ZAPIS DO PLIKÓW I ODCZYT Z PLIKÓW

Bardzo często w programowaniu używamy plików jako źródła danych i miejsca przeznaczenia wyników. Aby muc korzystać z plików zewnętrznych w programie, musimy zdefiniować strumień płynący s pliku i do pliku. Konieczne jest dodanie dyrektywy preprocesora #include<fstream> , gdzie jak wiecie f od file, stream to strumień, stąd fstream. Fstream to biblioteka (pakiet funkcji, procedur), obsługująca pliki.

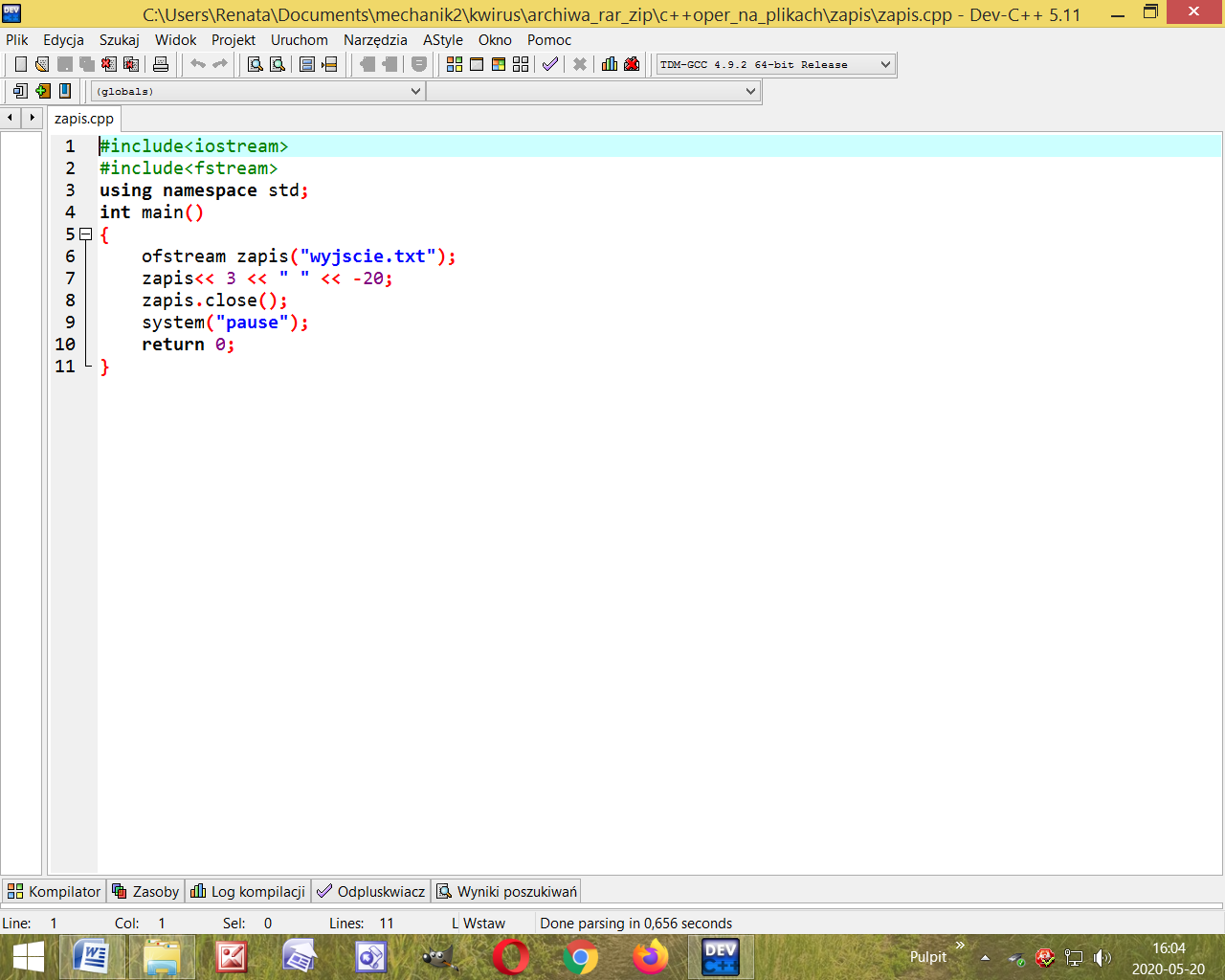
Zapis do pliku

Aby zapisać jakieś dane do pliku *wyjscie.txt*, musimy zdefiniować obiekt np*. zapis*, który pozwoli nam to zrobić. Jak to zrobić?.

Piszemy **ofstream zapis(„wyjscie.txt”);**

W istocie taki zapis to deklaracja strumienia o nazwie zapis, będącego obiektem klasy ofstream. O klasach i obiektach jeszcze nie mówiliśmy, więc przyjmijmy, że obiekt można kojarzyć ze zmienną, a klasę z typem tej zmiennej. Plik wyjściowy może mieć dowolne rozszerzenie.

Zadanie – Chcemy do pliku wyjscie.txt zapisać dwie liczby oddzielone spacją. Jak to zrobimy?



Plik wyjsce.txt pojawi się w folderze w którym został uruchomiony program.

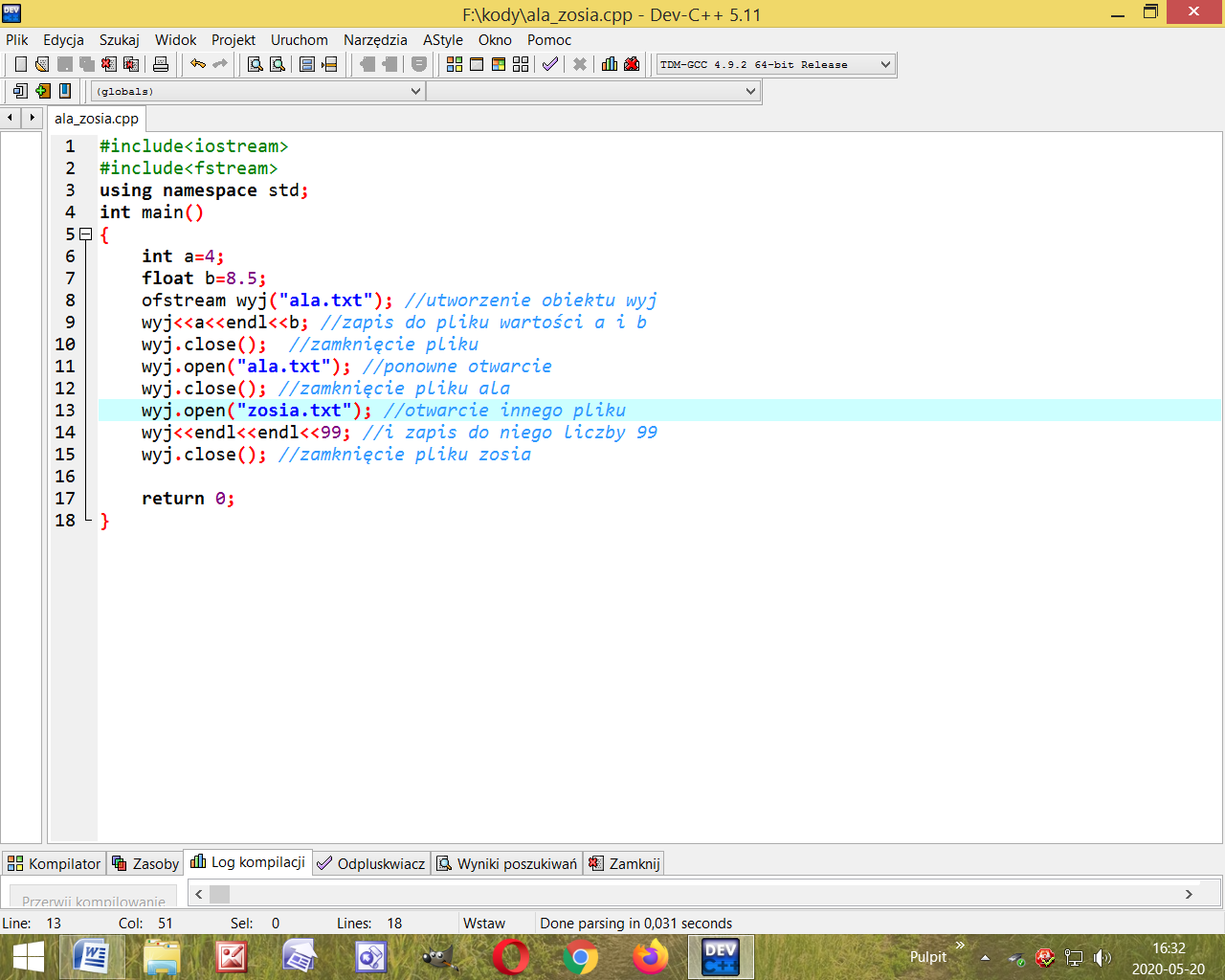
Funkcje obsługujące pliki uruchamiamy przez podanie nazwy strumienia (autostrady, którą płyną dane) i po kropce nazwy funkcji.

**zapis.close()** oznacza, że funkcja close() zapisze na dysku i zamknie plik powiązany z obiektem (zmienną) zapis. Obiekt zapis klasy ofstream po użyciu close nadal istnieje w naszym programie i możemy go powtórnie użyć do utworzenia innego pliku.

