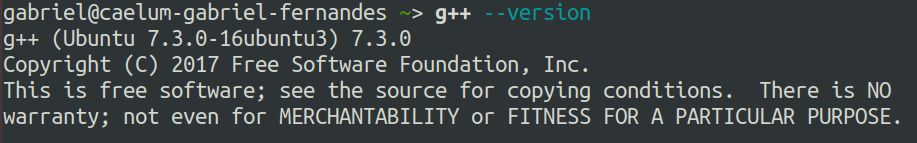
Para começar o curso, é necessário que se instale o compilador de C++.

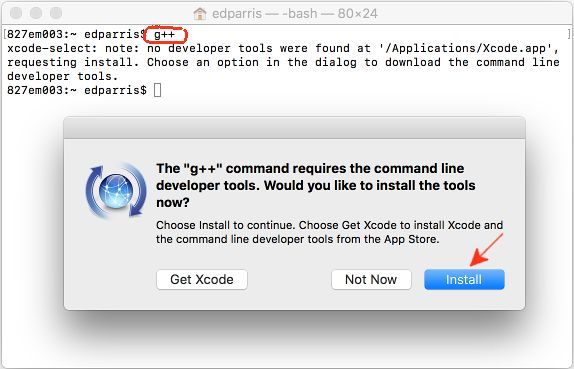
Para verificar se o compilador realmente está instalado, basta abrir um terminal e digitar g++ --version. Se o mesmo estiver instalado, o número de sua versão deverá aparecer, de modo similar ao seguinte:



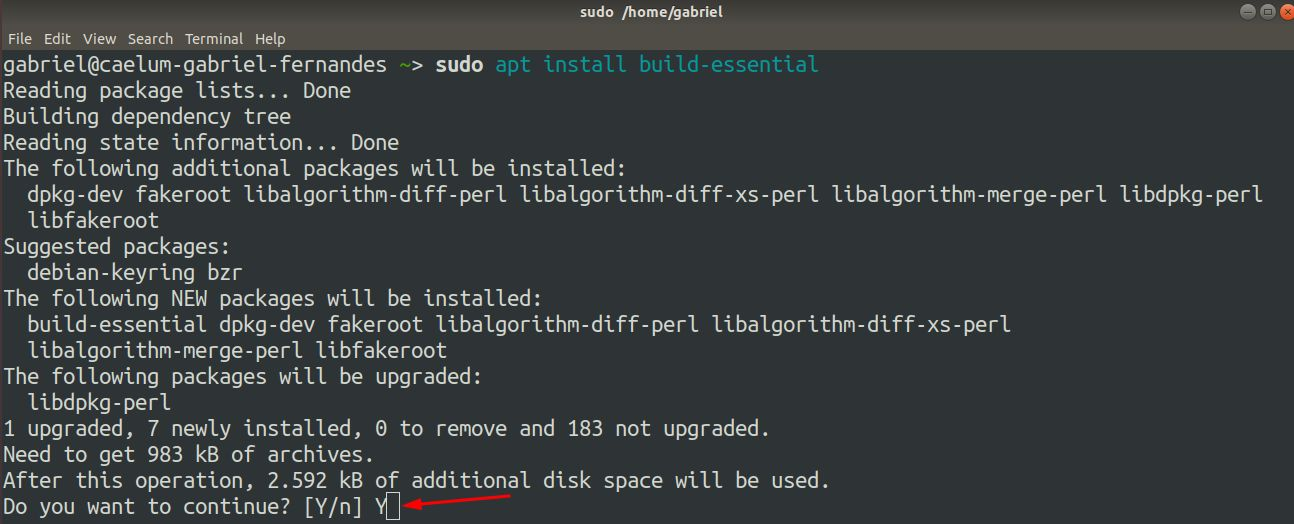
Caso você esteja usando um Linux ou Mac, ele já deve estar instalado.

Caso ele ainda não esteja instalado, você deverá seguir o seguinte passo a passo de acordo com seu sistema operacional:

* Um **Mac** oferecerá a opção de se instalar o Xcode, basta clicar em “Install”.



