Lista de Exercícios 1 - CAP-241 2017

Prof. Dr. Gilberto Ribeiro de Queiroz.

Aluno: Paulo Henrique Barchi[1a]

2 de abril de 2017

[1]paulobarchi@gmail.com

[a] Laboratório Associado de Computação e Matemática Aplicada (LAC)
Coordenação de Laboratórios Associados (CTE)
Instituto Nacional de Pesquisas Espaciais (INPE)
São José dos Campos, SP - Brasil.

Exercício 01. Considere o seguinte programa:

```
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  int main()
7  {
8     string nome;
9     cout << "Qual o seu nome? ";
10    cin >> nome;
11    cout << "\nOlá " << nome << "!" << endl;
12    return 0;
13 }
```

a) Comente a linha 01 e veja o que acontece quando você compila o programa. Reporte a mensagem de seu compilador, interpretando-a.

b) Comente a linha 04 e veja o que acontece quando você compila o programa. Reporte a mensagem de seu compilador, interpretando-a.

c) Altere o nome da função `main` para `principal` e veja o que acontece quando você compila o programa. Reporte a mensagem de seu compilador, interpretando-a.

Solução. a) A mensagem do compilador com a linha 01 comentada segue abaixo. Ao comentar a linha `#include <iostream>`, o compilador não compila a instrução de inclusão da biblioteca `iostream`, e, com isso, funções básicas de entrada e saída como `cout`, `cin` e `endl` não são reconhecidas. A mensagem de erro reporta isso, e, consequentemente, o programa não é compilado com sucesso.

```
exercicio1.cpp: In function ‘int main()’:
exercicio1.cpp:9:2: error: ‘cout’ was not declared in this scope
  cout << "Qual o seu nome? ";
  ^
exercicio1.cpp:10:2: error: ‘cin’ was not declared in this scope
  cin >> nome;
  ^
exercicio1.cpp:11:38: error: ‘endl’ was not declared in this scope
  cout << "\nOlá " << nome << "!" << endl;
                                     ^
```

b) A mensagem do compilador com a linha 04 comentada é apresentada abaixo. Ao comentar a linha `using namespace std;`, o compilador não compila a instrução de utilização do espaço de nomes `std`, onde estão utilitários da biblioteca padrão do `C++` (*C++ Standard Library*), como `string`. Como o código foi escrito considerando esta inclusão de utilização do espaço de nomes `std` no espaço de nomes global, o compilador não reconhece `string`, `cout`, `cin` e `endl`, por exemplo. Para que o código funcionasse com esta linha comentada, basta incluir `std::` antes de cada uma destas *features*, por exemplo `std::string`.

```
exercicio1.cpp: In function ‘int main()’:
exercicio1.cpp:8:2: error: ‘string’ was not declared in this scope
  string nome;
  ^
exercicio1.cpp:8:2: note: suggested alternative:
In file included from /usr/include/c++/4.8/iosfwd:39:0,
                 from /usr/include/c++/4.8/ios:38,
                 from /usr/include/c++/4.8/ostream:38,
                 from /usr/include/c++/4.8/iostream:39,
                 from exercicio1.cpp:1:
/usr/include/c++/4.8/bits/stringfwd.h:62:33: note:   ‘std::string’
   typedef basic_string<char>    string;
                                 ^
exercicio1.cpp:8:9: error: expected ‘;’ before ‘nome’
  string nome;
         ^
exercicio1.cpp:9:2: error: ‘cout’ was not declared in this scope
  cout << "Qual o seu nome? ";
  ^
exercicio1.cpp:9:2: note: suggested alternative:
In file included from exercicio1.cpp:1:0:
/usr/include/c++/4.8/iostream:61:18: note:   ‘std::cout’
   extern ostream cout;  /// Linked to standard output
                  ^
exercicio1.cpp:10:2: error: ‘cin’ was not declared in this scope
  cin >> nome;
  ^
exercicio1.cpp:10:2: note: suggested alternative:
In file included from exercicio1.cpp:1:0:
/usr/include/c++/4.8/iostream:60:18: note:   ‘std::cin’
   extern istream cin;  /// Linked to standard input
                  ^
exercicio1.cpp:10:9: error: ‘nome’ was not declared in this scope
  cin >> nome;
         ^
exercicio1.cpp:11:38: error: ‘endl’ was not declared in this scope
  cout << "\nOlá " << nome << "!" << endl;
                                      ^
exercicio1.cpp:11:38: note: suggested alternative:
In file included from /usr/include/c++/4.8/iostream:39:0,
                 from exercicio1.cpp:1:
/usr/include/c++/4.8/ostream:564:5: note:   ‘std::endl’
     endl(basic_ostream<_CharT, _Traits>& __os)
     ^
```

