Laboratório 2

Uso das ferramentas de desenvolvimento de programas em C++.

Objetivo

O objetivo deste laboratório é realizar a implementação de um programa na linguagem de programação C++ utilizando ferramentas de suporte ao programador, tais como, o compilador e um sistema de controle de versão, aplicando modularização e outras boas práticas de programação. O laboratório inclui ainda o uso do depurador (GDB).

Questão 01

A *Geometria* (do grego γεωμετρια; *geo-* = "terra", *-metron* = "medida") é um ramo da Matemática que estuda questões relacionadas à forma, tamanho e posição relativa de figuras e a propriedades do espaço. Essa ciência surgiu em diversas culturas da antiguidade como um conjunto de conhecimentos práticos sobre comprimento, área e volume, partindo das necessidades e observações do ser humano para resolver problemas em agricultura, astronomia, arquitetura e engenharia. Com efeito, conhecimentos em Geometria são aplicados ainda hoje nos mais variados campos do conhecimento humano, tais como física, química, geologia, astronomia, engenharia, biologia, cartografia e computação.

Dentre as divisões da Geometria, encontram-se as chamadas *Geometria Plana* e *Geometria Espacial*. A Geometria Plana refere-se ao estudo das figuras geométricas definidas em um plano de duas dimensões, enquanto que a Geometria Espacial se encarrega do estudo das figuras geométricas (também chamadas de sólidos geométricos) definidas no espaço, ou seja, aquelas que possuem mais de duas dimensões e ocupam um lugar no espaço. As principais figuras geométricas planas são o *triângulo*, o *quadrado*, o *retângulo* e o *círculo*. Já as principais figuras geométricas espaciais são o *cubo*, a *esfera*, o *cone*, a *pirâmide*, o *paralelepípedo* e o *cilindro*.

Três conceitos são de suma importância para o entendimento das Geometrias Plana e Espacial, a saber, a *área*, o *perímetro* e o *volume*. A área de uma figura geométrica, seja ela plana ou espacial, expressa o tamanho de tal figura sobre uma superfície, de modo que quanto maior a superfície da figura, maior a sua área. O perímetro de uma figura geométrica é definido como a medida do contorno que delimita a figura, sendo resultante da soma das medidas de todos os seus lados. Por fim, o volume corresponde à medida do espaço ocupado por uma figura geométrica. Para encontrar os valores dessas medidas, é importante analisar o tipo da figura (se plana ou espacial) e a forma da figura, isto é, quantos e quais são os lados.

Área, perímetro e volume de figuras geométricas planas e espaciais

As Tabelas 1 e 2 apresentam a definição das principais figuras geométricas planas e espaciais, bem como as fórmulas utilizadas para calcular as medidas de área, perímetro e volume. É importante notar que, pelo fato de as figuras geométricas planas serem definidas em um plano de duas dimensões, elas não possuem volume.

Tabela 1. Área e perímetro das figuras geométricas planas

Figura	Definição	Área	Perímetro
Triângulo*	Figura fechada formada por três lados	$A = \frac{base \times altura}{2}$	$P = lado1 + lado2 + lado3$
Retângulo	Figura fechada formada por quatro lados que formam ângulos retos (90^o)	$A = base \times altura$	$P = 2 \times (base + altura)$
Quadrado	Figura fechada formada por quatro lados congruentes (isto é, de medidas iguais) que formam ângulos retos	$A = lado^2$	$P = 4 \times lado$
Círculo	Figura fechada por uma linha curva chamada circunferência	$A = \pi \times r^2$	$P^{**} = 2 \times \pi \times r$

* Para este exercício, você deverá considerar um triângulo equilátero, no qual os três lados são congruentes.

** O perímetro de um círculo é chamado de *comprimento da circunferência*. π é uma constante definida com o valor aproximado de 3,1415 e r é a medida do *raio* do círculo, isto é, a distância entre o centro e a extremidade do círculo.

Tabela 2. Área e volume das figuras geométricas espaciais

Figura	Definição	Área	Volume
Pirâmide	Figura composta por uma base poligonal* (triangular, quadrangular, etc.) e um vértice que une as faces laterais da pirâmide	$A = area\_base + area\_lateral^{**}$	$V = \frac{1}{3} \times area\_base \times altura$
Cubo	Figura composta por seis faces quadrangulares	$A = 6 \times aresta^2$	$V = aresta^3$
Paralelepípedo	Figura composta por seis faces, tendo três pares de faces idênticas e paralelas entre si	$A = (2 \times aresta1 \times aresta2) + (2 \times aresta1 \times aresta3) + (2 \times aresta2 \times aresta3)$	$V = aresta1 \times aresta2 \times aresta3$
Esfera	Figura resultante do conjunto de pontos do espaço cuja distância ao centro é igual ou menor que o raio	$A = 4 \times \pi \times r^2$	$V = \frac{4}{3} \times \pi \times r^3$

* Para este exercício, você deverá considerar uma pirâmide com base quadrangular, ou seja, contendo uma base formando um quadrado e quatro faces laterais triangulares.

** A área lateral de uma pirâmide é dada pela soma das áreas de todas as faces laterais triangulares.

Tarefas

A tarefa principal a ser realizada neste exercício é a implementação de um programa em C++ que permita calcular as medidas de diversas figuras geométricas planas e espaciais. O programa deverá receber, **através da linha de comando**, a indicação da forma geométrica e os dados necessários para os cálculos. Como resultado, o programa deve exibir as respectivas medidas (área, perímetro ou volume) da figura escolhida. Note que, para figuras geométricas planas, o programa deve exibir apenas a área e o perímetro; para figuras geométricas espaciais, o programa deve exibir apenas a área e o volume da figura. Para calcular tais medidas, você deve fazer uso das equações matemáticas apresentadas nas Tabelas 1 e 2, além dos seus próprios conhecimentos matemáticos na área de Geometria.

