Bacharelado em Ciências da Computação – DInf/UFPR
CI067 – Oficina de Computação – 2º Semestre de 2008

Trabalho Prático

Prof. Daniel Weingaertner danielw@inf.ufpr.br
Prof. Aldri Luiz dos Santos aldri@inf.ufpr.br

v1.0: 26 de agosto de 2008

1 Introdução

A Universidade Federal do Paraná é bastante antiga – completa 100 anos em 2012. Neste período, as construções mais antigas da universidade passaram por diversas reformas e adaptações, de modo que andar por estes edifícios pode ser um pouco complicado, devido aos longos corredores, bifurcações inesperadas e locais sem saída.

Como conseqüência, muitos calouros têm dificuldade de encontrar o caminho para suas salas de aula, correndo sérios riscos de irem parar no Capão do Tigre[1], ou em outras áreas inóspitas da UFPR. Por isso, os alunos de CI067 (vocês) foram incumbidos de criar um jogo para ajudar os calouros a praticarem suas habilidades de orientação. O objetivo do jogo é encontrar a saída de um labirinto.

2 Especificação

O labirinto pode ser representado por uma matriz bidimensional, em que cada quadrado está livre ou preenchido por uma parede. Os quadrados livres podem estar vazios, conter portas ou chaves. Há quatro tipos diferentes de portas e chaves: *azuis*, *amarelas*, *verdes* ou *vermelhas*. Cada chave abre apenas portas da mesma cor.

[1] Área localizada no Campus Jardim Botânico da UFPR, entre as coordenadas $25°26'50'' - 25°27'33''S$ e $49°14'16'' - 49°14'33'W$.

O jogador pode mover-se entre quadrados livres adjacentes na direção vertical ou horizontal. Movimentos diagonais não são permitidos. Não é permitido atravessar paredes, nem abandonar a área do labirinto. Se um quadrado contiver uma porta, o jogador poderá acessá-lo somente se antes tiver "pisado" num quadrado com a chave apropriada. Depois que uma porta é "aberta", o quadrado correspondente é considerado livre.

2.1 Entrada dos Dados

A entrada dos dados consiste de um mapa do labirinto. A primeira linha contém dois números inteiros L e C ($1 \leq L, C \leq 100$) especificando o tamanho do mapa. Em seguida vêm L linhas contendo C caracteres por linha. Os caracteres válidos são os seguintes:

Caractere		**Significado**
cerquilha	`#`	parede
ponto	`.`	quadrado livre
subscrito	`_`	porta aberta (livre)
asterisco	`*`	posição do jogador
letras maiúsculas	`B Y R G`	porta azul, amarela, vermelha ou verde
letras minúsculas	`b y r g`	chave azul, amarela, vermelha ou verde
X maiúsculo	`X`	saída do labirinto

Note que é permitido que o labirinto tenha:

- mais de uma saída;
- nenhuma saída;
- mais de uma porta e/ou chave da mesma cor;
- chaves sem porta correspondente, e vice-versa.

Você pode assumir que o marcador da posição inicial do jogador (`*`) aparece exatamente uma vez no mapa dado.

A Seção 2.4 apresenta exemplos de labirintos de entrada e o respectivo resultado esperado do programa.

2.2 Execução do Programa

O pacote de software a ser construído consiste do programa `labirinto`, que deve ser executado da seguinte forma:

`labirinto [OPÇÕES]`

As seguintes opções **podem** (opcionalmente) ser fornecidas ao programa:

-i modo interativo (Seção 2.2.1). O jogo será jogado por um jogador que utiliza o teclado. Na ausência desta opção, o programa será executado no modo batch (Seção 2.2.2), ou seja, deve encontrar a saída do labirinto automaticamente.

-l arquivo carrega o mapa especificado em `arquivo`. Na ausência desta opção, a entrada deve ser lida de `stdin` (entrada padrão);

-o arquivo salva a saída do programa no arquivo `arquivo`. Na ausência desta opção, a saída deve ser escrita em `stdout` (saída padrão).

Na ausência de opções, o programa deve ser executado assumindo o modo padrão de cada opção.