c) A mensagem do compilador com o nome da função *main* alterado para *principal* é copiada abaixo. O principal trecho da mensagem de erro é *undefined reference to ‘main’*. O compilador C++ padrão requer sempre uma função `main` com retorno `int`. A sequência de instruções começa sempre

desta função `main`.

```
/usr/bin/ld: /usr/lib/debug/usr/lib/x86_64-linux-gnu/crt1.o(.debug_info):
    relocation 0 has invalid symbol index 11
/usr/bin/ld: /usr/lib/debug/usr/lib/x86_64-linux-gnu/crt1.o(.debug_info):
    relocation 1 has invalid symbol index 12
/usr/bin/ld: /usr/lib/debug/usr/lib/x86_64-linux-gnu/crt1.o(.debug_info):
    relocation 2 has invalid symbol index 2
/usr/bin/ld: /usr/lib/debug/usr/lib/x86_64-linux-gnu/crt1.o(.debug_info):
    relocation 3 has invalid symbol index 2
/usr/bin/ld: /usr/lib/debug/usr/lib/x86_64-linux-gnu/crt1.o(.debug_info):
    relocation 4 has invalid symbol index 11
/usr/bin/ld: /usr/lib/debug/usr/lib/x86_64-linux-gnu/crt1.o(.debug_info):
    relocation 5 has invalid symbol index 13
/usr/bin/ld: /usr/lib/debug/usr/lib/x86_64-linux-gnu/crt1.o(.debug_info):
    relocation 6 has invalid symbol index 13
/usr/bin/ld: /usr/lib/debug/usr/lib/x86_64-linux-gnu/crt1.o(.debug_info):
    relocation 7 has invalid symbol index 13
/usr/bin/ld: /usr/lib/debug/usr/lib/x86_64-linux-gnu/crt1.o(.debug_info):
    relocation 8 has invalid symbol index 12
/usr/bin/ld: /usr/lib/debug/usr/lib/x86_64-linux-gnu/crt1.o(.debug_info):
    relocation 9 has invalid symbol index 13
/usr/bin/ld: /usr/lib/debug/usr/lib/x86_64-linux-gnu/crt1.o(.debug_info):
    relocation 10 has invalid symbol index 13
/usr/bin/ld: /usr/lib/debug/usr/lib/x86_64-linux-gnu/crt1.o(.debug_info):
    relocation 11 has invalid symbol index 13
/usr/bin/ld: /usr/lib/debug/usr/lib/x86_64-linux-gnu/crt1.o(.debug_info):
    relocation 12 has invalid symbol index 13
/usr/bin/ld: /usr/lib/debug/usr/lib/x86_64-linux-gnu/crt1.o(.debug_info):
    relocation 13 has invalid symbol index 13
/usr/bin/ld: /usr/lib/debug/usr/lib/x86_64-linux-gnu/crt1.o(.debug_info):
    relocation 14 has invalid symbol index 13
/usr/bin/ld: /usr/lib/debug/usr/lib/x86_64-linux-gnu/crt1.o(.debug_info):
    relocation 15 has invalid symbol index 13
/usr/bin/ld: /usr/lib/debug/usr/lib/x86_64-linux-gnu/crt1.o(.debug_info):
    relocation 16 has invalid symbol index 13
/usr/bin/ld: /usr/lib/debug/usr/lib/x86_64-linux-gnu/crt1.o(.debug_info):
    relocation 17 has invalid symbol index 13
/usr/bin/ld: /usr/lib/debug/usr/lib/x86_64-linux-gnu/crt1.o(.debug_info):
    relocation 18 has invalid symbol index 13
/usr/bin/ld: /usr/lib/debug/usr/lib/x86_64-linux-gnu/crt1.o(.debug_info):
    relocation 19 has invalid symbol index 21
/usr/bin/ld: /usr/lib/debug/usr/lib/x86_64-linux-gnu/crt1.o(.debug_line):
    relocation 0 has invalid symbol index 2
/usr/lib/gcc/x86_64-linux-gnu/4.8/../../../x86_64-linux-gnu/crt1.o:
    In function `_start': (.text+0x20): undefined reference to `main'
collect2: error: ld returned 1 exit status
```

