

Documentação do Trabalho 1

Fundamentos de Sistemas Operacionais

Rafael Akiyoshi
13/0040878

Bruno Pinheiro
09/0107853

Questão 01)

- **Qual sistema operacional foi utilizado?**

O sistema operacional utilizado foi o Linux, distribuição Ubuntu 14.04 LTS e distribuição arch linux: 4.10.1-1-ARCH #1 SMP PREEMPT x86_64 GNU/Linux

- **Qual ambiente de desenvolvimento foi utilizado?**

Editor de texto: Atom e VIM
Compilador: GCC

- **Quais são as instruções de uso?**

- 1) Acesse o diretório onde está o makefile.
- 2) Digite o comando: `$ make`
- 3) Isso fará com que o software em questão compilado, gerando um arquivo chamado q1

```
[gagos@ultra-gagos q01]$ make
cc -c -o src/io.o src/io.c
cc -c -o src/operations.o src/operations.c
cc -c -o src/polygon.o src/polygon.c
cc -c -o src/main.o src/main.c
gcc -o q1 src/io.o src/operations.o src/polygon.o src/main.o
```

- 4) Agora para executar, basta digitar o comando: `$./q1`

5) Executando o comando acima, o sistema irá esperar a entrada de 4 coordenadas, tendo cada uma x e y.

Exemplo: Supondo que você queira entrar com as seguintes coordenadas:


q1 = (0,0)
q2 = (0,1)
q3 = (1,1)
q4 = (1,0)

Basta digitar as coordenadas separadas por espaço, ficando assim:

```
@rafael [~/workspace/cworkspace/fso/T01/questao01] (master) ./q1
0 0 0 1 1 1 1 0
```

6) Agora o programa lhe mostrará se o quadrilátero o qual você entrou com as coordenadas é convexo ou não e em caso positivo lhe mostrará sua área. (Um desenho ilustrativo foi feito do lado direito do console)

```
@rafael [~/workspace/cworkspace/fso/T01/questao01] (master) ▶ ./q1 0.1 1.1 1.0 0.0
0 0 0 1 1 1 1 0
Quadrilatero convexo.
Area: 1
@rafael [~/workspace/cworkspace/fso/T01/questao01] (master) ▶ ./q1 0.0 4.4 8.0 4.2
0 0 4 4 8 0 4 2
Quadrilatero nao convexo.
```



- O comando `$ make clean` pode ser usado para remover os objetos gerados para a compilação do programa (`src/*.o`) e o executável `q1`.

- **Quais são as limitações conhecidas?**

- É necessário ter acesso ao diretório da aplicação, contendo em si todos os arquivos sem terem sido alterados. São eles: `makefile`, `src/main.c` `src/io.c`, `src/operations.c`, `src/polygon.c`, `include/io.h`, `include/operations.h`, `include/polygon.h`.
- O programa espera que o usuário entre com 8 valores do tipo `double`, referente às coordenadas `x` e `y` do ponto no plano cartesiano. Se o programa for utilizado juntamente do operador pipe (`|`) a partir da saída de outra aplicação, caso sejam informados mais de 8 valores do tipo `double`, os valores extras serão desconsiderados. Caso sejam informados um total menor do que 8 valores do tipo `double`, o programa encerrará com o error 1 por ter identificado uma entrada do tipo `NaN` (Not a Number).
- Caso o usuário tente inserir um valor que não seja um número (como um `char`, por exemplo) o programa encerrará com o error 1, por ter identificado uma entrada do tipo `NaN` (Not a Number).

- **Casos de teste (TC) que demonstram que a aplicação contempla o comando do trabalho**

- TC1. Dados de entradas corretos para quadrilátero convexo:

```
@rafael [~/workspace/cworkspace/fso/T01/questao01] (master) ▶ ./q1 0.1 1.1 1.0 0.0
0 0 0 1 1 1 1 0
Quadrilatero convexo.
Area: 1
```

- TC2. Dados de entradas corretos para quadrilátero não convexo:

```
@rafael [~/workspace/cworkspace/fso/T01/questao01] (master) ▶ ./q1 0.0 4.4 8.0 4.2
0 0 4 4 8 0 4 2
Quadrilatero nao convexo.
```

- TC3. Chamada a partir do retorno de outra aplicação pelo operador pipe (`|`):

- TC3.1. Com entradas corretas:

```
[gagos@ultra-gagos q01]$ echo "0 0 0 1 1 1 1 0" | ./q1
Quadrilatero convexo.
Area: 1.
```