Aby powtórnie użyć już istniejącego obiektu zapis do obsługi strumienia skierowanego do innego pliku np. obliczenia.txt, należy skorzystać z funkcji open, która otwiera plik już istniejący.

zapis.open(„obliczenia.txt”);

Pamiętajmy, możemy tak zrobić po uprzednim zamknięciu funkcją close pliku, z którym wcześniej był powiązany obiekt zapis. Zobaczmy to na przykładzie:

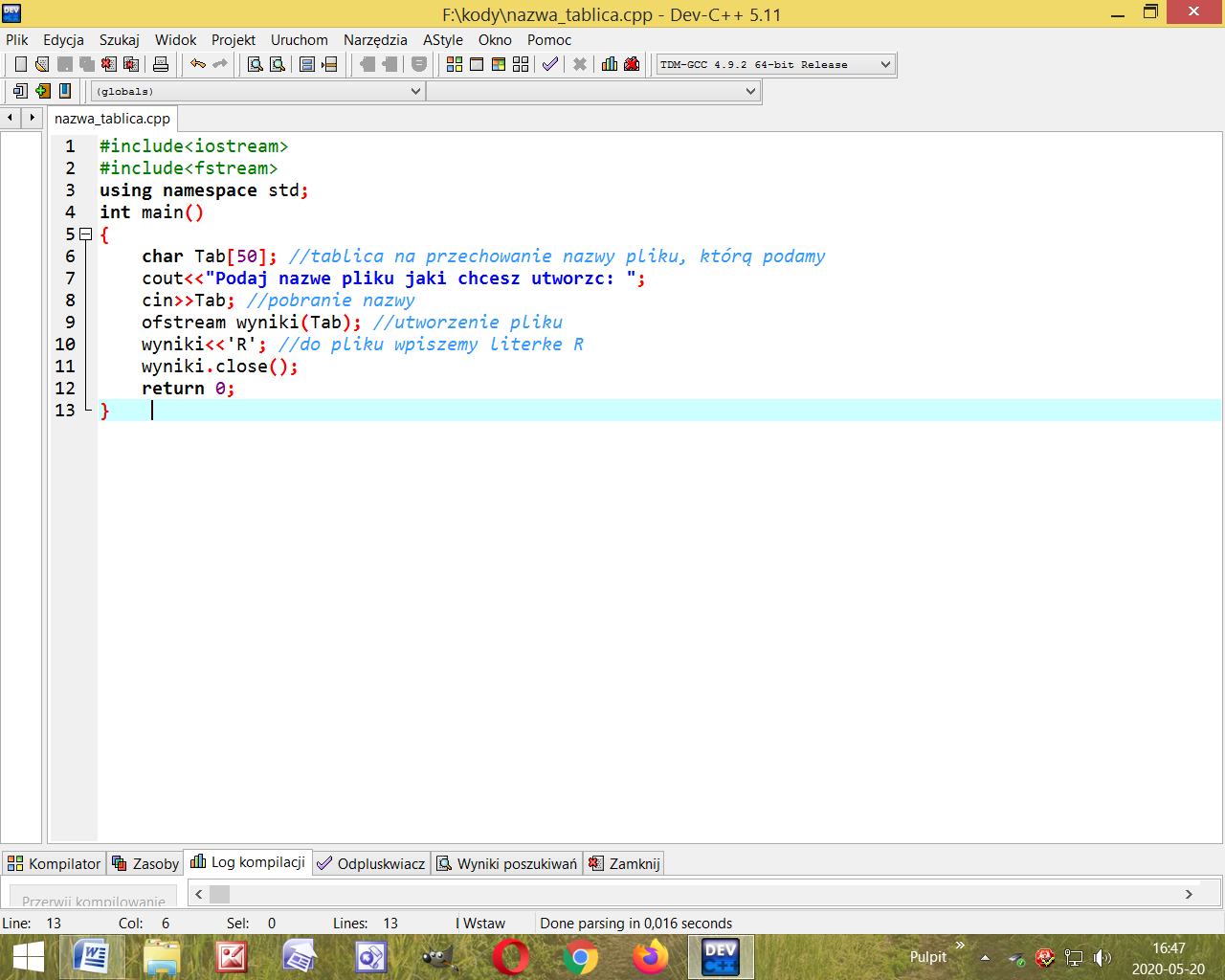


Komentarza wymaga linia 9, gdzie do pierwszego wiersza pliku ala.txt wpisujemy 4, a do drugiego wiersza ( bo jest endl) liczbę 8.5. Zauważmy, że zmienne nie muszą być tego samego typu!

Ostatecznie wynik działania tego programu to dwa pliki: ala.txt – pusty i zosia.txt – w którym w trzecim wierszu jest 99.

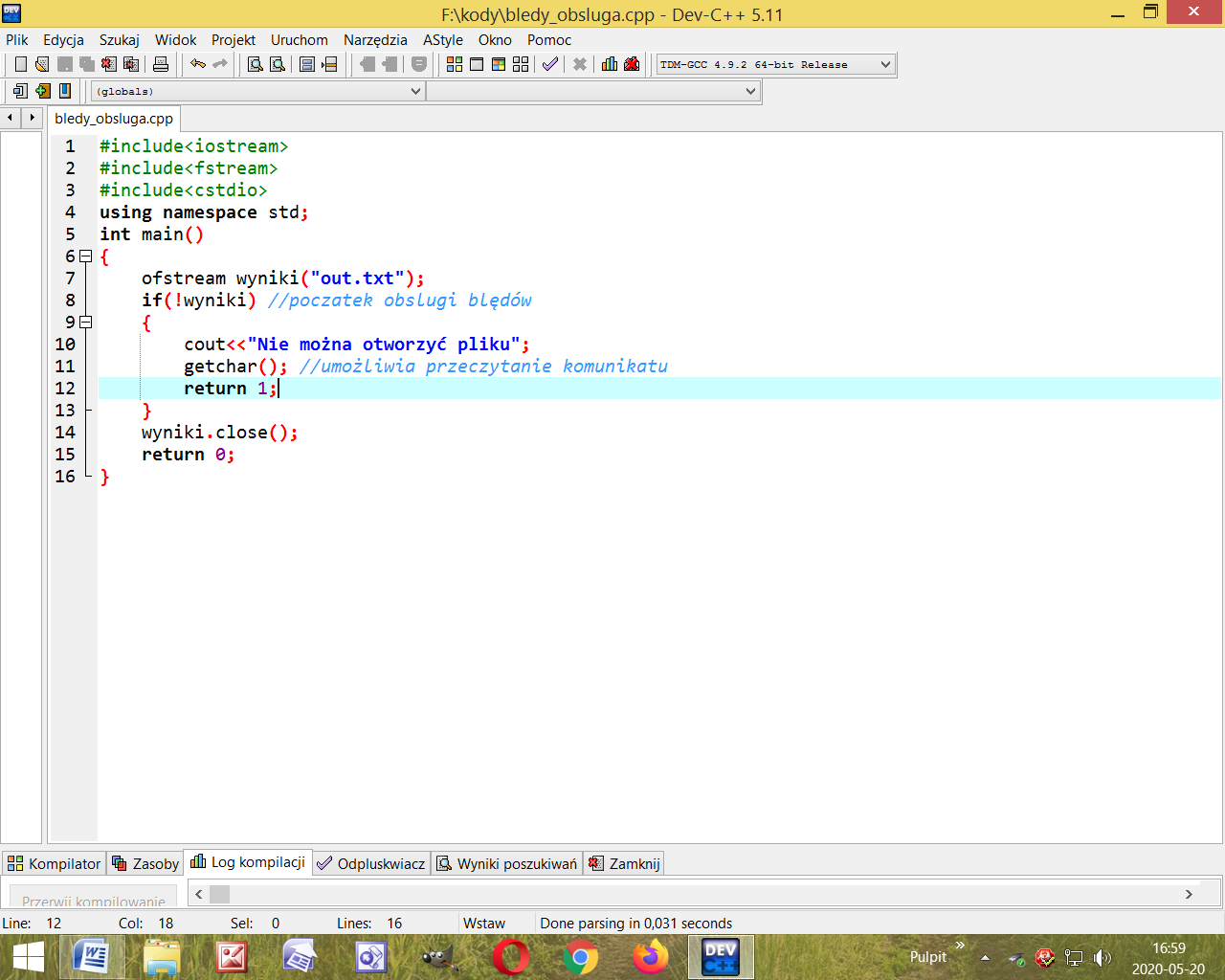
**Nieuchronną konsekwencją ponownego otwarcia pliku za pomocą obiektu klasy ofstream jest skasowanie całej jego zawartości. Stąd w pliku ala.txt pusto.**

Program w którym nazwę tworzonego pliku podamy w trakcie działania programu do tablicy znakowej.



W folderze w którym będzie zapisany i wykonany program utworzy się plik o podanej nazwie w którym będzie zapisana litera R.

Podczas prób zapisu do pliku lub otwarcia pliku mogą pojawiać się błędy. Napiszemy program, który obsłuży nam ewentualne błędy, sprawdzi czy plik istnieje.