Exercício 02. Escreva um programa que leia o tamanho dos lados de um triângulo e que avalie se esses valores realmente formam um triângulo. Em caso positivo, classifique o triângulo em isósceles, escaleno ou equilátero. Esse programa deverá ser executado de forma iterativa até que o usuário entre com uma opção para encerrar o programa.

Solução. O código referente a este programa está no arquivo `exercicio02.cpp`, compactado junto com este documento e demais arquivos.

Exercício 03. Faça um programa que preencha um vetor de tamanho fixo com números aleatórios e em seguida realize a soma dos elementos desse vetor. O vetor deve ser definido em tempo de compilação.

Solução. O código referente a este programa está no arquivo `exercicio03.cpp`, compactado junto com este documento e demais arquivos. O tamanho fixo do vetor definido foi 10. Para que o vetor seja definido em tempo de compilação, a variável `SEMENTE` – semente para gerar números aleatórios – deve ser definida no comando de compilação com a opção `-D`, como segue:

```
g++ -std=c++11 -DSEMENTE=NULL -o exercicio03 exercicio03.cpp
```

Na forma como está exemplificado, `SEMENTE = NULL`, e, com isso, a semente para gerar números aleatórios é baseada no tempo atual – momento da compilação. Essa variável é definida em tempo de compilação.

Exercício 04. Faça um programa que preencha um vetor com números aleatórios e em seguida calcule a média dos elementos desse vetor. O usuário do programa deverá informar o número de elementos do vetor a ser preenchido e processado.

Solução. O código referente a este programa está no arquivo `exercicio04.cpp`, compactado junto com este documento e demais arquivos.

Exercício 05. A função abaixo realiza a ordenação de um vetor de números inteiros de forma crescente. Altere esta função para que ela ordene o vetor de forma decrescente.

```
1  void ordena(int A[], int n)
2  {
3     for(int j = 0; j < n - 1; ++j)
4     {
5        int m = j;
6
7        for(int i = j + 1; i < n; ++i)
8        {
9           if(A[i] < A[m])
10             m = i;
11       }
12
13       std::swap(A[m], A[j]);
14    }
15 }
```

Solução. A função alterada para ordenar de forma decrescente é apresentada abaixo. O arquivo `exercicio05.cpp`, compactado junto com este documento e demais arquivos, utiliza parte do código referente ao Exercício 04 para utilizar e verificar esta função para ordenação de vetores em ordem decrescente.

```
1  void ordena(int A[], int n)
2  {
3     for(int j = 0; j < n - 1; ++j)
4     {
5        int m = j;
6
7        for(int i = j + 1; i < n; ++i)
8        {
9           if(A[i] > A[m])
10             m = i;
11       }
12
13       std::swap(A[m], A[j]);
14    }
15 }
```