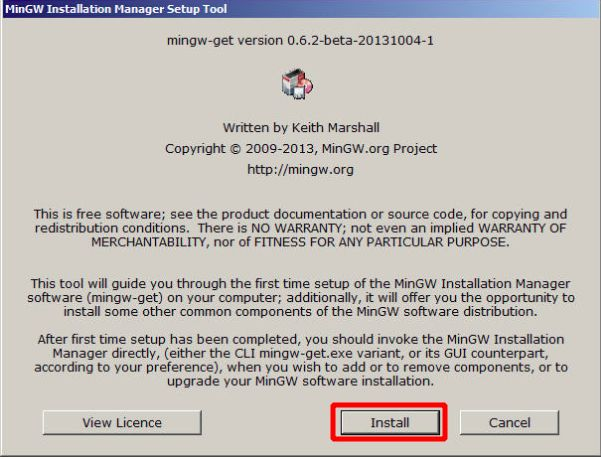
* Um **Linux** oferecerá um comando de terminal para se instalar o g++. Usando o Ubuntu por exemplo, você pode usar o comando: sudo apt install build-essential. Após rodar o comando, basta digitar “Y”, para confirmar a instalação. Finalizada a instalação, é necessário reiniciar o terminal.



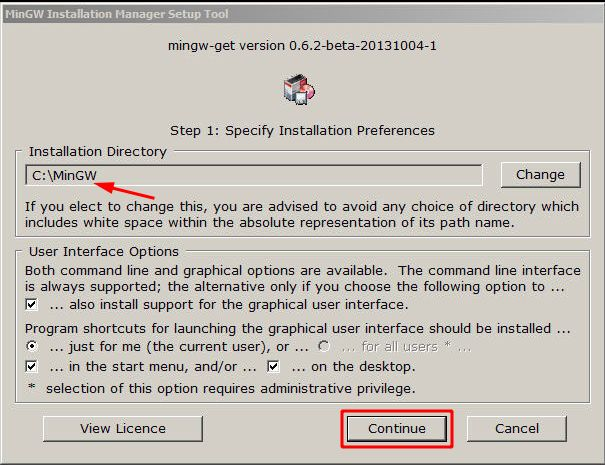
* Se você usa **Windows**, basta baixar e instalar o [MinGW](https://sourceforge.net/projects/mingw/files/latest/download).

Após o download você pode seguir esse passo a passo para instalar o MinGW no Windows:

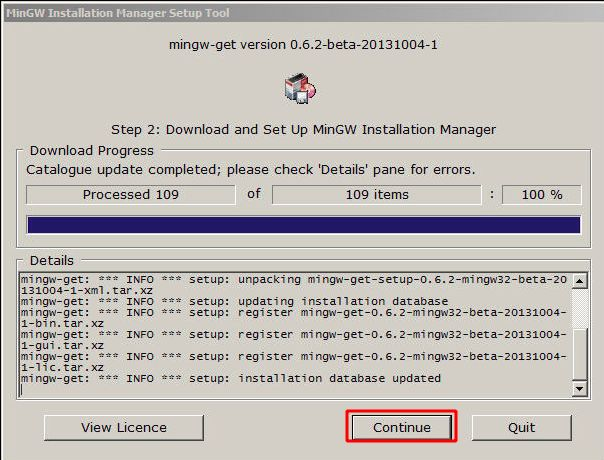
1. Ao executar o instalador uma tela como a seguinte deve aparecer, clique em “Install” para começar o processo de instalação:



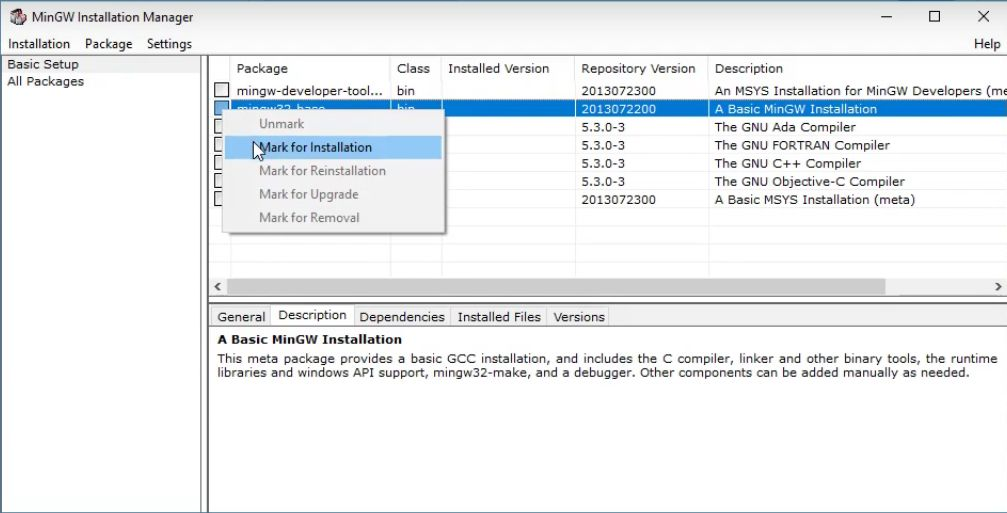
1. A seguinte tela então deve aparecer, verifique se o diretório de instalação (“Installation Directory”) é o endereço C:\MinGW, e então clique em “Continue” para seguir com o processo:

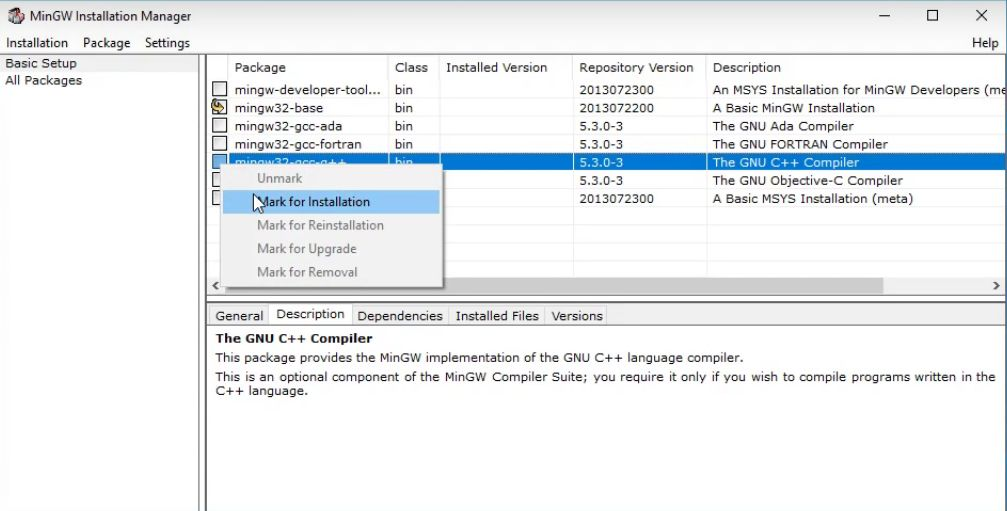


1. Após isso, você deve aguardar enquanto o MinGW baixa suas dependências. Quando ele tiver acabado, você deverá clicar novamente em “Continue”:

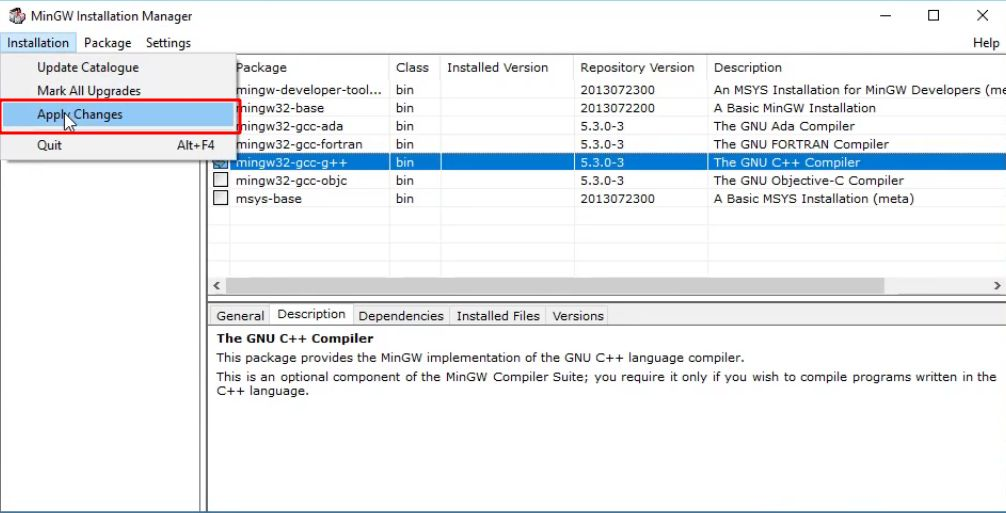


1. Você receberá, então, a opção de instalar alguns pacotes do MinGW, marque para instalação as opções mingw32-base e mingw32-gcc-g++:

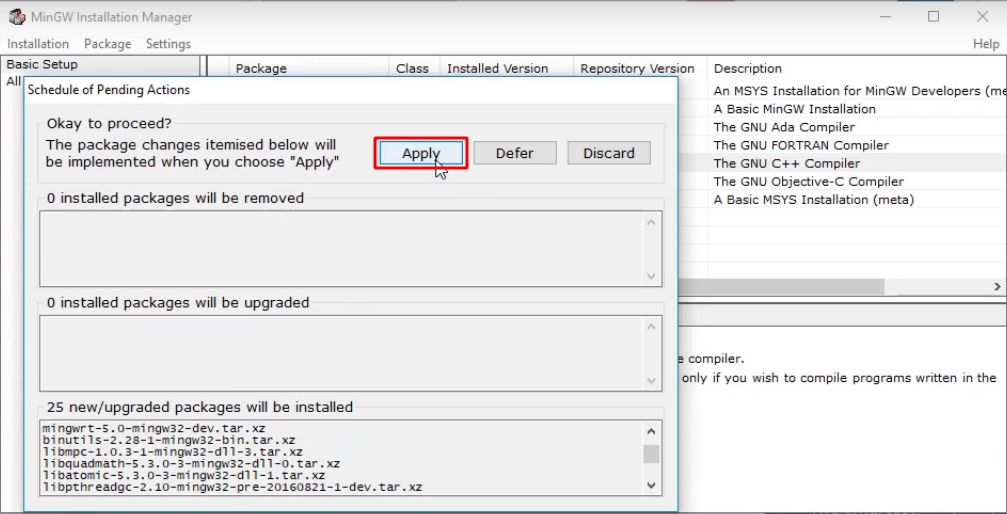




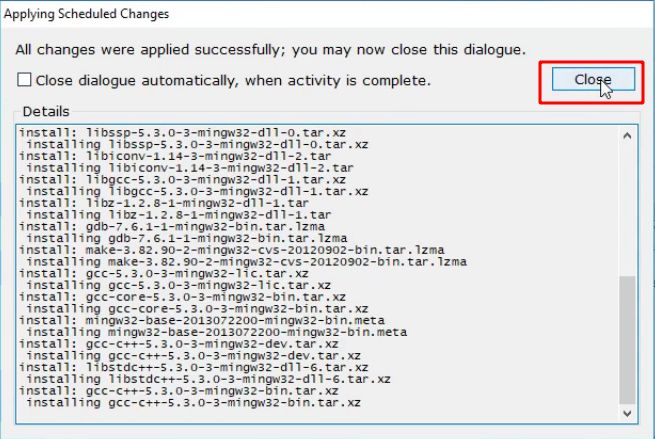
1. Após marcar os dois pacotes para instalação, basta clicar em “Installation”, e selecionar a opção “Apply Changes”, para iniciar a instalação dos pacotes selecionados:

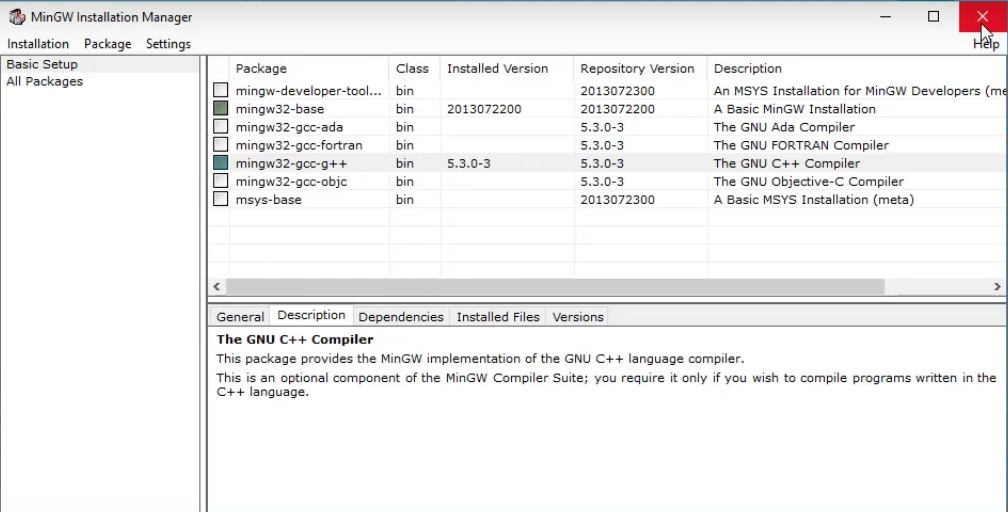


1. Outra janela então deve aparecer dizendo que novos pacotes serão instalados, basta clicar em “Apply” para começar sua instalação:

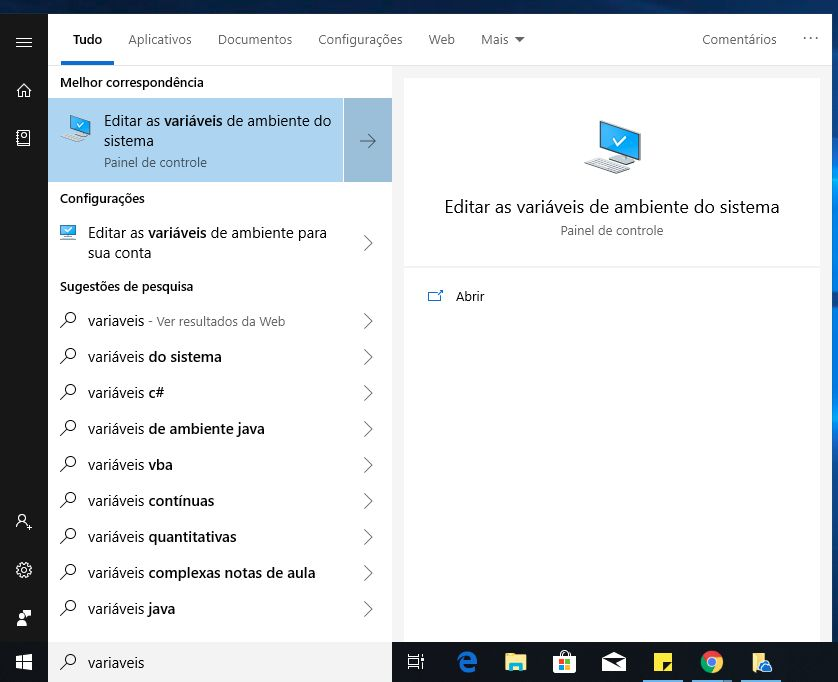


1. Ao final da instalação, basta clicar em “Close”. Já podemos fechar também o instalador do MinGW

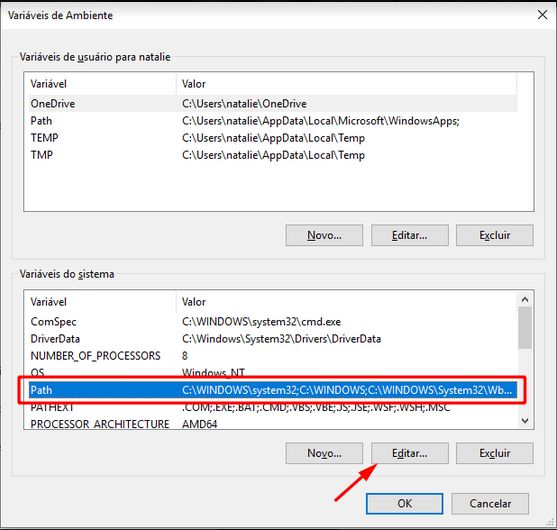




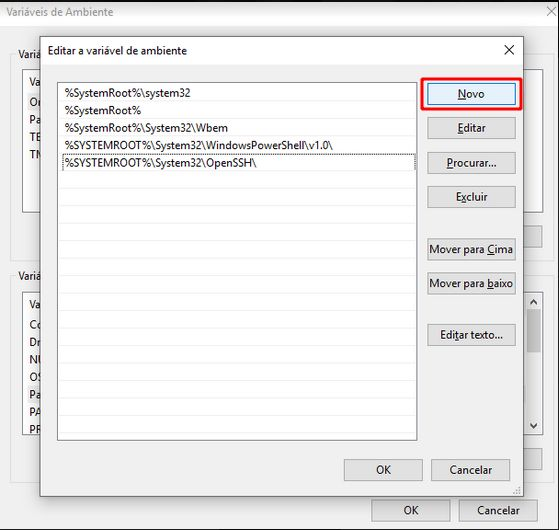
1. Agora, pesquise no Windows “Editar as variáveis de ambiente do sistema”:



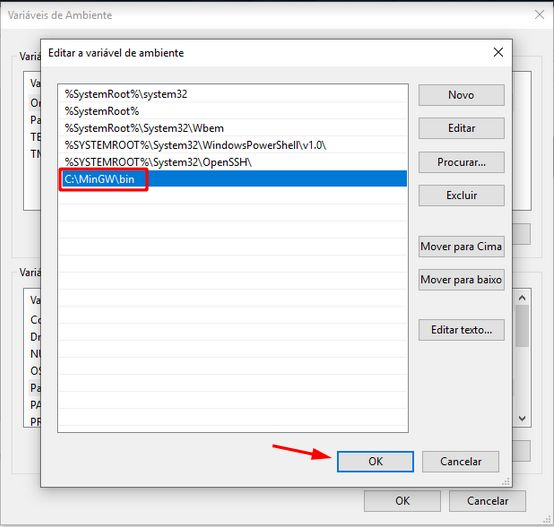
1. Selecione então a variável de nome “Path”, dentre as variáveis do sistema e clique em “Editar…”:



1. Aberta a página de edição da variável Path, clique em Novo:



1. Digite C:\MinGW\bin na caixa de texto que aparece e então clique OK:



1. Pronto! Agora seu computador está configurado para você começar a programar em C++!

Durante o curso usarei como editor de texto o Visual Studio Code. Seu uso é opcional, se desejar você pode baixá-lo gratuitamente por meio deste [link](https://code.visualstudio.com/Download).

Vamos escrever um programa que imprime o cabeçalho do jogo da adivinhação.

1. Começamos incluindo a biblioteca necessária para usar operações de saída no C++:

#include <iostream>

1. Adicionamos na linha seguinte, o marcador de uso das funções padrão de C++:

using namespace std;

1. Em seguida, criamos a função principal, que terá a seguinte estrutura:
2. int main () {

}

Usando os comandos cout para imprimir mensagens no terminal, e o endl para pular linhas, escreva o código que faz a impressão do seguinte cabeçalho:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Bem-vindos ao jogo da adivinhação \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Como ficou o código final?

Seu código final deve ser similar ao seguinte:

#include <iostream>

using namespace std;

int main () {

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\* Bem-vindos ao jogo da adivinhação \*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

}

Utilizamos o separador "<<" para imprimir mensagens usando o cout, além disso, para conseguirmos imprimir a mensagem no formato descrito no enunciado, precisamos pular a linha no final de cada linha, por isso usamos o endl.

Nesta aula fizemos as condições que determinavam se um chute é igual, maior ou menor que um número secreto. Para resolver esse problema definimos duas variáveis do tipo bool: acertou e maior.

Nosso código ficou mais ou menos assim:

const int NUMERO\_SECRETO = 42;

cout << "Qual seu chute? ";

cin >> chute;

bool acertou = ;

bool maior = ;

A única coisa que está faltando no trecho colado é a inicialização de acertou e maior.

Lembre-se que a variável acertou deve ser verdadeira se a variável chute for igual à variável NUMERO\_SECRETO e falsa caso contrário. Enquanto que maior deve ser verdadeiro se o chute for maior que o NUMERO\_SECRETO, e falso do contrário.

Complete o código com a inicialização de acertou e maior.

As inicializações devem ser as seguintes:

bool acertou = chute == NUMERO\_SECRETO;

bool maior = chute > NUMERO\_SECRETO;

Lembrando de sempre usar os dois sinais de igual, "==" para realizarmos comparações de igualdade.

Há situações em que devemos imprimir valores decimais com um número definido de casas depois da vírgula, como por exemplo:

* Para lidar com valores financeiros: devemos escrever R$ 12,50, com duas casas depois da vírgula, ao invés de apenas R$ 12,5
* Para escrever resultados de contas: não se escreve que a média de uma sala na prova foi 7,62156, geralmente usamos apenas uma casa depois da vírgula, dizemos que a média foi 7,6

O mesmo acontece no nosso jogo, por isso escolhemos seguir o padrão de sempre imprimir a pontuação com duas casas depois da vírgula.

Que linhas de código precisamos adicionar para que o cout imprima a pontuação seguindo esse padrão?

Primeiramente, precisamos modificar o cout usando o método precision, colocando entre parênteses a quantidade de casas depois da vírgula que desejamos:

cout.precision(2);

Logo em seguida, para garantir que o número não aparecerá em notação científica, é necessário usar o marcador fixed:

cout << fixed;

Deste modo, das próximas vezes que o cout for imprimir valores reais, ele sempre escreverá os valores com duas casas depois da vírgula.

Para que o valor do número secreto variasse entre as execuções do nosso jogo, usamos as funções srand() e rand().

A primeira serve pra determinar a semente que vai ser usada na hora de gerar os valores pseudo-aleatórios do nosso programa, e a segunda para retornar um valor aleatório.

Escreva o trecho que usamos no nosso jogo para fazer o sorteio de um número aleatório entre 0 e 99 usando essas funções.

O trecho escrito deve ser semelhante ao seguinte:

srand(time(NULL));

const int NUMERO\_SECRETO = rand() % 100;

Usamos a função time(NULL) como semente dos números aleatórios por ser uma função que está sempre mudando de valor.