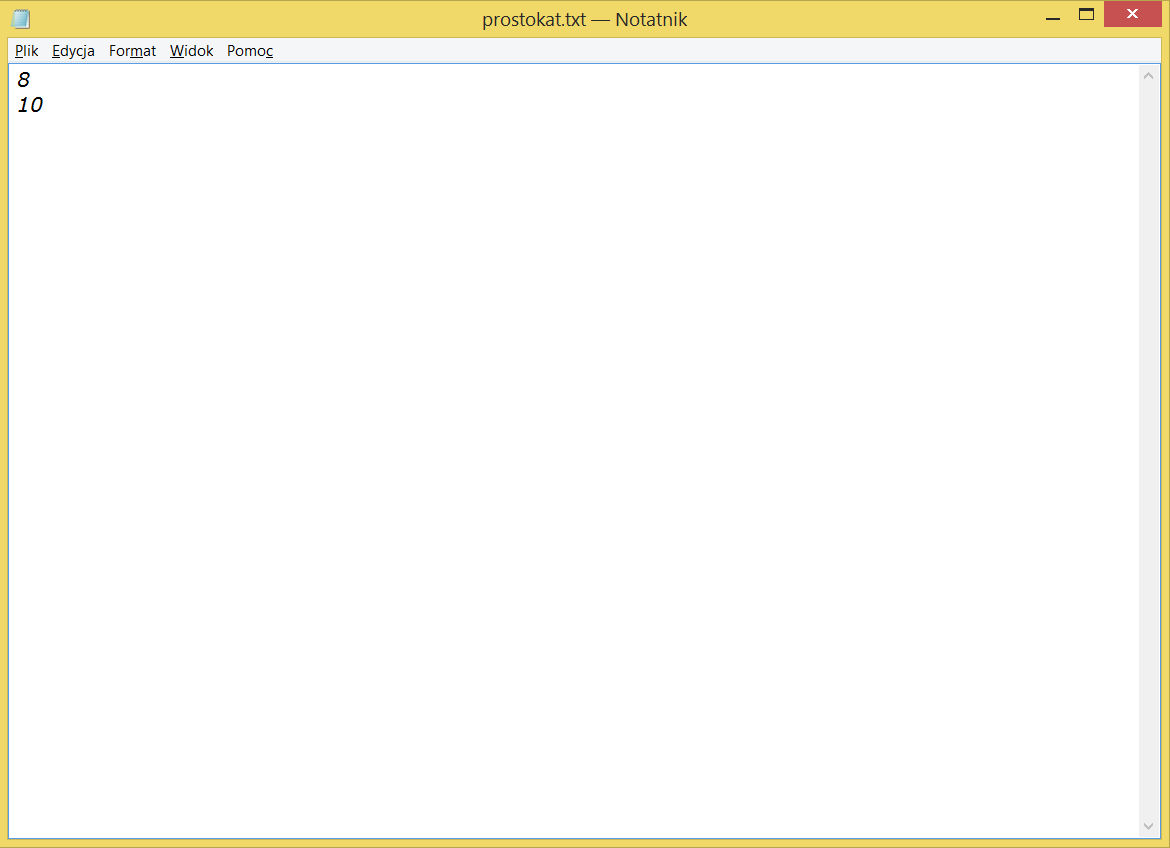


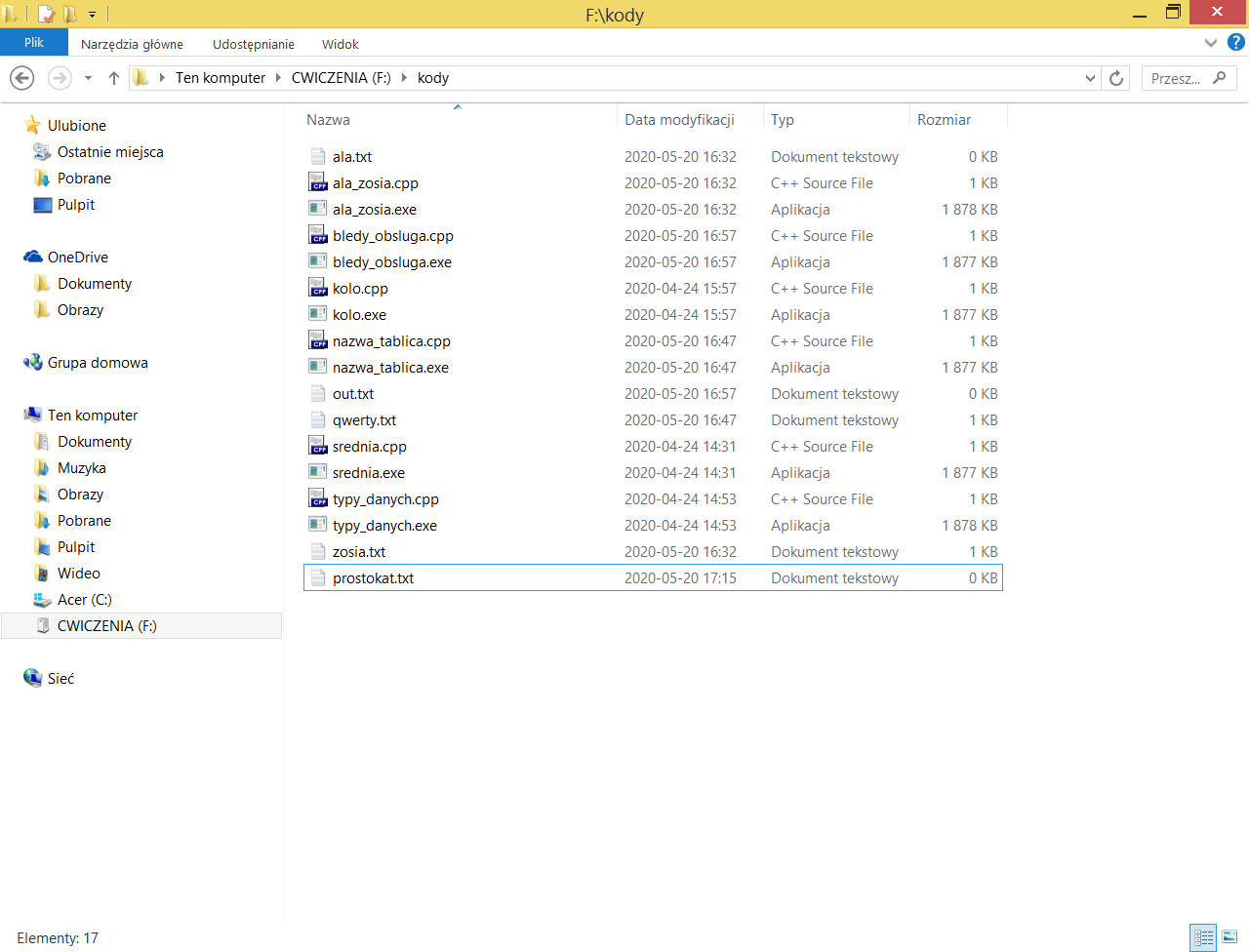
Program ten tylko otworzy istniejący plik lub utworzy nowy o nazwie out.txt i go zamknie. Niczego innego nie robi, a plik out.txt jest w środku pusty.

**Odczyt danych z pliku**

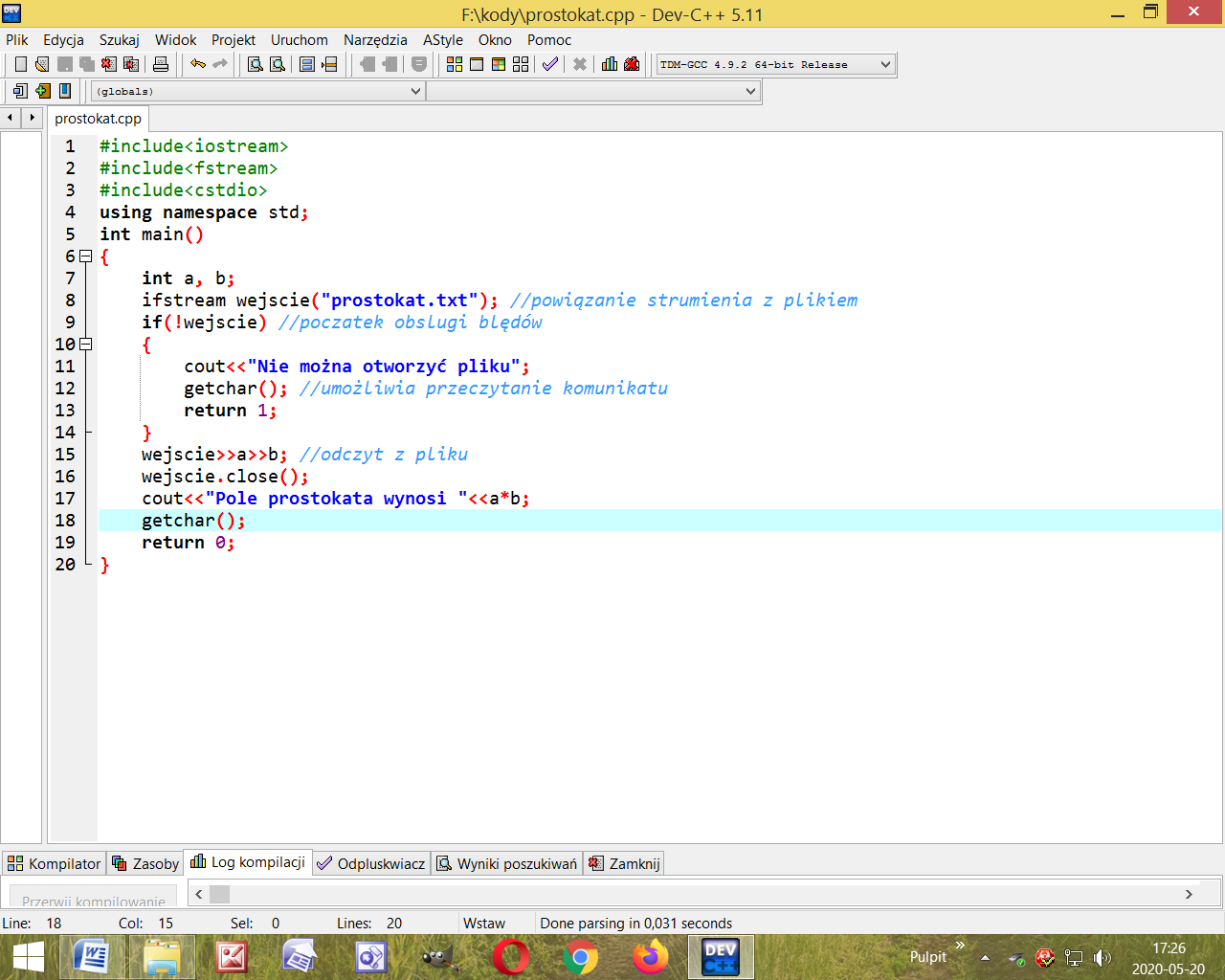
Aby odczytać dane z pliku musimy podobnie jak przy zapisie – zdefiniować obiekt (strumien) skojarzony z plikiem, z którego chcemy czytać. Służy do tego klasa ofstream, które też jest zdefiniowana w bibliotece ofstream.