Exercício 06. A função `ordena` do Exercício 05 utiliza a função `swap` da biblioteca padrão, definida no arquivo <utiliy>, para realizar a troca dos valores dos elementos `A[i]` e `A[j]` (linha 13). No trecho de código abaixo, o programador resolveu criar sua própria função para troca de valores chamada `troca` (linha 04). Por que o programa abaixo não produz a saída esperada, com o vetor ordenado? Corrija o problema.

```
1  #include <iostream>
2  #include <utility>
3
4  void troca(int a, int b)
5  {
6    int tmp = a;
7    b = a;
8    a = tmp;
9  }
10
11 void ordena(int A[], int n)
12 {
13   for(int j = 0; j < n - 1; ++j)
14   {
15     int m = j;
16
17     for(int i = j + 1; i < n; ++i)
18     {
19       if(A[i] < A[m])
20         m = i;
21     }
22
23     troca(A[m], A[j]);
```

```
  }
}

int main()
{
  int A[] = { 30, 12, 76, 28, 3, 78, 4, 3, 11 };

  const std::size_t n = sizeof(A) / sizeof(int);

  std::copy(A, A + n, std::ostream_iterator<int>(std::cout, " "));
  std::cout << std::endl;

  ordena(A, n);

  std::copy(A, A + n, std::ostream_iterator<int>(std::cout, " "));
  std::cout << std::endl;

  return EXIT_SUCCESS;
}
```

Solução. O programa não produz a saída esperada porque a função `troca` tem 2 problemas: (1) o valor que estava guardado em `b` está sendo perdido na forma como está e (2) os parâmetros não estão sendo passado por referência, então, há alteração de valores, mas estes valores não correspondem às posições do *array*. Sobre (1), da forma como está apresentada no exercício, o valor de `a` é atribuído a `tmp`, e também a `b`. Com isso `a` também recebe o mesmo valor, e nada é trocado. E, mesmo corrigingo este problema, por causa do problema (2), a troca não estava sendo realizada no *array*.

Há duas formas para corrigir o problema (1): alterar a primeira atribuição da função troca, ou, alterar as outras duas atribuições. Abaixo, é apresentada a solução com a troca da segunda e terceira atribuição, bem como, a correção dos parâmetros passados por referência. O código completo referente a este programa corrigido está no arquivo `exercicio06.cpp`, compactado junto com este documento e demais arquivos.

```
void troca(int &a, int &b) // Primeira alteracao: parametros passados por referencia
{
  // Segunda alteracao: valor de b estava sendo perdido
  int tmp = a;
  a = b; // a = b em vez de b = a
  b = tmp; // b = temp em vez de a = tmp
}
```

Exercício 07. Crie estruturas para representar os seguintes elementos geométricos no espaço bidimensional: pontos, segmentos de reta, linhas poligonais e polígonos simples.

Solução. As estruturas para estes elementos geométricos constam no trecho de código `struct_poly.cpp` fornecido pelo professor. Conforme sugerido por e-mail, para este exercício, classes foram criadas para estes elementos geométricos. Para testar o funcionamento, analogamente ao que foi feito em `struct_poly.cpp`, o perímetro de um polígono simples também foi calculado. Este código consta no arquivo `exercicio07.cpp`, compactado junto com este documento e demais arquivos.

Exercício 08. Escreva um programa que pergunte ao usuário as coordenadas de latitude e longitude, em grau-decimal, de dois pontos quaisquer na esfera terrestre e que apresente a distância entre eles. Essa distância deverá ser calculada de acordo com a fórmula de Haversine[1]:

$$d(p,q) = 2r\ arcsin\sqrt{sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + cos\phi_1 cos\phi_2 sin^2\left(\frac{\lambda_2 - \lambda 1}{2}\right)}$$

onde

- r: é o raio da esfera ($\sim$ 6371km)
- ϕ_1 e ϕ_2: latitude dos pontos em radianos.
- λ_1 e λ_2: longitude dos pontos em radianos.

Neste exercício você deverá:

- Criar uma função chamada `DistanciaHaversine` de acordo com a fórmula apresentada.
- Utilizar a estrutura para representação de pontos criada no *exercício 07* como tipo dos parâmetros formais da função. Lembre-se que a longitude irá corresponder ao valor associado à componente `x` da estrutura, e a latitude, à componente `y`.