Além disso, a função rand() pode retornar valores muito maiores que 99, por isso precisamos de algum jeito de diminuir esses valores grandes.

Para isso usamos a operação resto, denotada por %, como queremos valores de 0 a 99, basta tomarmos o resto da divisão do valor sorteado por 100, que desse modo sempre teremos um valor dentro dos limites desejados.

Jogo da Forca

O equivalente do comando make do Linux e Mac para o Windows se chama mingw32-make, caso se tenha realizado a instalação do pacote mingw do início do curso.

Seu uso se faz do mesmo modo que o comando make, basta rodar:

mingw32-make forca

O comando acima compila e gera o executável do programa chamado forca.cpp.

Na aula fizemos a alteração do seguinte laço no estilo de C++ para C++11:

for(int i = 0; i < PALAVRA\_SECRETA.size();i++){

if(chute == PALAVRA\_SECRETA[i]){

return true;

}

}

Agora é sua vez...

Altere o trecho anterior para usar o for em intervalos de C++11 e observe como o código ficará.

Após a modificação o trecho deverá ficar assim:

for(char letra : PALAVRA\_SECRETA){

if(chute == letra){

return true;

}

}

Deste modo, não precisamos usar uma variável inteira i apenas para passar, posição por posição, pelas letras de PALAVRA\_SECRETA.

Iterando diretamente podemos substituir onde usávamos PALAVRA\_SECRETA[i] por simplesmente letra.

Nesta aula declaramos a seguinte variável no nosso jogo:

map<char, bool> chutou;

Usamos ela para saber quais letras já foram chutadas.

Se, por exemplo, a letra 'A' já foi chutada chutou['A'] tem valor verdadeiro. Do contrário, se 'A' não foi chutada, chutou['A'] tem valor falso.

Escreva a função nao\_acertou que retorna **verdadeiro** caso a palavra secreta ainda possua alguma letra que não foi chutada, e **falso** caso a pessoa já tenha chutado todas as letras que compõem a palavra.

A função deverá ser semelhante ao seguinte trecho:

bool nao\_acertou(){

for(char letra : PALAVRA\_SECRETA){

if(!chutou[letra]){

return true;

}

}

return false;

}

Primeiramente, iteramos, letra por letra da palavra secreta, e verificamos se ela ainda não foi chutada usando o map:

for(char letra : PALAVRA\_SECRETA){

if(!chutou[letra]){

}

}

Se encontramos alguma letra que ainda não foi chutada, na hora já sabemos que o jogador ainda não acertou a palavra secreta, então retornamos verdadeiro.

for(char letra : PALAVRA\_SECRETA){

if(!chutou[letra]){

return true;

}

}

Caso contrário, se todas as letras da palavra já foram chutadas, então sabemos que o usuário já acertou a palavra secreta. Então apenas retornamos falso.

for(char letra : PALAVRA\_SECRETA){

if(!chutou[letra]){

return true;

}

}

return false;

Finalizamos assim então nossa função que retorna se o jogador ainda não acertou a palavra secreta.

Nessa aula implementamos a função le\_arquivo() que retorna um vector<string> das palavras escritas no banco de palavras.

Escrevemos também o código da função sorteia\_palavra, que faz uma chamada à função le\_arquivo para receber a lista de palavras salvas, sorteia uma dessas palavras, e atribui ela à variável global palavra\_secreta.

Reescreva o código da função sorteia\_palavra, seguindo o que fizemos na aula.

Sua função deve ficar da seguinte forma:

void sorteia\_palavra(){

vector<string> palavras = le\_arquivo();

srand(time(NULL));

int indice\_sorteado = rand() % palavras.size();

palavra\_secreta = palavras[indice\_sorteado];

}

Primeiramente recebemos a lista de palavras salva no arquivo, e armazenamos a mesma em uma variável chamada palavras:

vector<string> palavras = le\_arquivo();

Então sorteamos o índice da palavra que deverá ser a palavra secreta. Para fazer isso, precisamos primeiro atribuir a semente dos números aleatórios:

srand(time(NULL));

Agora sim, sorteamos um inteiro que será o indice da palavra secreta. Esse inteiro deverá ter valor de 0 ao número de palavras no banco menos um, já que as palavras são salvas no vector a partir da posição 0.