- TC3.2. Com número de entradas corretas maior que 8:

```
[gagos@ultra-gagos q01]$ echo "0 0 0 1 1 1 1 0 1 2" | ./q1
Quadrilatero convexo.
Area: 1.
```

- TC3.3. Com número de entradas corretas menor que 8:

```
[gagos@ultra-gagos q01]$ echo "0 0 0 1 1 1" | ./q1
Dado informado NaN. Abortando..
```

- TC4. Inserindo dado de tipo inválido:

```
[gagos@ultra-gagos q01]$ ./q1
0 1
1 a
Dado informado NaN. Abortando..
```

- **Referências para a resolução do exercício:**

Fundamentos Matemáticos para computação gráfica Tópico 8.10(http://www.inf.pucrs.br/~pinho/CG/Aulas/FundMatemat/FUND_MAT.html)

Um Estudo Sobre área de triângulos e polígonos convexos e não-convexos(<http://mat.ufcg.edu.br/profmat2/wp-content/uploads/sites/5/2015/07/Fernando.pdf>)

Questão 02)

- **Qual sistema operacional foi utilizado?**

O sistema operacional utilizado foi o Linux, distribuição Ubuntu 14.04 LTS e distribuição arch linux: 4.10.1-1-ARCH #1 SMP PREEMPT x86_64 GNU/Linux

- **Qual ambiente de desenvolvimento foi utilizado?**

Editor de texto: Atom e VIM
Compilador: GCC

- **Quais são as instruções de uso?**

- 1) Acesse o diretório onde está o makefile do programa em questão.
- 2) Digite o comando **\$ make** para compilar o programa, gerando um executável com o mesmo nome do diretório em que se encontra (executável q02 caso não tenha sido alterado).

```
[gagos@ultra-gagos q02]$ make
gcc -o q02 main.c
```

- 3) Execute o programa gerado passando os parâmetros desejados. Neste exemplo, o comando usado foi: **\$./q02 a b c d e f**

```
[gagos@ultra-gagos q02]$ ./q02 a b c d e f
# de parametros: 6
Nome do executavel: ./q02
Parametro #1: a
Parametro #2: b
Parametro #3: c
Parametro #4: d
Parametro #5: e
Parametro #6: f
```

- o comando `$ make clean` pode ser usado para remover o executável gerado pela etapa de compilação, como mostra a figura a seguir:

```
[gagos@ultra-gagos q02]$ ls
main.c  makefile  q02
[gagos@ultra-gagos q02]$ make clean
rm -f q02
[gagos@ultra-gagos q02]$ ls
main.c  makefile
```

- **Quais são as limitações conhecidas?**
 - o É necessário ter acesso ao diretório da aplicação, contendo em si todos os arquivos sem terem sido alterados. São eles: `makefile` e `main.c`
 - o A saída referente ao nome do executável trás em si a PATH utilizada para executar o programa, a partir do diretório atual em que o usuário se encontra ao executar o programa.
 - o Se o usuário deseja passar uma string como parâmetro, e a mesma possui espaços entre as palavras, o mesmo deve ser feito dentro de aspas duplas. Ex: “dia bonito”
- **Casos de teste que demonstram que a aplicação contempla o comando do trabalho**
 - o TC1. Execução passando nenhum parâmetro:

```
[gagos@ultra-gagos q02]$ ./q02
# de parametros: 0
Nome do executavel: ./q02
[gagos@ultra-gagos q02]$
```

- TC2. Execução passando N parâmetros:

```
[gagos@ultra-gagos q02]$ ./q02 1 2 "ab" $(echo "x y z")
# de parametros: 6
Nome do executavel: ./q02
Parametro #1: 1
Parametro #2: 2
Parametro #3: ab
Parametro #4: x
Parametro #5: y
Parametro #6: z
[gagos@ultra-gagos q02]$
```

- TC3. Execução passando parâmetros do tipo string com espaçamento e sem:

```
[gagos@ultra-gagos q02]$ ./q02 que "dia bonito"
# de parametros: 2
Nome do executavel: ./q02
Parametro #1: que
Parametro #2: dia bonito
[gagos@ultra-gagos q02]$
```

- TC4. Execução a partir de um diretório pai, passando N parâmetros:

```
[gagos@ultra-gagos T01]$ ./q02/q02 chamada em diretorio pai
# de parametros: 4
Nome do executavel: ./q02/q02
Parametro #1: chamada
Parametro #2: em
Parametro #3: diretorio
Parametro #4: pai
[gagos@ultra-gagos T01]$
```

- **Referências para a resolução do exercício:**

Advanced Linux Programming(Manual disponível no Moodle)

Questão 03)

- **Qual sistema operacional foi utilizado?**

O sistema operacional utilizado foi o Linux, distribuição Ubuntu 14.04 LTS e distribuição arch linux: 4.10.1-1-ARCH #1 SMP PREEMPT x86_64 GNU/Linux

- **Qual ambiente de desenvolvimento foi utilizado?**

Editor de texto: Atom e VIM

Compilador: GCC

- **Quais são as instruções de uso?**

1) Acesse o diretório onde está o makefile do programa em questão.

