerda
  turnLeft( );
  turnLeft( );
  turnLeft( );
} // end turnRight( )
```

```

/**
 * moveN - Metodo para mover certa quantidade de passos.
 * @param steps - passos a serem dados.
 */
void moveN( int steps )
{
    // testar se a quantidade de passos e' maior que zero
    while ( steps > 0 )
    {
        // dar um passo
        move( );
        // tentar fazer de novo, com menos um passo
        steps = steps - 1;
    } // end if
} // end moveN( )

/**
 * doPartialTask - Metodo para especificar parte de uma tarefa.
 */
void doPartialTask( )
{
    // especificar acoes dessa parte da tarefa
    moveN( 3 );
    turnLeft( );
} // end doPartialTask( )

/**
 * doTask - Metodo para especificar toda a tarefa.
 */
void doTask( )
{
    // especificar acoes da tarefa
    doPartialTask( );
    doPartialTask( );
    // testar se ha' marcador antes ...
    if ( nextToABeeper( ) )
    {
        // ... de tentar carrega-lo
        pickBeeper( );
    } // end if
    doPartialTask( );
    // testar se carrega marcador antes ...
    // (forma alternativa)
    if ( beepers( ) > 0 )
    {
        // ... de tentar descarrega-lo
        putBeeper( );
    } // end if
    doPartialTask( );
    turnLeft( );
    turnOff( );
} // end doTask( )

```

```

// ----- acao principal

/**
 * Acao principal: executar a tarefa descrita acima.
 */

int main ( )
{
    // definir o contexto
    world v_world;      ref_world ref_v_world = ref v_world;      world_now = ref_v_world;
    robot v_robot;      ref_robot ref_v_robot = ref v_robot;      robot_now = ref_v_robot;
    box v_box ;         ref_box ref_v_box = ref v_box ;           box_now = ref_v_box ;

    // criar o mundo
    create_World ( "Guia_01_09_v01" );

    // criar o ambiente com um marcador
    // OBS.: executar pelo menos uma vez,
    //      antes de qualquer outra coisa
    //      (depois de criado, podera' ser comentado)
    decorateWorld( "Guia0109.txt" );

    // comandos para tornar o mundo visivel
    reset_World();      // limpar configuracoes
    set_Speed ( 1 );     // escolher velocidade
    read_World( "Guia0109.txt" ); // ler configuracao do ambiente

    // colocar o robo no necessario
    create_Robot ( 1, 1, EAST, 0, "Karel" );

    // executar acoes
    doTask ( );

    // preparar o encerramento
    close_World ( );

    // encerrar o programa
    getchar ( );
    return ( 0 );
} // end main ( )

```

```
// ----- testes

/*
----- documentacao complementar
----- notas / observacoes / comentarios

----- previsao de testes

----- historico

Versao      Data      Modificacao
0.1         _/_      esboco

----- testes

Versao      Teste
0.1         0.1 ( OK )      teste inicial
0.2         0.1 ( OK )      teste da tarefa
0.3         0.1 ( OK )      teste da tarefa parcial
0.4         0.1 ( OK )      teste do apanhar marcador
0.5         0.1 ( OK )      teste do colocar marcador
0.6         01. ( OK )      teste da repeticao do movimento
0.7         01. ( OK )      teste com marcador na posicao (4,4)
0.7         02. ( OK )      teste sem marcador na posicao (4,4)
0.8         01. ( OK )      teste com a quantidade de marcadores
0.9         01. ( )        teste com outra forma de repeticao
*/
```

34.) Compilar o programa novamente.

Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.

Se não houver erros, seguir para o próximo passo.

35.) Executar o programa.

Observar as saídas.

Registrar os resultados.

```
// ----- testes
//
// Versao      Teste
// 0.1         01. ( OK )      teste inicial
// 0.2         01. ( OK )      teste da tarefa
// 0.3         01. ( OK )      teste da tarefa parcial
// 0.4         01. ( OK )      teste do apanhar marcador
// 0.5         01. ( OK )      teste do colocar marcador
// 0.6         01. ( OK )      teste da repeticao do movimento
// 0.7         01. ( OK )      teste com marcador na posicao (4,4)
//             02. ( OK )      teste com marcador na posicao (4,4)
// 0.8         01. ( OK )      teste com a quantidade de marcadores
// 0.9         01. ( OK )      teste com outra forma de repeticao
//
```

36.) Copiar a versão atual do programa para outra (nova) – Guia0110.c.

37.) Alterar as identificações de programa e acrescentar as modificações indicadas abaixo:

```
/*
  Guia0110 - v0.0. - __ / __ / ____
  Author: _____

  Para compilar em uma janela de comandos (terminal):