Solução. O código referente a este programa está no arquivo `exercicio08.cpp`, compactado junto com este documento e demais arquivos.

Exercício 09. Escreva uma função que calcule a menor distância entre um ponto e uma reta. Faça um programa que utilize esta função, possibilitando que o usuário entre com as informações de dois pontos pertencentes a reta, bem como o ponto para o qual deva ser avaliada a distância.

Dica: A menor distância entre um ponto P e uma reta r corresponde ao segmento de reta perpendicular a r que parte de P e chega a r.

Uma forma de computar esta distância consiste na utilização da forma normal de Hessean para retas. Para uma reta r que passa pelos ponto $P_1 = (x_1, y_1)$ e $P_2 = (x_2, y_2)$, temos a seguinte fórmula:

$$h(x,y) = \frac{(y_2 - y_1) \times (x - x_1) - (x_2 - x_1) \times (y - y_1)}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}} = 0$$

com: $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} > 0$.

A distância de $P = (x, y)$ a r é dada por $|h(x, y)|$.

Solução. O código referente a este programa está no arquivo `exercicio09.cpp`, compactado junto com este documento e demais arquivos.

[1] Veja: https://en.wikipedia.org/wiki/Haversine_formula.

Exercício 10. Escreva uma função que avalie se dois segmentos de reta se interceptam ou não. Utilize como tipo do parâmetro formal dessa função o tipo de dados criado no *exercício 07* para representação de segmentos de reta.

Dica: Seja os segmentos de reta, $S = \overline{P_1P_2}$ e $T = \overline{P_3P_4}$. Para saber se esses dois segmentos se interceptam, basta avaliar se os pontos P_1 e P_2 de S encontram-se em lados opostos da reta que contém o segmento T , e se os pontos P_3 e P_4 de T estão em lados opostos da reta que contém o segmento S.

A equação de Hessean, mostrada no *exercício 9*, pode ser usada para solução desse problema. Os pontos P_1 e P_2 encontram-se em lados opostos da reta que contém o segmento T , se $h(P_1) \times h(P_2) < 0$. Nesta inequação, a avaliação do denominador da equação de Hessean se torna desnecessária.

Se $h(P_1) = 0$ ou $h(P_2) = 0$, significa que um desses pontos encontra-se sobre a reta que contém T. No caso de $h(P_1) = 0 \wedge h(P_2) = 0$, os segmentos S e T são colineares e um simples teste de intervalo com os pontos extremos dos segmentos é o suficiente para dizer se há ou não interseção.

Veja que não é pedido no exercício para que seja computado o ponto de interseção.

Além disso, deve ser considerado que os segmento S e T possuem interseção, se eles tiverem qualquer ponto em comum, incluindo as extremidades P_1, P_2 , P_3 , ou P_4.

Solução. O código referente a este programa está no arquivo `exercicio10.cpp`, compactado junto com este documento e demais arquivos.

Exercício 11. Polígono Simples: Seja $v = v_1, \ldots, v_n$ um conjunto de n pontos do plano, com $n > 3$. Seja $s = (s_1, \ldots, s_{n-1})$, uma sequência de $n - 1$ segmentos conectando os n pontos de v, tal que $s_i = \overline{v_i v_{i+1}}$, com $1 < i < (n - 1)$. Os segmentos formam um **polígono simples**, P, se:

- As interseções entre segmentos adjacentes em P ocorrerem apenas nos pontos extremos ligando esses segmentos, isto é, se $s_i \cap s_{i+i} = v_{i+i}$.
- Não ocorrer interseção entre segmentos não adjacentes, isto é $s_i \cap s_j = \emptyset$ para todo i, j, tal que $j \neq i + 1$.

Usando a representação para polígonos criada no *exercício 07*, faça uma nova função que verifique se um dado polígono, formado por um conjunto de vértices, é ou não simples. Apresente seus casos de teste em um programa principal.