Utwórzmy w folderze gdzie zapisujemy programy plik prostokat.txt, w którego pierwszym wierszu zapiszemy cyfrę 8, a w drugim 10.

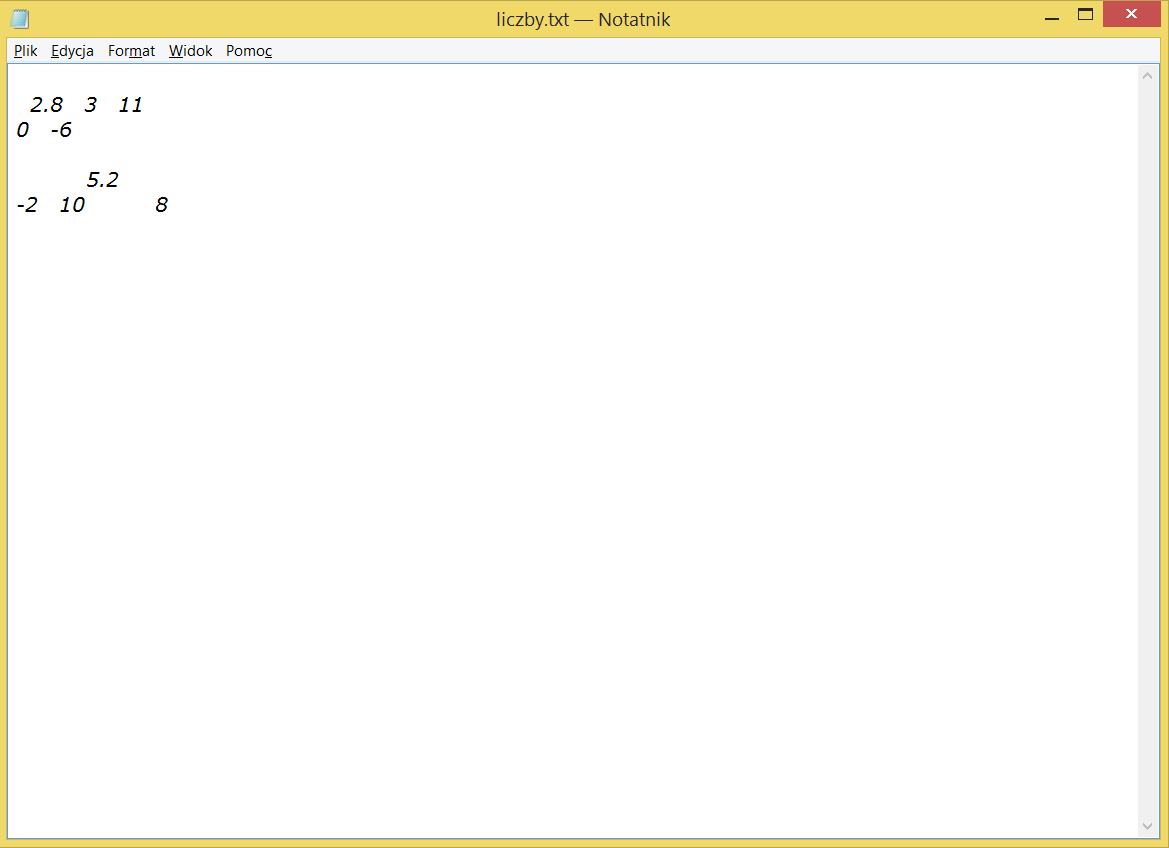




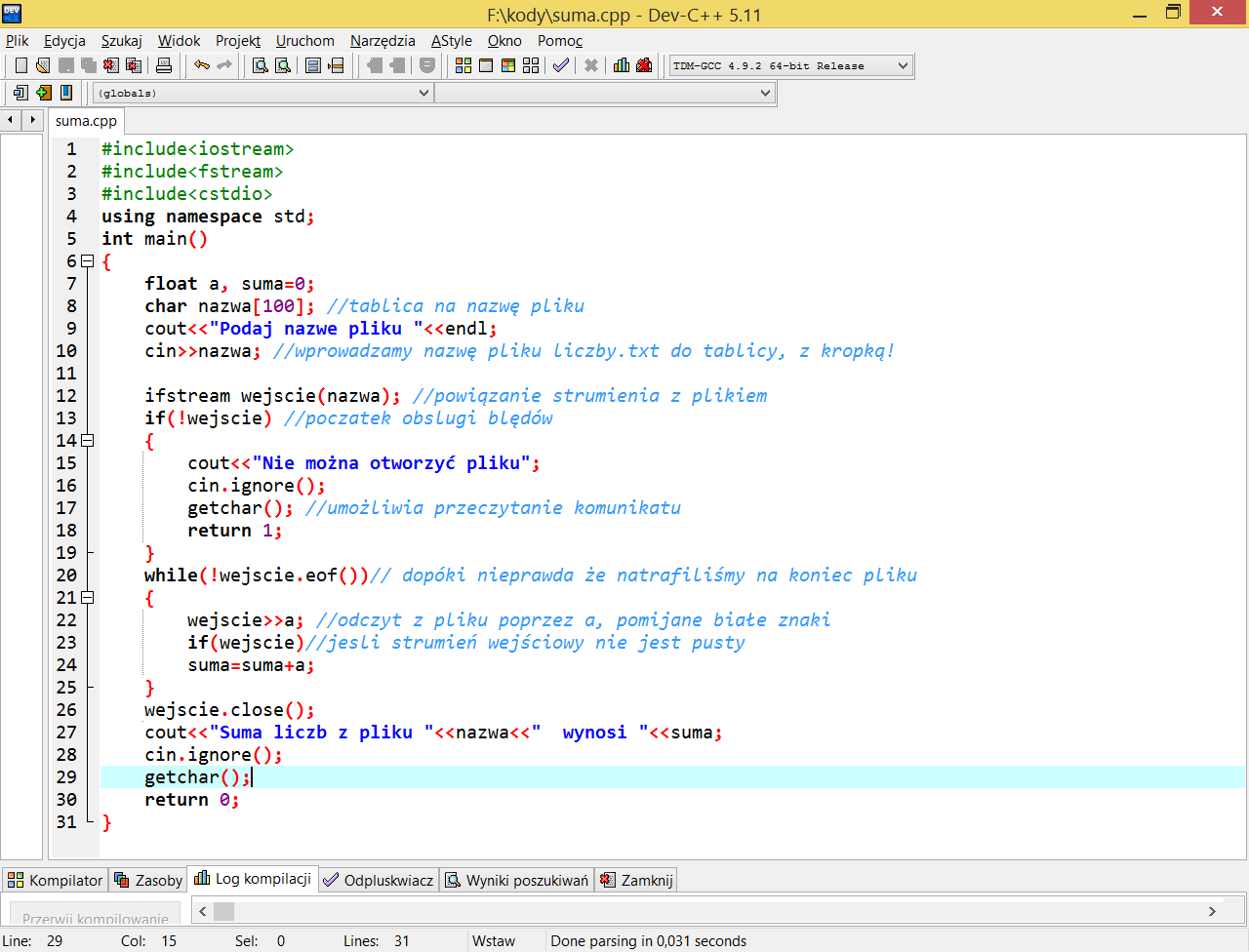
Przykład1 – napiszemy program, który odczyta dwie liczby zapisane w pliku i obliczy pole prostokąta o długościach boków równych tym liczbom. Napiszemy też obsługę błędów, gdyby nie powiodła się próba odczytu danych.