2.2.1 Modo Iterativo

No modo iterativo, o programa deve ler o labirinto da entrada especificada na linha de comando, e mostrar na tela as teclas que o usuário pode utilizar para controlar o jogo. São elas:

0–9 dígitos de `0` a `9`. Determinam o raio de visão do jogador. O padrão é um $raio = 1$, ou seja, são mostrados apenas os quadrados vizinhos à posição do jogador. Caso $raio = 0$, então o labirinto inteiro deve ser mostrado (*for kids*). Estas teclas podem ser digitadas a qualquer momento do jogo, alterando a forma de visualização do mesmo.

S letra `S`. Inicia o jogo. Mostra na tela a posição atual do jogador no labirinto e as casas na vizinhança, de acordo com o *raio* definido.

→←↑↓ setas. Movem o jogador no sentido indicado pela seta e atualizam seu campo de visão de acordo com o *raio* definido. A movimentação deve respeitar as regras do jogo (paredes, chaves e portas) e o jogo encerra quando o jogador atinge uma saída.

Toda vez que uma porta é aberta, o caractere correspondente (`B Y R G`) deve ser substituído pelo caractere de porta aberta (`_`), que é equivalente ao quadrado livre.

Q letra `Q`. Encerra a execução do jogo e grava o resultado na saída especificada na linha de comando.

P letra `P`. Funcionalidade PAUSA. Grava o estado atual do jogo de forma que ele possa ser recarregado mais tarde, e a execução prossiga do ponto em que parou.

- Lembre-se que o caminho já percorrido, as chaves que o usuário já possui e as portas já abertas (portanto o estado atual do labirinto) devem ser salvos de forma que possam ser recuperados.
- O estado atual deve ser gravado no arquivo especificado pela opção `-o`. Caso esta opção não tenha sido definida, você deve solicitar ao usuário que digite o nome de um arquivo.
- O formato deste arquivo pode ser especificado livremente, desde que o programa seja capaz de lê-lo da mesma forma que lê os labirintos, conforme descrito na Seção 2.1.

2.2.2 Modo Batch

No modo batch, o programa deve ler o labirinto da entrada especificada na linha de comando e procurar automaticamente uma saída. Idealmente, o caminho encontrado deve ser o menor caminho possível.

2.3 Resultado do Programa

O resultado da execução do programa `labirinto` deve ser escrito na saída especificada na linha de comando (Seção 2.2). Este resultado independe do modo de execução (iterativo ou batch), e consiste de: 1) uma frase indicando o número de passos necessários para encontrar a saída, 2) as coordenadas percorridas desde a posição inicial até a saída, e 3) o mapa do labirinto após o término do jogo. Ou seja, o programa deve imprimir:

1. `O aluno encontrou a saida em N passos.`
   caso o jogador encontre a saída, sendo `N` o número de movimentos efetuados até encontrar a saída. Caso a saída não seja encontrada, ou não seja possível alcançá-la, o programa deve imprimir:
   `O aluno nao tem saida.`

2. As coordenadas dos passos dados pelo jogador (caso tenha encontrado a saída), desde a posição inicial (x_0, y_0) até a posição da saída (x_N, y_N), obedecendo ao seguinte formato:
   $(x_0, y_0) : (x_1, y_1) : ... : (x_N, y_N)$
   Não deve ocorrer quebra de linha (`\n`) entre as coordenadas do caminho. O canto superior esquerdo do labirinto possui coordenadas $(0, 0)$.

3. O estado final do labirinto, no mesmo formato da entrada de dados (Seção 2.1).

A Seção 2.4 apresenta exemplos de labirintos de entrada e o respectivo resultado esperado do programa.