2) Digite o comando \$ make para compilar o programa, gerando um executável de

nome q03.

```
[gagos@ultra-gagos q03]$ make
cc -c -o src/io.o src/io.c
cc -c -o src/array.o src/array.c
cc -c -o src/main.o src/main.c
gcc -o q03 src/io.o src/array.o src/main.o
[gagos@ultra-gagos q03]$
```

3)Execute o programa passando um *switch* desejado. Caso o switch seja '-r', a ordenação será decrescente, e '-d' para crescente. Caso não seja informado nenhum *switch* a ordenação será crescente por *default*. Exemplo: [\\$./q03 -r](#).

4)Após sua chamada, o programa irá aguardar entradas do usuário até que o mesmo informe o número -1, no qual encerrará o programa.

```
[gagos@ultra-gagos q03]$ ./q03 -d
20
1
30
47
0
-1
Saida crescente:
0 1 20 30 47
[gagos@ultra-gagos q03]$
```

- O comando: [\\$./q03 -r](#) pode ser usado para remover os arquivos src/*.o e o executável q03.

```
[gagos@ultra-gagos q03]$ make clean
rm -f src/*.o q03
[gagos@ultra-gagos q03]$
```

- **Quais são as limitações conhecidas?**

- O programa q03 irá receber N parâmetros informados pelo usuário onde fará uma análise de todos. Caso o parâmetro seja "-r" ou "-d" o programa o identificará e o salvará como *switch*.
- A *switch* definida será o ultimo parâmetro "-r" ou "-d" informado/identificado pelo programa.
- Apesar do escopo da questão ser apenas os **números naturais não-negativos** e fator de parada, o número -1, o programa não retornará erro e nem encerrará sua execução caso o usuário tente inserir números menores que -1 ou valores do tipo char, que serão ignorados pelo programa.

- **Casos de teste que demonstram que a aplicação contempla o comando do trabalho**

- TC1. Teste inserindo N números naturais não-negativos e controle de parada -1:

```
[gagos@ultra-gagos q03]$ ./q03
20
5
4
9
222
3567
-1
Saida crescente:
4 5 9 20 222 3567
[gagos@ultra-gagos q03]$
```

- TC2. Teste inserindo N valores, incluindo números menores que -1 e valores do tipo char e “strings”:

```
[gagos@ultra-gagos q03]$ ./q03
20
4
a
-34
983
"hahaha"
hehe
84
9
0
34
-1
Saida crescente:
0 4 9 20 34 84 983
[gagos@ultra-gagos q03]$
```

- TC3. Teste dos parâmetros de entrada:
 - TC3.1. Passando 1 parâmetros “-r” e quaisquer entradas N:

```
[gagos@ultra-gagos q03]$ ./q03 -r
55
42
3
17
2
101
teste
string
-1
Saida decrescente:
101 55 42 17 3 2
[gagos@ultra-gagos q03]$
```

- TC3.2. Passando 1 parâmetro “-d” e quaisquer entradas N:

```
[gagos@ultra-gagos q03]$ ./q03 -d
55
42
3
17
2
101
teste
string
a
-1
Saída crescente:
2 3 17 42 55 101
[gagos@ultra-gagos q03]$
```

- TC3.3. Passando N parâmetros e 1 “-d”:

```
[gagos@ultra-gagos q03]$ ./q03 a b c muitos -d parametros
44
0
8
71
-45
-1
Saída crescente:
0 8 44 71
[gagos@ultra-gagos q03]$
```

- TC3.4. Passando N parâmetros e vários “-d”ou “-r”

```
[gagos@ultra-gagos q03]$ ./q03 -d -r repeticoes -d -r -r
45
12
245
653
-444
teste
-1
Saída decrescente:
653 245 45 12
[gagos@ultra-gagos q03]$
```

- **Referências para a resolução do exercício:**

ponteiro de função

<[http://www.cs.yale.edu/homes/aspnes/pinewiki/C\(2f\)FunctionPointers.html](http://www.cs.yale.edu/homes/aspnes/pinewiki/C(2f)FunctionPointers.html)>