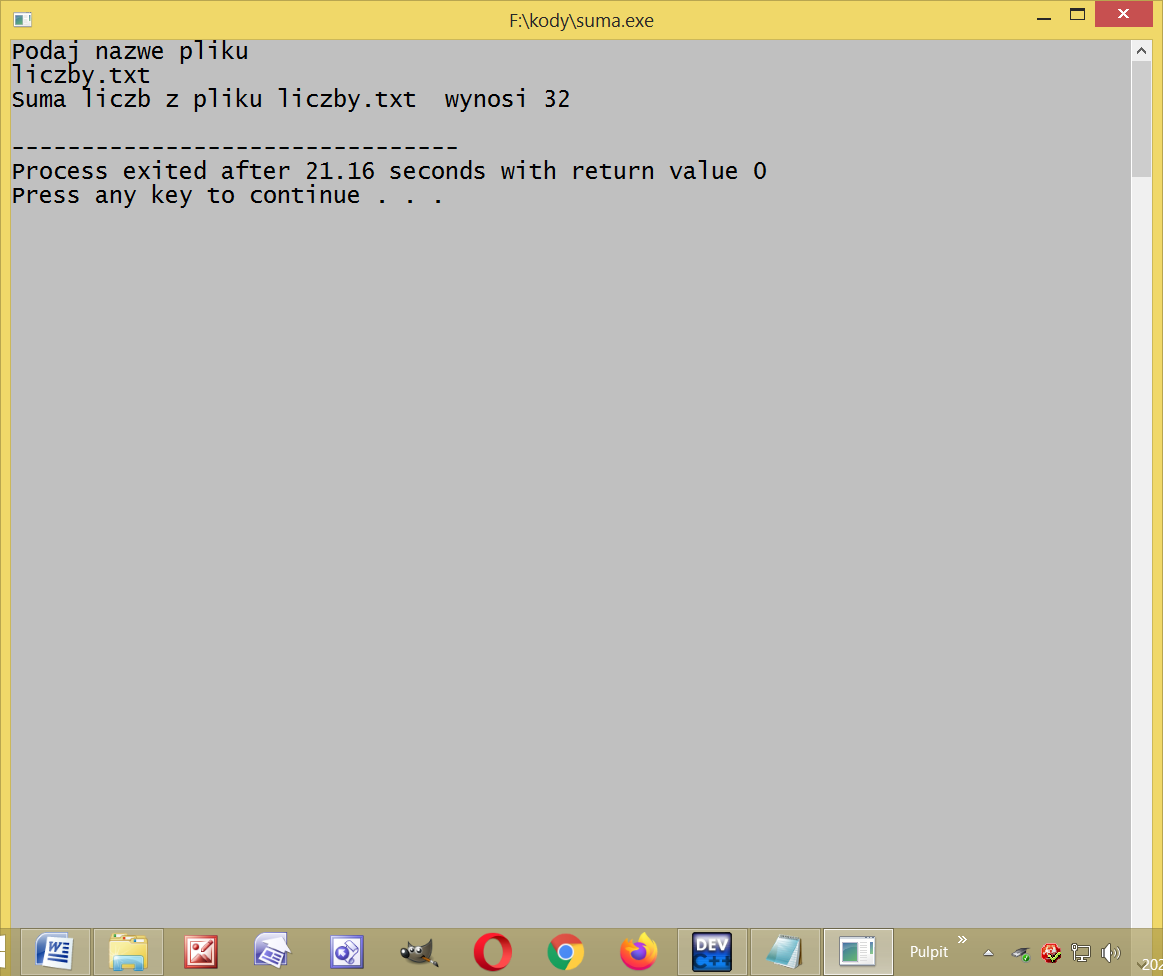
Przykład2 – **Obliczmy sumę liczb** typu float zapisanych w pliku liczby.txt. Plik ma dość nieuporządkowaną strukturę. Są spacje i entery. Plik liczby.txt , który należy utworzyć w folderze z programem wygląda tak:



Oto realizacja przykładu. Pojawi się tu nowa funkcja eof(), która przyjmuje wartość logiczną „prawda” jeśli natrafimy na koniec pliku - eof (end of file) – koniec pliku.

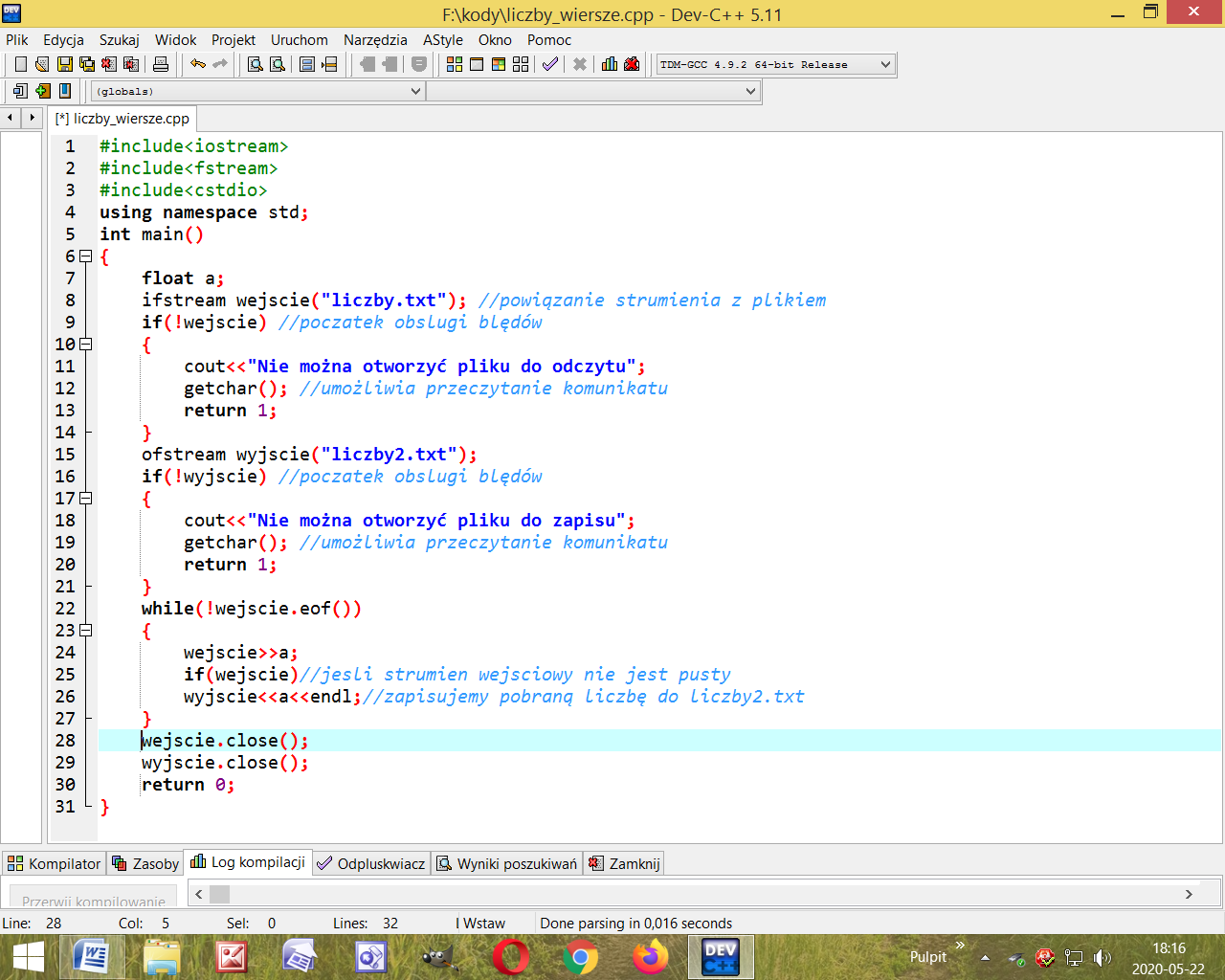


Wynik działania programu:

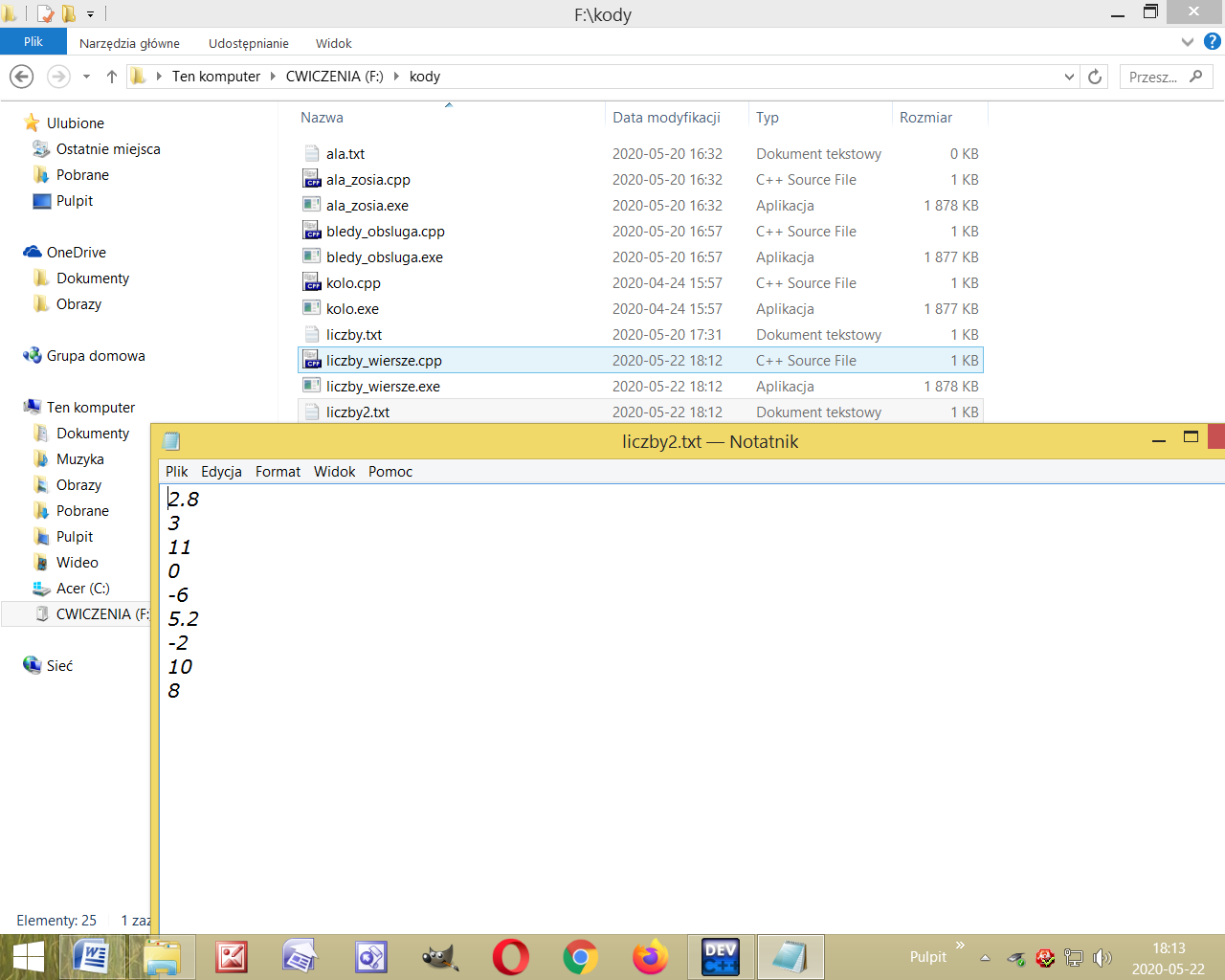


Sprawdźcie co się stanie jak podacie nazwę pliku którego nie ma w folderze.