2.4 Exemplos de Entrada de Dados e Resultados Esperados

- Entrada:

```
1 10
*........X
```

- Resultado:

```
O aluno encontrou a saida em 9 passos.
(0,0):(0:1):(0:2):(0:3):(0:4):(0:5):(0:6):(0:7):(0:8):(0:9)
1 10
*........X
```

- Entrada:

```
1 3
*#X
```

- Resultado:

```
O aluno nao tem saida.
1 3
*#X
```

- Entrada:

```
3 20
####################
#X..gBr.*.Rb.G.GG..#
####################
```

- Resultado:

```
O aluno encontrou a saida em 18 passos.
(1,8):(1,7):(1,6):(1,7):(1,8):(1,9):(1,10):(1,11):(1,10):(1,9):
(1,8):(1,7):(1,6):(1,5):(1,4):(1,3):(1,2):(1,1)
3 20
####################
#X..g_r.*._b.G.GG..#
####################
```

3 Produto a ser Entregue

Cada trabalho deve ser desenvolvido por um grupo composto de no mínimo UM e no máximo DOIS membros. Ambos membros devem ser alunos regularmente matriculados na mesma turma de CI067.

O grupo deve entregar um pacote de software completo contendo os fontes em linguagem C dos programas solicitados e documentação. O pacote deve ser arquivado e compactado com *tar(1)* e *bzip2(1)* em um arquivo chamado `login1.tar.bz2` (se grupo com 1 membro) ou `login1-login2.tar.bz2` (se grupo com 2 membros), onde `login1` e `login2` são os logins dos alunos que compõem o grupo.

O pacote deve ter a seguinte estrutura de diretórios e arquivos:

- `./login1-login2/`: diretório principal. Ao serem executados, os programas devem assumir que este é o diretório corrente;
- `./login1-login2/LEIAME`;
- `./login1-login2/bin/`: diretório que conterá os programas executáveis, após compilados via *make*;
- `./login1-login2/src/`: diretório contendo o arquivo Makefile e todos os arquivos fonte em linguagem C (`*.c` e `*.h`);
- `./login1-login2/html/`: diretório que conterá a documentação do código fonte em `html`, gerada pelo *doxygen(1)*;

Note que a extração dos arquivos em `login1-login2.tar.bz2` deve criar o diretório `login1-login2` contendo todos os arquivos e diretórios acima.

3.1 Documentação

O pacote deve conter um arquivo de documentação em texto plano (ASCII). Este arquivo, chamado LEIAME, deve conter as seguintes informações:

- autoria do software, isto é, nome e RA dos membros do grupo;
- lista dos arquivos e diretórios contidos no pacote e sua descrição (breve);
- como compilar o código fonte;
- um capítulo especial descrevendo os algoritmos e as estruturas de dados utilizadas, as alternativas de implementação consideradas e/ou experimentadas e os motivos que o levaram a optar pela versão entregue, as dificuldades encontradas e as maneiras pelas quais foram contornadas.
- *bugs* conhecidos;
- outras informações que forem julgadas importantes.

Além disso, o código fonte deve estar documentado de acordo com o padrão aceito pelo *doxygen(1)*. A documentação `html` deve ser gerada a partir da execução deste comando via *make*.

3.2 Arquivo Makefile

O arquivo Makefile deve possuir as regras necessárias para compilar os módulos individualmente e gerar o programa executável. As seguintes regras devem existir OBRIGATORIAMENTE:

- tudo: compila e instala todos os arquivos. Instalar significa copiar para `login1-login2/bin` o programa `labirinto` gerado durante a compilação;
- limpa: limpa os arquivos temporários como `*.~*, *.bak, core*`;
- faxina: limpa todos os arquivos temporários e os arquivos gerados pelo Makefile (*.o, executável em bin, documentação em html, etc.).

3.3 Arquivos Fonte

O programa deve ser implementado exclusivamente em linguagem C , e deve ser composto de no mínimo 3 (três) módulos, sendo obrigatória a criação do módulo `labirinto.c`. Este módulo deve conter apenas a função `main()` e diretivas `#include` e `#define` que se fizerem necessárias.

As definições de macros, tipos de dados, estruturas e protótipos de funções devem estar em arquivos de cabeçalho (extensão `.h`). A implementação das diversas funções do software deverá estar em arquivos com extensão `.c`.