Na implementação das funções que calculam as medidas de área, perímetro e volume das figuras, você pode fazer uso da biblioteca `<cmath>`. Dentre as funções disponibilizadas por essa biblioteca, existem: (i) a função `pow`, que realiza a operação de potenciação de uma base e um expoente, recebidos como parâmetro, e (ii) a função `sqrt`, que calcula a raiz quadrada de um número recebido como parâmetro.

Exemplos de entrada e saída

Conforme mencionado anteriormente, o programa em execução deve receber por linha de comando a indicação da forma geométrica, seguida dos dados necessários para os cálculos. No exemplo a seguir, é passado na linha de comando a indicação da forma "retângulo", seguido dos dados "base" e "altura". Com isso, o programa irá instanciar a forma apropriada e imprimir o resultado dos cálculos.

```
#./geometria retangulo 5 3
# Area do retangulo: 15
# Perimetro do retangulo: 16
```

Seu programa deverá imprimir as instruções de uso, incluindo as opções de formas possíveis, toda vez que o programa for executado **sem parâmetros**!

Questão 01A

Uma vez que será necessário compilar e ligar vários arquivos de uma vez, você deverá escrever um arquivo `Makefile` para facilitar esse processo. Utilize a estrutura de diretórios apresentada em sala.

Questão 01B

Utilizando o *Doxygen*, realize as anotações necessárias para a geração automática da documentação.

Questão 02

Com a ajuda do depurador GDB, apresente os valores das variáveis *arg1* e *arg2* após a chamada de cada função no código abaixo, **justificando os valores apresentados**. Relate a sequência de comandos do GDB que você utilizou, indicando o propósito.

```
#include <iostream>

int funcX (int a, int b)
{
    ++a;
    b++;
    int result = a + b;
    return result;
}

int funcY (int* a, int b)
{
    int* y = new int;
    (*y) = (*a);
    (*y) *= 5;
    int result = (*y) + b;
    return result;
}

void funcZ (int a, int b, int* result)
{
    a++;
    (*result) += a + 2*b;
}

int main(int argc, char* argv[])
{
    int arg1 = 11;
    int arg2 = 23;
    funcX ( arg1, arg2);
    funcY ( &arg1, arg2);
    int resultado = 0;
    funcZ (arg1,arg2,&resultado);

    return 0;
}
```

Autoria e política de colaboração

Esta atividade é individual. Todos os arquivos de código fonte deverão ser mantidos em um repositório remoto Git para controle de versões, hospedado em algum serviço da Internet, a exemplo do GitHub, Bitbucket, Gitlab ou outro de sua preferência. Registre **todas as atividades** de implementação por meio de *commits*, cujo histórico será analisado durante a avaliação.

O trabalho em cooperação entre estudantes da turma é estimulado, sendo aceitável a discussão de ideias e estratégias. Contudo, tal interação não deve ser entendida como permissão para utilização de (parte de) código fonte de outros colegas, o que pode caracterizar situação de plágio. Trabalhos copiados em todo ou em parte de outros colegas ou da Internet serão sumariamente rejeitados.

Entrega

Você deverá submeter um único arquivo compactado no formato `.zip` contendo todos os códigos fonte resultantes da implementação deste exercício, sem erros de compilação e devidamente testados e documentados, **até às 23h59 do dia 22 de abril de 2018** através da opção *Tarefas* na Turma Virtual do SIGAA. Você deverá ainda informar, no campo *Comentários* do formulário de submissão da

tarefa, o endereço do repositório Git no qual foram disponibilizados todos os arquivos referentes ao programa implementado. O repositório deve incluir um arquivo README contendo a identificação completa do aluno e do laboratório, a descrição de como compilar e rodar o programa - descrevendo inclusive as dificuldades encontradas.

Orientações gerais

Você deverá observar as seguintes observações gerais na implementação deste exercício:

1) Apesar da completa compatibilidade entre as linguagens de programação C e C++, seu código fonte não deverá conter recursos da linguagem C nem ser resultante de mescla entre as duas linguagens, o que é uma má prática de programação. Dessa forma, deverão ser utilizados estritamente recursos da linguagem C++.
2) Durante a compilação do seu código fonte, você deverá habilitar a exibição de mensagens de aviso (*warnings*), pois eles podem dar indícios de que o programa potencialmente possui problemas em sua implementação que podem se manifestar durante a sua execução.
3) Aplique boas práticas de programação. Codifique o programa de maneira legível (com indentação de código fonte, nomes consistentes, etc.).
4) Busque desenvolver o seu programa com qualidade, garantindo que ele funcione de forma correta e eficiente. Pense também nas possíveis entradas que poderão ser utilizadas para testar apropriadamente o seu programa e trate adequadamente possíveis entradas consideradas inválidas.
5) A fim de garantir a boa manutenção de seu repositório, configure corretamente o arquivo `.gitignore` em seu repositório Git.

Avaliação

O trabalho será avaliado sob os seguintes critérios: (i) utilização correta dos conteúdos vistos anteriormente e nas aulas presenciais da disciplina; (ii) a corretude da execução do programa implementado, que deve apresentar saída em conformidade com a especificação e as entradas de dados fornecidas; (iii) a aplicação correta de boas práticas de programação, incluindo legibilidade, organização e documentação de código fonte, e; (iv) a utilização correta do repositório Git, no qual deverá estar registrado todo o histórico da implementação por meio de *commits*. A presença de mensagens de aviso (*warnings*) ou de erros de compilação e/ou de execução, a modularização inapropriada e a ausência de documentação são faltas que serão penalizadas. Este trabalho contabilizará nota de até 1,0 ponto na 1ª Unidade da disciplina.