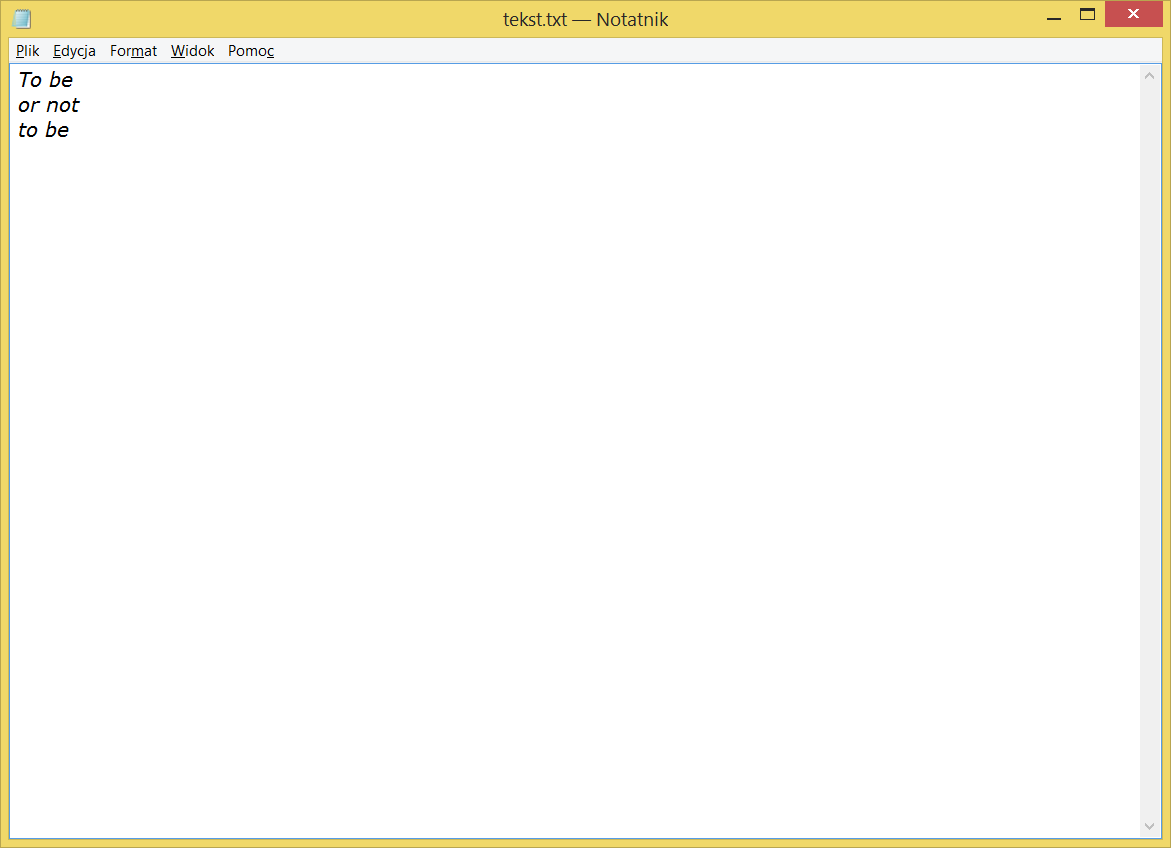
Przykład3 – Napiszemy program, który przepisze zawartość pliku liczby.txt z poprzedniego przykładu do pliku liczby2.txt w taki sposób, żeby każda kolejna liczba była w osobnym wierszu pliku wynikowego. W tym programie będziemy obsługiwać przypadki, gdy nie da się otworzyć pliku, z którego chcemy czytać, lub pliku do którego chcemy zapisywać.



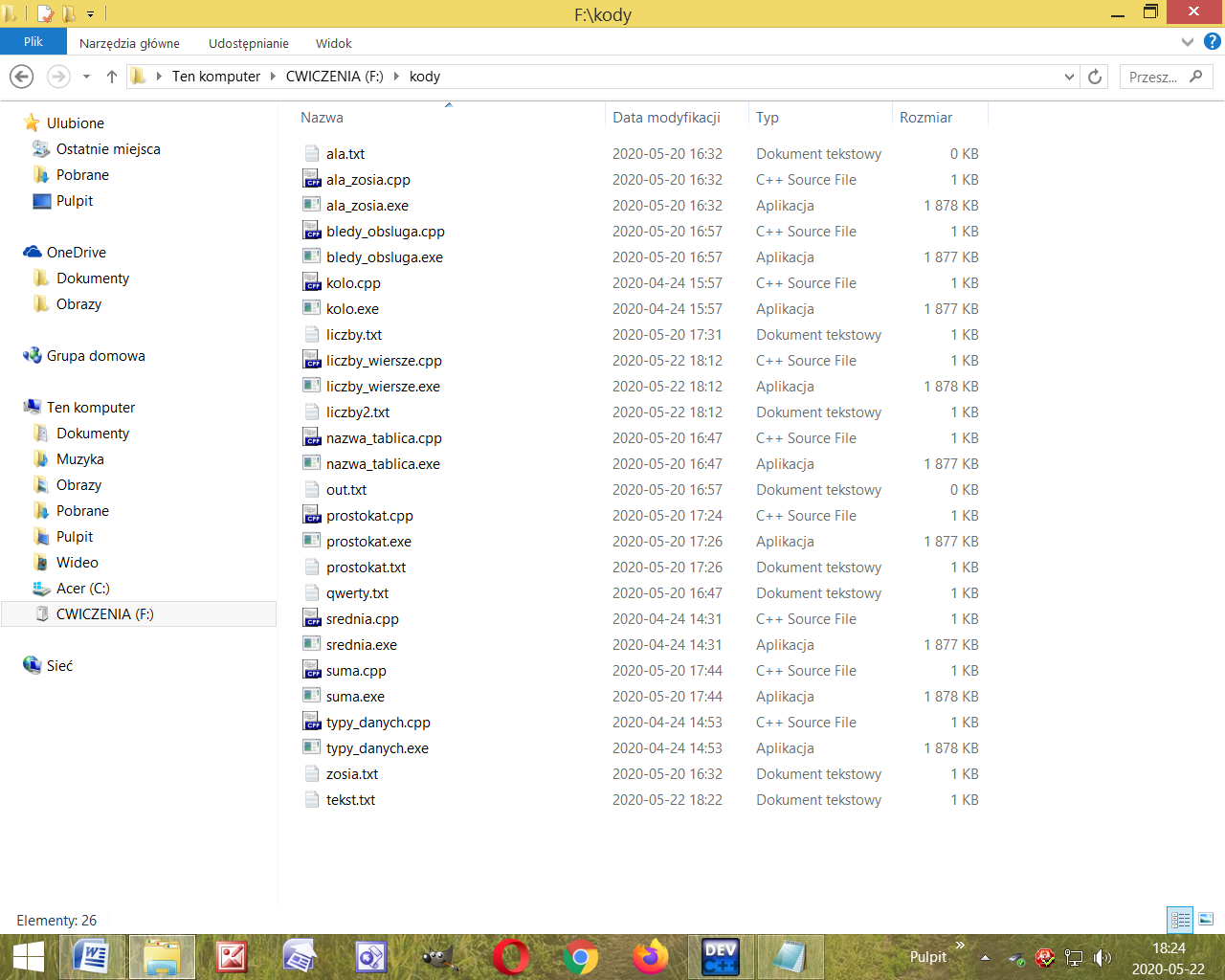
Jeśli wszystko pójdzie dobrze otrzymamy w folderze, gdzie zapisaliśmy program i był tam plik z liczbami liczby.txt, nowy plik o nazwie liczby2.txt z wpisanymi liczbami pobranymi z pierwszego pliku, każda liczba w osobnym wierszu:



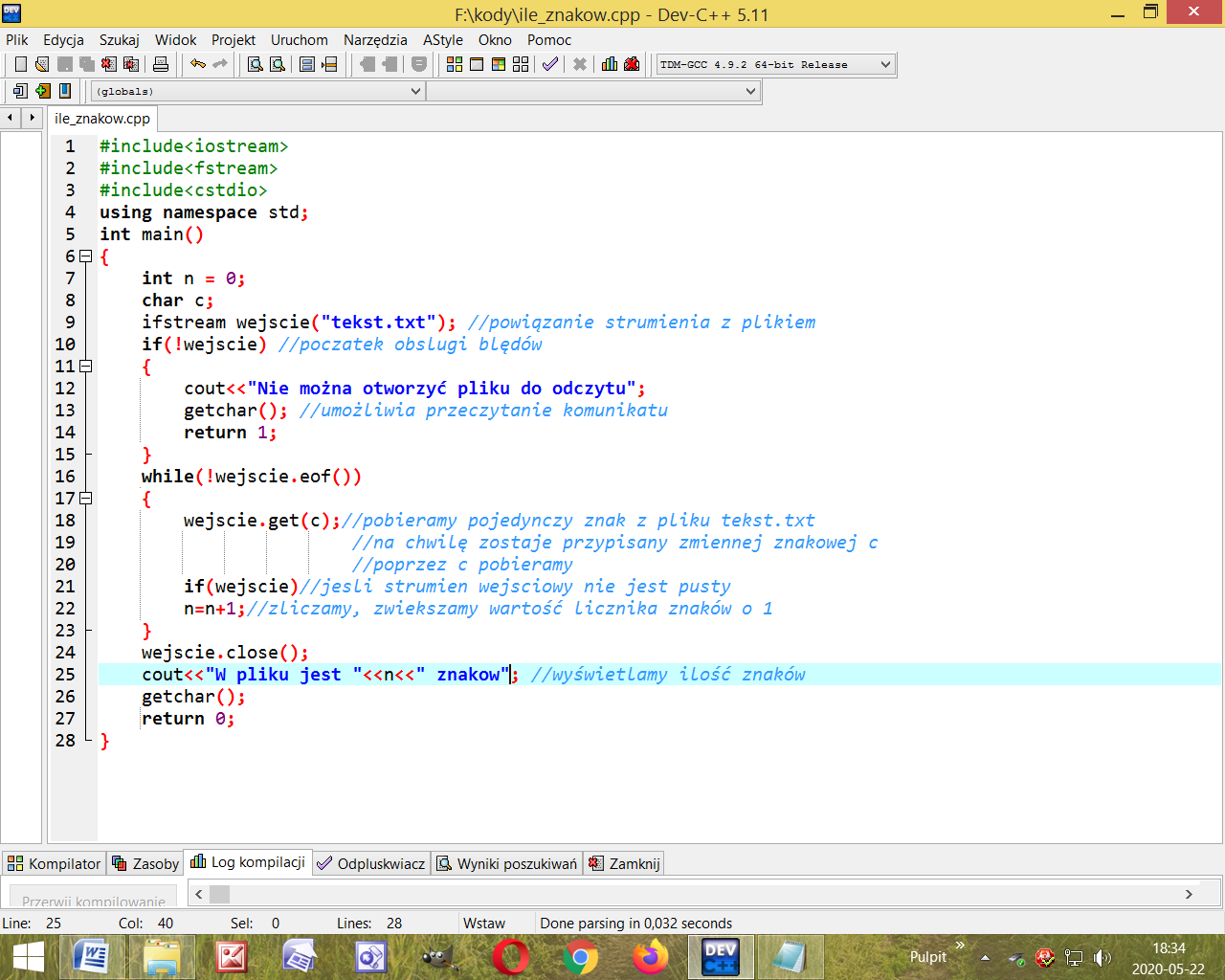
Przykład4 – Napiszemy program, który zlicza liczbę znaków występujących w pliku tekst.txt. Wykorzystamy funkcję get(), która pobiera pojedynczy znak. Zawartość pliku tekst.txt (po każdym wierszu jest enter, po ostatnim też)



W trzeciej linijce naciskamy enter , każdy wiersz musi kończyć się znakiem końca wiersza, czyli enterem. Plik oczywiście zapisujemy w folderze z kodami programów, tak jak ja to robię:

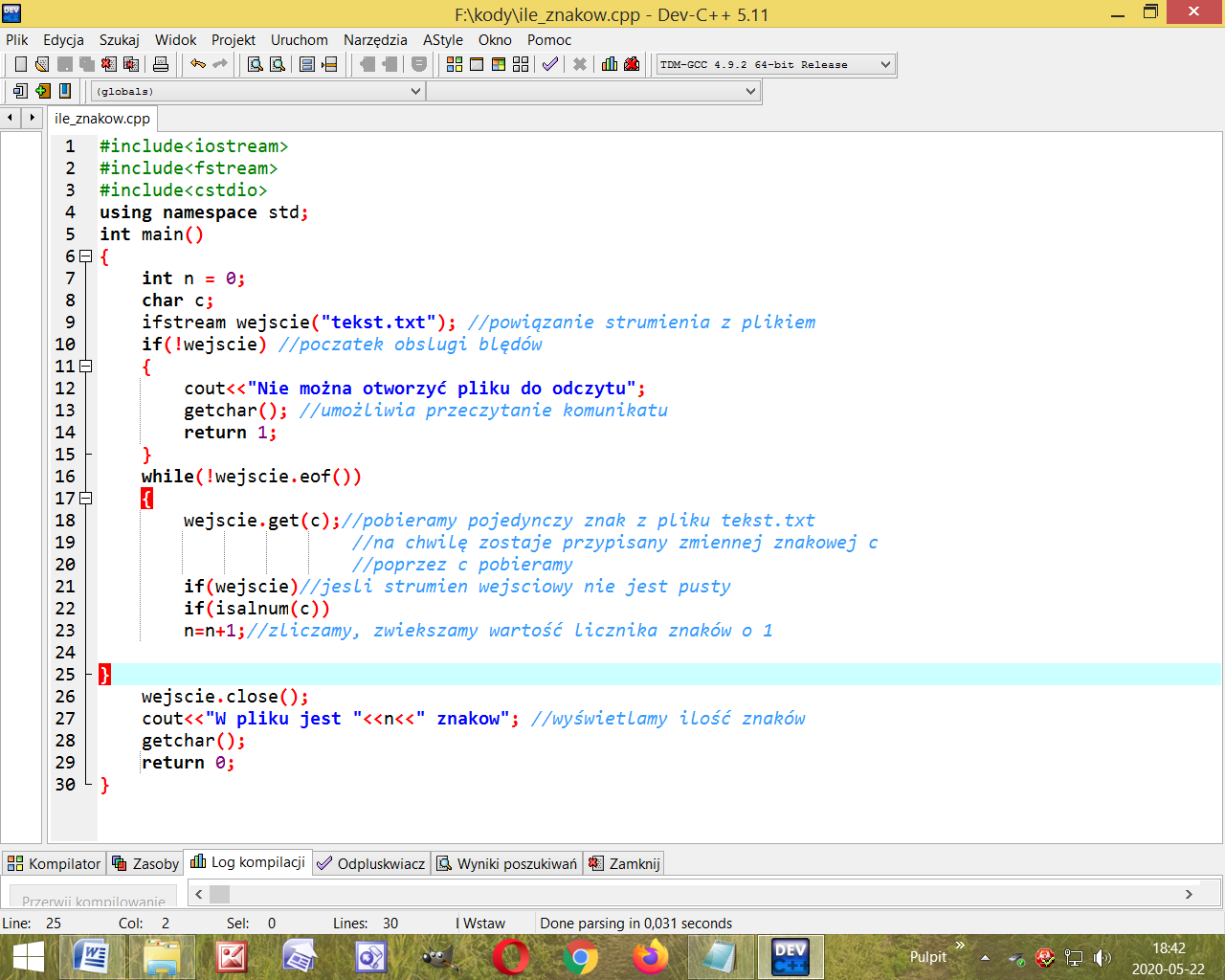


Bazując na poprzednich przykładach właściwie dodamy tylko licznik i zmienną znakową c, gdzie będziemy zliczać ilość znaków:



W wyniku dostaniemy liczbę 19 a nie 13 – bo tyle jest widocznych znaków. Jeśli chcemy policzyć tylko znaki z pominięciem białych znaków(spacji, tabulatorów, znaków końca lini, czy pliku), to musimy dokonać pewnych modyfikacji w pętli while, wystarczy dopisać warunek:

if (islanum(c)) n=n+1;



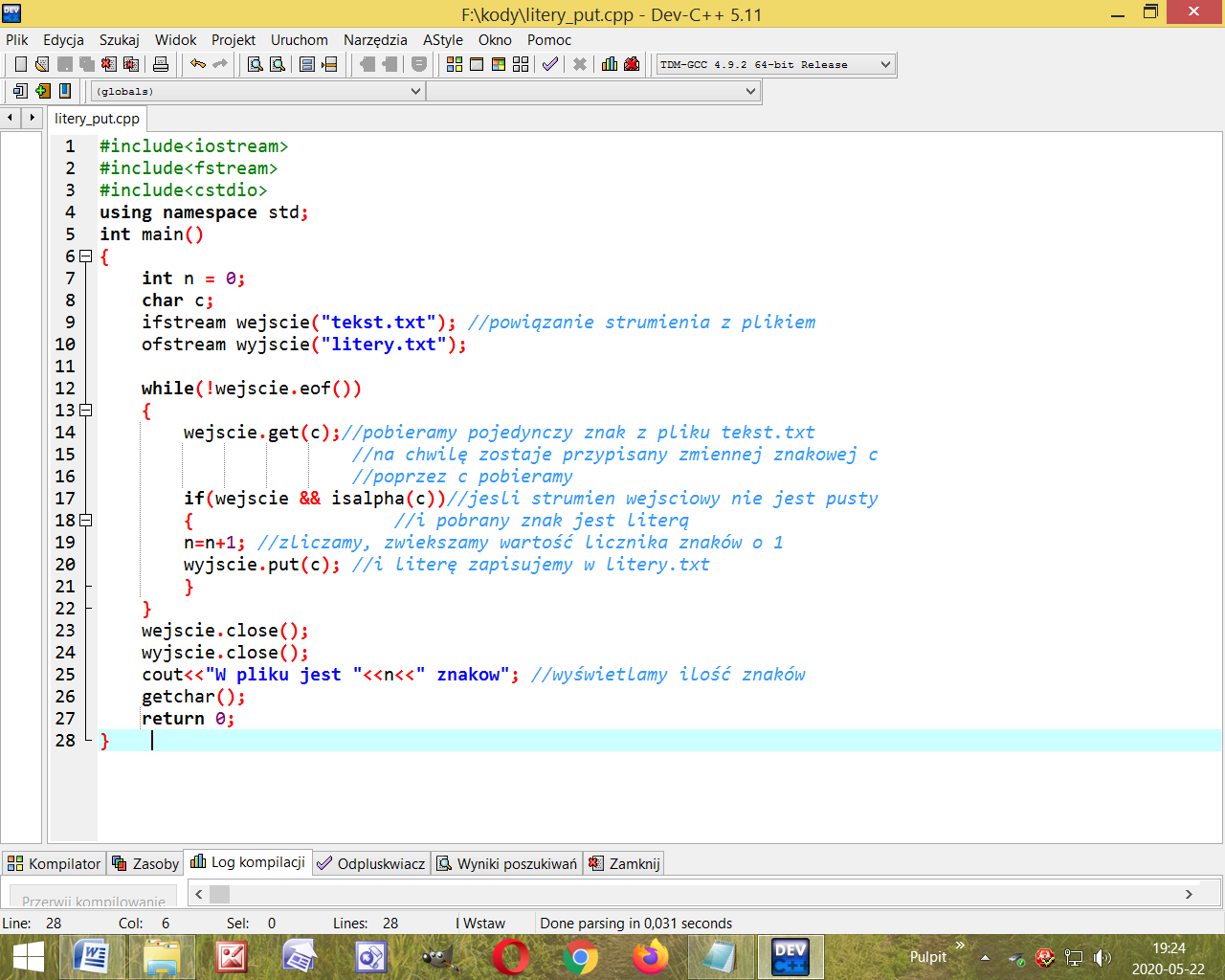
Funkcja islanum() przyjmuje true, gdy argument jest literą lub cyfrą