  No Linux   : gcc -o Guia0110   ./Guia0110.c
  No Windows: gcc -o Guia0110.exe Guia0110.c

  Para executar em uma janela de comandos (terminal):

  No Linux   : ./Guia0110.c
  No Windows: Guia0110

*/
// lista de dependencias
#include "karel.h"    // na pasta do programa

// ----- definicoes de metodos

/**
  decorateWorld - Metodo para preparar o cenario.
  @param fileName - nome do arquivo para guardar a descricao.
*/
void decorateWorld ( char* fileName )
{
  // colocar um marcador no mundo
  set_World ( 4, 4, BEEPER );

  // salvar a configuracao atual do mundo
  save_World( fileName );
} // decorateWorld ( )

/**
  * turnRight - Metodo para virar 'a direita.
*/
void turnRight( )
{
  // o executor deste metodo
  // deve virar tres vezes 'a esquerda
  turnLeft( );
  turnLeft( );
  turnLeft( );
} // end turnRight( )
```



```

/**
 * moveN - Metodo para mover certa quantidade de passos.
 * @param steps - passos a serem dados.
 */
void moveN( int steps )
{
    // definir dado local
    int step = 0;
    // testar se a quantidade de passos e' maior que zero
    for ( step = 1; step < steps; step = step + 1 )
    {
        // dar um passo
        move( );
    } // end if
} // end moveN( )

/**
 * doPartialTask - Metodo para especificar parte de uma tarefa.
 */
void doPartialTask( )
{
    // especificar acoes dessa parte da tarefa
    moveN( 3 );
    turnLeft( );
} // end doPartialTask( )

/**
 * doTask - Metodo para especificar toda a tarefa.
 */
void doTask( )
{
    // especificar acoes da tarefa
    doPartialTask( );
    doPartialTask( );
    if ( nextToABeeper( ) )
    {
        // ... de tentar carrega-lo
        pickBeeper( );
        doPartialTask( );
        // ... e tentar descarrega-lo
        putBeeper( );
    }
    else
    {
        // continuar para concluir a tarefa
        doPartialTask( );
    } // end if
    doPartialTask( );
    turnLeft( );
    turnOff( );
} // end doTask( )

```

```

// ----- acao principal

/**
 * Acao principal: executar a tarefa descrita acima.
 */

int main ( )
{
    // definir o contexto
    world v_world;      ref_world ref_v_world = ref v_world;      world_now = ref_v_world;
    robot v_robot;      ref_robot ref_v_robot = ref v_robot;      robot_now = ref_v_robot;
    box v_box ;         ref_box ref_v_box = ref v_box ;          box_now = ref_v_box ;

    // criar o mundo
    create_World ( "Guia_01_10_v01" );

    // criar o ambiente com um marcador
    // OBS.: executar pelo menos uma vez,
    //      antes de qualquer outra coisa
    //      (depois de criado, podera' ser comentado)
    decorateWorld( "Guia0110.txt" );

    // comandos para tornar o mundo visivel
    reset_World();      // limpar configuracoes
    set_Speed ( 1 );     // escolher velocidade
    read_World( "Guia0110.txt" ); // ler configuracao do ambiente

    // colocar o robo no necessario
    create_Robot ( 1, 1, EAST, 0, "Karel" );

    // executar acoes
    doTask ( );

    // preparar o encerramento
    close_World ( );

