Universidade Federal do Rio Grande do Norte
Departamento de Informática e Matemática Aplicada
DIM0120 – Linguagem de Programação I

Problema: desenvolvimento de habilidades de programação na linguagem C++.

Subproblema 1: introdução ao C++, compilação e depuração de programas, modularização.

Produtos do subproblema: documento PDF contendo respostas às questões abaixo, acompanhado dos códigos-fontes dos programas implementados.

Data de entrega via SIGAA: 05 de agosto de 2019.

Instruções: neste problema o aluno deve consultar as referências indicadas pelo docente para se familiarizar com os recursos necessários à criação de programas simples em C++, sem prejuízo à consulta de outras fontes como manuais e tutoriais. Usar as questões e programas mostrados a seguir como guia para as discussões em grupo e para orientar a exploração da linguagem C++. Para facilitar o aprendizado, recomenda-se que o aluno compare os recursos e conceitos de C++ com seu conhecimento prévio acerca de outras linguagens de programação. Leia e modifique os códigos mostrados e utilize os conceitos e recursos explorados para a criar os programas solicitados. Recursos exclusivos da linguagem C devem ser ignorados e substituídos por seus correspondentes em C++.

Questões[1]:

1. Quais as principais características do C++?
2. Podemos afirmar que C++ é um super conjunto de C?
3. Quais extensões de arquivo são comumente utilizadas para identificar códigos-fonte C++?
4. Utilizando o editor de texto Sublime, crie um novo documento a partir da listagem abaixo e, então, salve-o com o nome `list0101.cpp`.

```
/// Sort the standard input alphabetically.                                  1
/// Read lines of text, sort them, and print the results to the standard output. 2
/// If the command line names a file, read from that file. Otherwise, read from 3
/// the standard input. The entire input is stored in memory, so don't try   4
```

[1]Em grande parte inspiradas em *Exploring C++ 11*, Ray Lischner. Programas retirados desta mesma fonte.

```cpp
/// this with input files that exceed available RAM.

#include <algorithm>
#include <fstream>
#include <iostream>
#include <iterator>
#include <string>
#include <vector>

void read(std::istream& in, std::vector<std::string>& text)
{
   std::string line;
   while (std::getline(in, line))
       text.push_back(line);
}

int main(int argc, char* argv[])
{
   // Part 1. Read the entire input into text. If the command line names a file,
   // read that file. Otherwise, read the standard input.
   std::vector<std::string> text; ///< Store the lines of text here
   if (argc < 2)
     read(std::cin, text);
   else {
      std::ifstream in(argv[1]);
      if (not in)
      {
         std::perror(argv[1]);
         return EXIT_FAILURE;
      }
      read(in, text);
   }
   // Part 2. Sort the text.
   std::sort(text.begin(), text.end());
   // Part 3. Print the sorted text.
   std::copy(text.begin(), text.end(),
             std::ostream_iterator<std::string>(std::cout, "\n"));
}
```

lists/list0101.cpp

(a) O que este programa faz? Neste primeiro momento, não será necessário compreender todos os detalhes do código, mas perceba como os comentários ajudam na compreensão do programa, assim como a indentação ajuda a perceber a estrutura do código-fonte. Perceba, também, como uma boa escolha dos nomes de variáveis melhora a legibilidade do código.

(b) Observe a função `read()` (linhas 14-19). Qual a vantagem da utilização de subprogramas? Qual o princípio por trás desta estruturação do código?

(c) O que a seguinte linha de compilação realiza?

```
g++ -o list0101 -pedantic -std=c++11 list0101.cpp
```

(d) Execute o arquivo resultante da compilação. Como você pode testá-lo?

5. Considere o código mostrado na listagem `list0201.cpp`:

```cpp
/** @file list0201.cpp */
/** Listing 2-1. Reading Test */
/// Read the program and determine what the program does.

#include <iostream>
#include <istream>
#include <limits>
#include <ostream>

int main()
{
    int min(std::numeric_limits<int>::max());
    int max(std::numeric_limits<int>::min());
    bool any(false);
    int x;
    while (std::cin >> x)
    {
        any = true;
        if (x < min)
            min = x;
        if (x > max)
            max = x;
    }

    if (any)
        std::cout << "min = " << min << "\nmax = " << max << '\n';
}
```

lists/list0201.cpp

(a) O que este programa faz?

(b) Nas linhas 1-3 podemos identificar duas diferentes maneiras de comentar o código. Defina as regras adotadas por tais formas de comentários. O que acontecerá se trocarmos o comentário `///` por outro no estilo `/**...*/` no código acima?