Solução. O código referente a este programa está no arquivo `exercicio11.cpp`, compactado junto com este documento e demais arquivos.

Exercício 12. A área de um polígono simples pode ser calculada através da seguinte expressão:

$$area = \left| \frac{(x_1y_2 - y_1x_2) + (x_2y_3 - y_2x_3) + \cdots + (x_ny_1 - y_nx_1)}{2} \right|$$

Construa uma função que receba como parâmetro formal um polígono simples e que compute a área desse polígono. Use a mesma representação de polígonos criada no *exercício 07*.

Solução. O código referente a este programa está no arquivo `exercicio12.cpp`, compactado junto com este documento e demais arquivos.

Exercício 13. Quais são as principais abstrações fornecidas pelas linguagens de programação de alto nível e sua importância? Escreva uma pequena resenha sobre esse tópico.

Solução. Em linguagens de baixo nível, precisamos lidar com registradores, endereços de memória e pilha de execução, por exemplo. Um exemplo de linguagem de baixo nível seria `Assembly`. Em linguagens de alto nível lidamos com abstrações como variáveis, vetores, objetos, aritméticas complexas, expressões booleanas, *loops*, subrotinas e funções. Tais abstrações são conceitos abstratos de Ciência da Computação que focam na usabilidade sobre a eficiência ótima do programa. Linguagens de programação de alto nível têm poucos, se algum, elemento de linguagem que se traduz diretamente para a linguagem nativa de máquina (*operation code – opcode*). Com isso, tais linguagens fornecem padrões para realização de tarefas comuns sem tratar detalhadamente da arquitetura da máquina. A priori, pode parecer que códigos que precisam ser executados de maneira rápida e eficiente devam ser produzidos em linguagens de baixo-nível. No entanto, com o aumento da complexidade da arquitetura dos microprocessadores modernos, compiladores bem desenvolvidos para linguagens de alto nível frequentemente produzem código comparável em eficiência ao que a maioria dos programadores de linguagem de baixo nível conseguiriam produzir, e a alta abstração permite técnicas mais poderosas, e, com isso, melhores resultados no geral do que suas contrapartidas em baixo-nível para as mesmas configurações. Além do aumento da complexidade da arquitetura dos microprocessadores modernos, com a popularização da internet, a alta conectividade dos programadores pela rede junto com a grande disponibilização e troca de informação, favoreceu a aparição de otimizações e melhorias para compiladores, interpretadores, bibliotecas e abstrações.

Exercício 14. Qual a sua linguagem de programação preferida? Quais são os elementos dessa linguagem que te despertam o interesse em usá-la ou que tornam sua atividade de programação mais fácil/simples de ser realizada?

Solução. Nos últimos anos, minha linguagem de programação preferida é `PYTHON`, por todas as facilidades que ela promove. Além da sintaxe simples e direta, um destaque é a instalação simples de pacotes e bibliotecas com o comando `pip`. Outro, é o grande suporte para diversas tarefas através das bibliotecas disponíveis, como `pandas` para manipulação de *dataframes* – similar ao `R`; e `scikit-learn` para experimentos no contexto de aprendizado de máquina. Tarefas desafiadoras podem ser abordadas com poucas linhas de código. Há também o `Cython`, caso seja necessário/desejado obter uma perfomance próxima à da linguagem `C` com um código escrito em sua maior parte em `Python`.

Exercício 15. Qual a sua maior dificuldade com a lista de exercícios e com os tópicos ministrados nas duas primeiras aulas do curso? Algum assunto que você gostaria de maior detalhes?

Solução. Tive dificuldade em relembrar como é feita a criação de um ponteiro para um vetor de objetos. Não me lembro de ter visto anteriormente a passagem de parâmetros em tempo de compilação. O nível e diversidade da lista estão equilibrados e consistentes com o conteúdo apresentado em aula. Bons exercícios para conhecer ou revisitar princípios de programação e a linguagem `C++`.