4 Entrega

O PRAZO FINAL PARA ENTREGA DAS SOLUÇÕES DESTE TRABALHO É **11 de novembro de 2008, 22:00h**, IMPRETERIVELMENTE. NÃO SERÃO ACEITOS TRABALHOS ENTREGUES APÓS ESTA DATA E HORÁRIO. O trabalho deve ser enviado como anexo por e-mail ao professor responsável pela disciplina:

- A mensagem com o anexo deve ser enviada ao Prof. de sua turma: Daniel Weingaertner `danielw@inf.ufpr.br` ou Aldri Luiz dos Santos `aldri@inf.ufpr.br` com o Assunto (*Subject*): CI067 - Trabalho .
- No corpo da mensagem DEVE CONSTAR OBRIGATORIAMENTE o Nome e Números de Registro Acadêmico (RA) dos membros do grupo;
- O grupo deverá considerar o trabalho como entregue SOMENTE APÓS RECEBER DO PROFESSOR UMA MENSAGEM DE CONFIRMAÇÃO DE RECEBIMENTO dentro de 24 horas desde o envio da mensagem;
- ATENÇÃO: Uma versão impressa do arquivo LEIAME deve ser entregue até o dia 13 de novembro de 2008, 14:00h. A entrega deve ser feita diretamente ao professor ou à Recepção do DInf[2]. Somente serão aceitas as versões impressas dos grupos que entregaram o trabalho via e-mail.

5 Critérios de Avaliação

APENAS OS TRABALHOS QUE FUNCIONAREM SERÃO CORRIGIDOS. Se o trabalho não compilar ou acusar falha de segmentação (*Segmentation fault*) durante os testes realizados pelo professor (sem que qualquer operação se efetue a contento), trará para o grupo NOTA 0 (ZERO). Também receberão NOTA 0 (ZERO) trabalhos plagiados de qualquer fonte, e/ou com códigos idênticos ou similares. Além disso, apenas trabalhos entregues no prazo marcado receberão nota.

[2] Ao entregar o documento LEIAME na Recepção do DINF, solicitar ao funcionário que o documento seja colocado no escaninho do professor

Legibilidade, elegância, eficiência, modularidade, coesão e acoplamento entre módulos, e uso adequado de estruturas de dados serão levados em conta na avaliação. Os algoritmos de manipulação das estruturas de dados (inserção, ordenação, etc.) devem ser tão eficientes quanto possível, usando todos os conceitos vistos nas disciplinas de Algoritmos do Curso. Ponteiros e alocação dinâmica de memória devem ser usados onde for eficiente e conveniente. Estruturas dinâmicas abordadas em Alg II, devidamente utilizadas, colaboram fortemente para uma avaliação positiva.

Caso seu programa possua *bugs* conhecidos, descreva-os no arquivo LEIAME. A documentação de um *bug* pode evitar que o grupo receba nota zero.

Os itens de avaliação do trabalho e respectivas pontuações são:

Qualidade da documentação: 10 pontos

Qualidade do código: 20 pontos

Modo iterativo: 40 pontos (Seção 2.2.1)

- **Eficiência da implementação:** 20 pontos
- **Funcionalidade Raio de visão:** 10 pontos
- **Funcionalidade Pausa:** 10 pontos

Modo batch: 30 pontos (Seção 2.2.2)

- **Eficiência da implementação:** 30 pontos

Pontos extra 20 pontos

- **Caminho mínimo:** 10 pontos. Dado um conjunto de labirintos de teste, as 3 equipes que obtiverem os menores caminhos no *modo batch* receberão esta pontuação extra.
- **Eliminação de laços:** 10 pontos. Se o caminho impresso na saída do programa não contiver laços desnecessários (caminhos que passam por um mesmo quadrado duas vezes desnecessariamente), a equipe receberá esta pontuação extra.

Defesa: A defesa do trabalho será oral, e definirá a nota individual de cada membro da equipe, de acordo com seu conhecimento a respeito do trabalho.

O item **Qualidade da documentação** se refere ao arquivo LEIAME. O item **Qualidade do código** inclui os comentários no código fonte e grau de portabilidade do código. O item **Eficiência da implementação** inclui análise de estruturas de dados, de algoritmos utilizados e sua eficiência.