(c) Qual a função das diretivas das linhas 5-8?

(d) Justifique a inclusão das bibliotecas para o funcionamento do programa `list0201.cpp`. Para isso, relacione as bibliotecas incluídas com os comandos utilizados neste programa.

(e) Na linha 7, substitua `limits` por `stimil`. O que acontecerá na compilação do programa?

(f) Na linha 10, substitua `main` por `MAIN`. O que acontecerá na execução do programa?

(g) Em quais condições o teste do `while` (linha 16) será avaliado para verdadeiro?

(h) Suponha uma nova versão deste código com a linha 19 modificada para `if (x)`.

i. O que você espera acontecer ao compilar o programa?

ii. O que você espera acontecer quando executar o programa?

iii. O que você espera obter como resultado ao fornecer os valores `0`, `1`, `2` e `3` como entrada?

iv. O que você espera obter como resultado ao fornecer os valores `3`, `2`, `1`, e `0` como entrada?

v. Explique os resultados obtidos por meio da execução do programa modificado.

6. Implemente e compile o programa abaixo.

```
/** @file list0203.cpp */
/** Listing 2-3. Determining the Number of Bits in a bool. */
#include <iostream>
#include <limits>
#include <ostream>

int main()
{
  // Note that "digits" means binary digits, i.e., bits.
  std::cout << "bits per bool: " << std::numeric_limits<bool>::digits << '\n';
  //std::cout << "not quoted; \"in quotes\", not quoted";
}
```

lists/list0203.cpp

   (a) Explique o resultado de sua execução.

   (b) Considerando as listagens `list0201.cpp` e `list0203.cpp`, o que você deduz sobre o comportamento do template `numeric_limits`?

   (c) Crie uma tabela resumindo os tipos básicos do C++.

   (d) Inclua o seguinte comando imediatamente após a linha 10.

   ```
   std::cout << "not quoted; \"in quotes\", not quoted";
   ```

   O resultado deste comando foi o esperado por você?

7. A listagem a seguir mostra um programa simples que executa entradas e saídas de dados:

```
/** @file list0501.cpp */
/** Listing 5-1. Demonstrating Input and Output */
#include <iostream>
#include <istream>
#include <ostream>

int main()
{
   std::cout << "Enter a number: ";
   int x;
   std::cin >> x;
   std::cout << "Enter another number: ";
   int y;
   std::cin >> y;

   int z(x + y);
   std::cout << "The sum of " << x << " and " << y << " is " << z << "\n";
}
```

lists/list0501.cpp

   (a) Suponha que usuário informa os números 42 e 21 como valores de entrada, separados por espaços. Qual a saída esperada? Execute o programa e verifique sua previsão.

   (b) Para cada valor de entrada mostrado a seguir, explique o resultado obtido:

      i. 42*21

ii. xyz

iii. 42-21

iv. 42.21

(c) Como a falta de inicialização das variáveis influencia nos resultados anteriores?

8. Suponha que você precisa escrever um programa que imprima os quadrados e cubos dos inteiros entre 1 e 20, de maneira a obter o seguinte resultado:

N	N^2	N^3
1	1	1
2	4	8
3	9	27
4	16	64
⋮	⋮	⋮
20	400	8000

Para ajudá-lo neste tarefa, tome o código esquemático a seguir como ponto de partida. Dica: observe os cabeçalhos incluídos neste esquema!

```
/** @file list0801.cpp */
/** Listing 8-1. Print a Table of Squares and Cubes */
#include <iomanip>
#include <iostream>
#include <ostream>

int main()
{
  std::cout << " N   N^2    N^3\n";
  for (int i = 1; i != 21; ++i)
  {
    // fill in the loop body here
  }
}
```

lists/list0801.cpp

9. O seguinte programa poderá ser compilado com sucesso? Identifique todos os novos recursos introduzidos e justifique sua resposta.

```
/** @file list0803.cpp */
/** Listing 8-3. Using Alternative Fill Characters */
#include <iomanip>
#include <iostream>
#include <ostream>

int main()
{
  using namespace std;

  int day(14);
  int month(3);
  int year(2006);
```

```
  int dollars(42);
  int cents(7);

  // Print date in USA order.
  cout << "Date: " << setfill('0') << setw(2) << month
                   << '/' << setw(2) << day
                   << '/' << setw(2) << year << '\n';
  cout << "Pay to the order of: CASH\n";
  cout << "The amount of $" << setfill('*') << setw(8) << dollars << '.'
                   << setfill('0') << setw(2) << cents << '\n';
}
```