    // encerrar o programa
    getchar ( );
    return ( 0 );
} // end main ( )

```

// ----- testes

/*

----- documentacao complementar

----- notas / observacoes / comentarios

----- previsao de testes

----- historico

Versao	Data	Modificacao
0.1	__/__/__	esboco

----- testes

Versao	Teste	
0.1	0.1 (OK)	teste inicial
0.2	0.1 (OK)	teste da tarefa
0.3	0.1 (OK)	teste da tarefa parcial
0.4	0.1 (OK)	teste do apanhar marcador
0.5	0.1 (OK)	teste do colocar marcador
0.6	01. (OK)	teste da repeticao do movimento
0.7	01. (OK)	teste com marcador na posicao (4,4)
	02. (OK)	teste sem marcador na posicao (4,4)
0.8	01. (OK)	teste com a quantidade de marcadores
0.9	01. (OK)	teste com outra forma de repeticao
1.0	01. ()	teste com outra forma de repeticao
	01. ()	teste com outra forma de alternativa

*/

38.) Compilar o programa novamente.

Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.

Se não houver erros, seguir para o próximo passo.

39.) Executar o programa.

Observar as saídas.

Registrar os resultados.

// ----- testes

//

// Versao	Teste	
// 0.1	01. (OK)	teste inicial
// 0.2	01. (OK)	teste da tarefa
// 0.3	01. (OK)	teste da tarefa parcial
// 0.4	01. (OK)	teste do apanhar marcador
// 0.5	01. (OK)	teste do colocar marcador
// 0.6	01. (OK)	teste da repeticao do movimento
// 0.7	01. (OK)	teste com marcador na posicao (4,4)
//	02. (OK)	teste com marcador na posicao (4,4)
// 0.8	01. (OK)	teste com a quantidade de marcadores
// 0.9	01. (OK)	teste com outra forma de repeticao
// 1.0	01. (OK)	teste com outra forma de alternativa
//		

Exercícios:

DICAS GERAIS: Consultar o Anexo C para mais informações e outros exemplos.

Prever, realizar e registrar todos os testes efetuados.

Fazer um programa para atender a cada uma das situações abaixo envolvendo definições e ações básicas.

Os programas deverão ser desenvolvidos em C usando as bibliotecas indicadas.

01.) Definir um conjunto de ações em um programa Guia0111 para:

- o robô partir da posição inicial (coluna=1, linha=1), voltado para leste, com três marcadores ("beepers");
- o robô deverá colocar um marcador nas posições indicadas: (3,3), (6,3) e (6,6), nessa ordem;
- retornar à posição inicial, voltar-se para o leste. e desligar-se.

OBS.: Para fazer o robô começar com marcadores, rever sua definição inicial:

```
create_Robot ( 1, 1, EAST, 0, "Karel" );
```

02.) Definir um conjunto de ações em um programa Guia0112 para:

- configurar o mundo para conter inicialmente três marcadores ("beepers") nas posições anteriormente indicadas: (3,3), (6,3) e (6,6) nessa ordem;
- o robô deverá partir da posição inicial (coluna=1, linha=1), voltado para leste e nenhum marcador;
- buscar os marcadores nas posições indicadas, na ordem inversa à qual foram colocados;
- retornar à posição inicial, voltar-se para o leste e desligar-se.

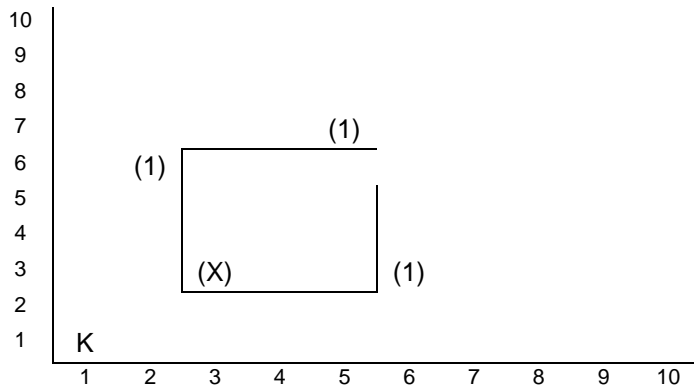
OBS.: Para colocar desde o início os marcadores nas posições indicadas, rever o método *decorateworld (filename)*:

03.) Definir um conjunto de ações em um programa Guia0113 para:

- o robô deverá partir da posição (coluna=1, linha=1), voltado para leste e sem marcadores
- buscar os marcadores ("beepers") nas mesmas posições iniciais do problema anterior (configurar o mundo com marcadores nas posições)
- descarregar todos os marcadores obtidos nas posições (4,1); (3,1) e (2,1), respectivamente;
- retornar à posição inicial, voltar-se para leste e desligar-se.

04.) Definir um conjunto de ações em um programa Guia0114 para:

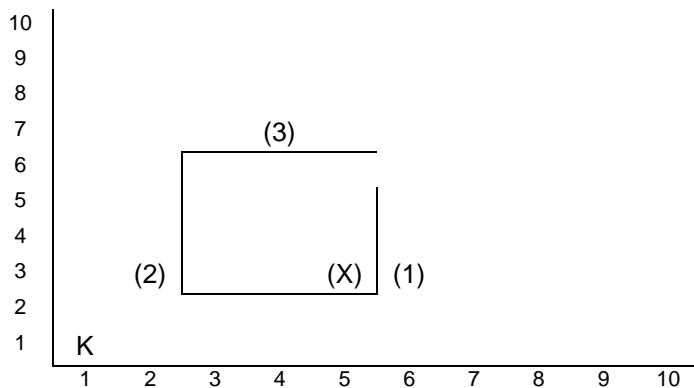
- configurar o mundo semelhante ao diagrama abaixo inicialmente com três marcadores nas posições indicadas (1):



- o robô deverá partir da posição inicial (coluna=1, linha=1), voltado para leste e com nenhum marcador;
- buscar os três marcadores ("beepers") e colocá-los na posição indicada por (X);
- retornar à posição inicial, voltar-se para o leste e desligar-se.

05.) Definir um conjunto de ações em um programa Guia0115 para:

- configurar o mundo semelhante ao diagrama abaixo inicialmente com seis marcadores na posição indicada (X):



- o robô deverá partir da posição inicial (coluna=1, linha=1), voltado para leste e com nenhum marcador;
- buscar os marcadores e distribuí-los nas quantidades indicadas e na ordem crescente das quantidades (1-2-3)
- retornar à posição inicial, voltar-se para o leste e desligar-se.

Tarefas extras

- E1.) Definir um conjunto de ações em um programa Guia01E1 para que se possa colocar e pegar vários marcadores (n) de uma só vez, se estiver carregando o suficiente.