Por esse motivo, tiramos o resto da divisão do valor aleatório gerado pelo número de palavras no banco, que é basicamente o palavras.size().

int indice\_sorteado = rand() % palavras.size();

Tendo então o índice sorteado, sabemos que a palavra associada a esse índice será a palavra: palavras[indice\_sorteado], então basta agora atribuirmos à variável palavra\_secreta essa palavra escolhida.

palavra\_secreta = palavras[indice\_sorteado];

Terminamos assim a implementação da nossa função sorteia\_palavra.

Nesta aula aprendemos:

* Como ler e imprimir valores em arquivos usando os tipos ifstream e ofstream
* Como checar se um arquivo pode ser acessado usando o método is\_open()
* Como usar o método close() para indicar a finalização de uso de um arquivo previamente aberto
* Como interromper a execução de um programa usando exit(0)

Código fonte ao final do curso:

#include <iostream>

#include <string>

#include <map>

#include <vector>

#include <fstream>

#include <ctime>

#include <cstdlib>

using namespace std;

string palavra\_secreta;

map<char, bool> chutou;

vector<char> chutes\_errados;

bool letra\_existe(char chute){

for(char letra : palavra\_secreta){

if(chute == letra){

return true;

}

}

return false;

}

bool nao\_acertou(){

for(char letra : palavra\_secreta){

if(!chutou[letra]){

return true;

}

}

return false;

}

bool nao\_enforcou(){

return chutes\_errados.size() < 5;

}

void imprime\_cabecalho(){

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\* Jogo da Forca \*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << endl;

}

void imprime\_erros(){

cout << "Chutes errados: ";

for(char letra: chutes\_errados){

cout << letra << " ";

}

cout << endl;

}

void imprime\_palavra(){

for(char letra : palavra\_secreta){

if(chutou[letra]){

cout << letra << " ";

}

else{

cout << "\_ ";

}

}

cout << endl;

}

void chuta(){

cout << "Seu chute: ";

char chute;

cin >> chute;

chutou[chute] = true;

if(letra\_existe(chute)){

cout << "Você acertou! Seu chute está na palavra." << endl;

}

else{

cout << "Você errou! Seu chute não está na palavra." << endl;

chutes\_errados.push\_back(chute);

}

cout << endl;

}

vector<string> le\_arquivo(){

ifstream arquivo;

arquivo.open("palavras.txt");

if(arquivo.is\_open()){

int quantidade\_palavras;

arquivo >> quantidade\_palavras;

vector<string> palavras\_do\_arquivo;

for(int i=0;i<quantidade\_palavras;i++){

string palavra\_lida;

arquivo >> palavra\_lida;

palavras\_do\_arquivo.push\_back(palavra\_lida);

}

arquivo.close();

return palavras\_do\_arquivo;

}

else{

cout << "Não foi possível acessar o banco de palavras." << endl;

exit(0);

}

}

void sorteia\_palavra(){

vector<string> palavras = le\_arquivo();

srand(time(NULL));

int indice\_sorteado = rand() % palavras.size();

palavra\_secreta = palavras[indice\_sorteado];

}

void salva\_arquivo(vector<string> nova\_lista){

ofstream arquivo;

arquivo.open("palavras.txt");

if(arquivo.is\_open()){

arquivo << nova\_lista.size() << endl;

for(string palavra : nova\_lista){

arquivo << palavra << endl;

}

arquivo.close();

}

else{

cout << "Não foi possível acessar o banco de palavras." << endl;

exit(0);

}

}

void adiciona\_palavra(){

cout << "Digite a nova palavra, usando letras maiúsculas." << endl;

string nova\_palavra;

cin >> nova\_palavra;

vector<string> lista\_palavras = le\_arquivo();

lista\_palavras.push\_back(nova\_palavra);

salva\_arquivo(lista\_palavras);

}

int main () {

imprime\_cabecalho();

le\_arquivo();

sorteia\_palavra();

while(nao\_acertou() && nao\_enforcou()){

imprime\_erros();

imprime\_palavra();

chuta();

}

cout << "Fim de jogo!" << endl;

cout << "A palavra secreta era: " << palavra\_secreta << endl;

if(nao\_acertou()){

cout << "Você perdeu! Tente novamente!" << endl;

}

else{

cout << "Parabéns! Você acertou a palavra secreta!" << endl;

cout << "Você deseja adicionar uma nova palavra ao banco? (S/N) ";

char resposta;

cin >> resposta;

if(resposta == 'S'){

adiciona\_palavra();

}

}

}

Conteúdo do arquivo “palavras.txt”

6

ABACAXI

BANANA

UVA

MELANCIA

MORANGO

LARANJA