lists/list0803.cpp

10. Considere o seguinte programa:

```
/** @file list0804.cpp */
/** Listing 8-4. Exploring Field Width, Padding, and Alignment  */
#include <iomanip>
#include <ios>
#include <iostream>
#include <ostream>

int main()
{
  using namespace std;
  cout << '|' << setfill('*') << setw(6) <<  1234 << '|' << '\n';
  cout << '|' << left <<         setw(6) <<  1234 << '|' << '\n';
  cout << '|' <<                 setw(6) << -1234 << '|' << '\n';
  cout << '|' << right <<        setw(6) << -1234 << '|' << '\n';
}
```

lists/list0804.cpp

(a) O que você espera como resultado da execução deste programa?

(b) Crie um programa que utilize larguras de campos, caracteres de preenchimento e manipuladores de alinhamento para produzir o seguinte resultado:

```
000042
420000
42
-42-
```

11. O seguinte programa imprime dez números não negativos:

```
/** @file list0701.cpp */
/** Listing 7-1. Using a for Loop to Print Ten Non-negative Numbers */
#include <iostream>
#include <ostream>

int main()
{
  for (int i(0); i != 10; i = i + 1)
    std::cout << i << '\n';
```

```
}
```

lists/list0701.cpp

Considerando este exemplo, crie um programa para computar e imprimir a soma dos inteiros de 10 a 20 a partir do esboço dado pela pelo programa `list0704.cpp`:

```
/** @file list0704.cpp */
/** Listing 7-4. Compute Sum of Integers from 10 to 20 */
#include <iostream>
#include <ostream>

int main()
{
  int sum(0);

  // Fill in the loop here.

  std::cout << "Sum of 10 to 20 = " << sum << '\n';
}
```

lists/list0704.cpp

12. Considere o programa definido pela listagem `list1902.cpp`:

```
/** @file list1902.cpp */
/** Listing 19-2. Separating Function Declarations from Definitions */
#include <iostream>
#include <istream>
#include <ostream>
#include <string>

void ignore_line();
int prompted_read(std::string prompt);
void print_result(int count, int sum);

/** Main program.
 * Read integers from the standard input and print statistics about them.
 */
int main()
{
  int sum(0);
  int count(0);

  while (std::cin)
  {
    int x(prompted_read("Value: "));
    if (std::cin)
    {
      sum = sum + x;
      ++count;
    }
  }
  print_result(count, sum);
}
```

```
                                                                    31
/** Prompt the user, then read a number, and ignore the rest of the line.  32
 * @param prompt the prompt string                                   33
 * @return the input number or -1 for end-of-file                    34
 */                                                                  35
int prompted_read(std::string prompt)                                36
{                                                                    37
  std::cout << prompt;                                               38
  int x(-1);                                                         39
  std::cin >> x;                                                     40
  ignore_line();                                                     41
  return x;                                                          42
}                                                                    43
                                                                     44
/** Ignore the rest of the input line. */                            45
void ignore_line()                                                   46
{                                                                    47
  char c;                                                            48
  while (std::cin.get(c) and c != '\n')                              49
    /*empty*/;                                                       50
}                                                                    51
                                                                     52
/** Print the statistics.                                           53
 * @param count the number of values                                 54
 * @param sum the sum of the values                                  55
 */                                                                  56
void print_result(int count, int sum)                                57
{                                                                    58
  if (count == 0)                                                    59
  {                                                                  60
    std::cout << "no data\n";                                       61
    return;                                                          62
  }                                                                  63
                                                                     64
  std::cout << "\ncount = " << count;                                65
  std::cout << "\nsum   = " << sum;                                  66
  std::cout << "\nmean  = " << sum/count << '\n';                    67
}                                                                    68
```

lists/list1902.cpp

(a) O que este programa faz?

(b) As linhas 8-10 apresentam declarações de funções. Estas declarações são obrigatórias?

(c) Qual a diferença entre a declaração e a definição de uma função?

(d) O programa apresentará resultados diferentes se a variável `x` for inicializada com `0` na linha 39? Por que?

(e) Utilize o GDB (GNU Debugger) para rastrear a execução do executável obtido pela compilação desta listagem. Durante a compilação devemos utilizar o parâmetro `-g` para que os símbolos necessários ao processo de depuração (como nomes das variáveis e referências às linhas do código fonte, por exemplo) sejam incluídos no executável gerado. Consulte as referências ao GDB indicadas no tópico de aula referente a este subproblema no SIGAA.