```
public void putBeepers ( int n )
{
    // incluir comandos extras
} // end putBeepers ( )

public void pickBeepers ( int n )
{
    // incluir comandos extras
} // end putBeepers ( )
```

Testar o método mediante substituição de vários usos consecutivos de chamadas aos métodos de posicionamento unitário, por chamadas desses novos métodos.

DICA: Rever a definição do exercício anterior para usar repetições.

- E2.) Redefinir as repetições em um programa Guia01E2 para usar variações crescentes ao invés de decrescentes.

- E3.) Definir um conjunto de ações em um programa Guia01E3 para reunir em um novo arquivo myKarel.h todas as novas definições feitas, que poderão ser reaproveitadas no futuro, como o moveN(), putBeepers(), pickBeepers(), e substituir a dependência por apenas essa.

```
#include "myKarel.h"
```

Testar todos os métodos sob essa nova organização.

DICA: Os nomes deverão ser únicos, portanto, se desejar guardar métodos que realizem a mesma função, alterar o nome de acordo.

Atividade suplementar

Associar os conceitos de representações de dados e a metodologia sugerida para o desenvolvimento de programa (passo a passo), para modificar o modelo proposto (e exemplos associados) e introduzir, pouco a pouco, as modificações necessárias, cuidando de realizar a documentação das definições, procedimentos e operações executadas.

Para pensar a respeito

Qual a estratégia de solução ?

Como definir uma classe com um método principal que execute essa estratégia ?

Serão necessárias definições prévias (extras) para se obter o resultado ?

Como dividir os passos a serem feitos e organizá-los em que ordem ?

Que informações deverão ser colocadas na documentação ?

Como lidar com os erros de compilação ?

Como lidar com os erros de execução ?

Fontes de informação

apostila de C (anexos)

exemplos (0-9) na pasta de arquivos relacionada

bibliografia recomendada

lista de discussão da disciplina

websites

Processo

1 relacionar claramente seus objetivos e registrar isso na documentação necessária para o desenvolvimento;

2 organizar as informações de cada proposição de problema:

2.1 escolher os armazenadores de acordo com o tipo apropriado;

2.2 realizar as entradas de dados ou definições iniciais;

2.3 realizar as operações;

2.4 realizar as saídas dos resultados;

2.5 projetar testes para cada operação, considerar casos especiais

3 especificar a classe:

- 3.1 definir a identificação do programa na documentação;
- 3.2 definir a identificação do programador na documentação;
- 3.3 definir armazenadores necessários (se houver)
- 3.4 definir a entrada de dados para cada valor
- 3.5 testar se os dados foram armazenados corretamente
- 3.6 definir a saída de cada resultado ou (execução de cada ação)
- 3.7 testar a saída de cada resultado com valores (situações) conhecidas
- 3.8 definir cada operação
- 3.9 testar isoladamente cada operação, conferindo os resultados

4 especificar as ações da parte principal:

- 4.1 definir o cabeçalho para identificação;
- 4.2 definir as constantes, armazenadores e dados auxiliares (se houver);
- 4.3 definir a estrutura básica de programa que possa permitir a execução de vários dos testes programados;

5. realizar os testes isolados de cada operação e depois os testes de integração;

5.1 registrar todos os testes realizados.

Dicas

- Digitar os exemplos fornecidos e testá-los.
- Identificar exemplos que possam servir de modelos para os exercícios, e usá-los como sugestões para o desenvolvimento.
- Fazer rascunhos, diagramas e esquemas para orientar o desenvolvimento da solução, previamente, antes de começar a digitar o novo programa.
- Consultar os modelos de programas e documentação disponíveis.
- Anotar os testes realizados e seus resultados no final do texto do programa, como comentários.
- Anotar erros, dúvidas e observações no final do programa, também como comentários. Usar /* ... */ para isso.

Conclusão

Analisar cada resultado obtido e avaliar-se ao fim do processo.