Podobne funkcje:

isdigit() –prawda gdy argument jest cyfrą

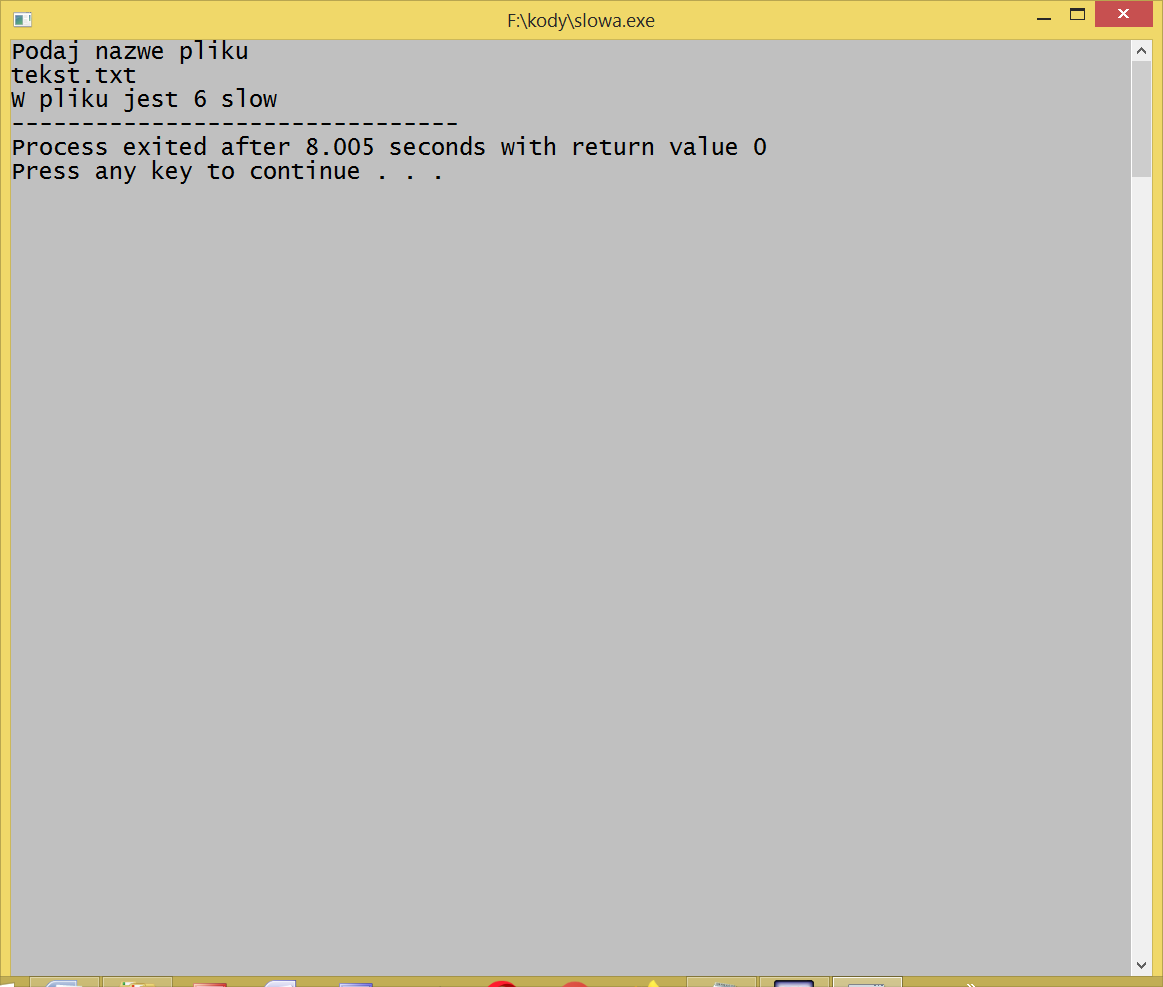
isalpha() – prawda gdy argument jest literą

Przykład5 – Litery (bez białych znaków) z pliku tekst.txt wpiszemy do nowego pliku o nazwie literki.txt. Zliczymy i wyświetlimy również liczbę liter w pliku. w programie wykorzystamy funkcję put(), która wpisuje znak po znaku poprzez zmienną c do pliku powiązanego ze strumieniem.

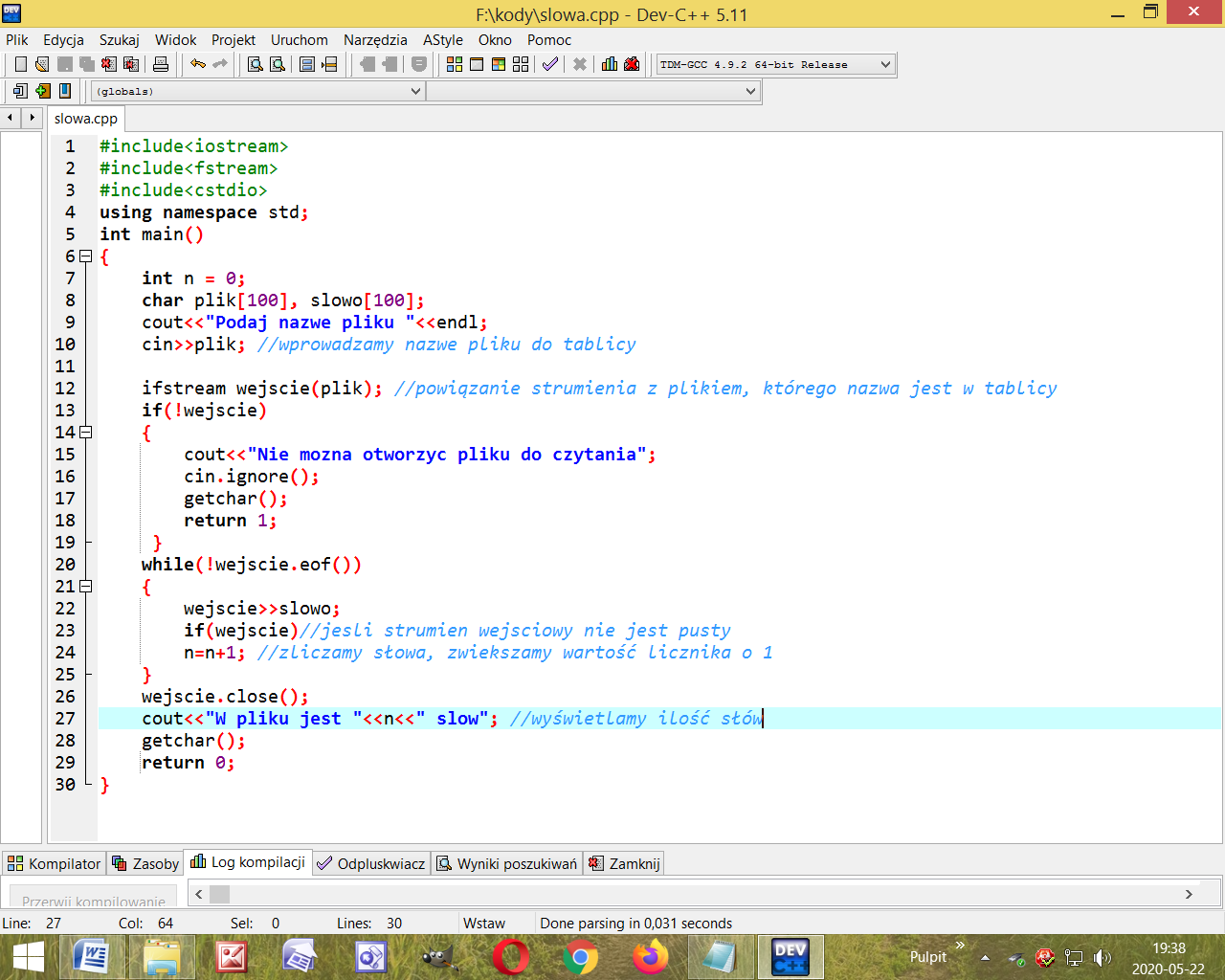


Przykład6 – Napiszemy program, który zliczy liczbę słów w pliku. Przez „słowo” rozumiemy ciąg znaków zakończony dowolnym białym znakiem. Może to być słowo które coś oznacza, a może być zwykły ciąg znaków, np.: qwerty. Musimy zadeklarować 2 tablice znakowe. Jedną na przechowanie nazwy pliku, drugą do przechowania słowa.

Wynik działania programu:



Kod źródłowy:



Myślę, że po wykonaniu tych ćwiczeń nie będzie żadnego problemu z wykonaniem zadania domowego. Ci, którzy nie zrobili – macie jeszcze tydzień na naukę do 31 maja mam mieć na poczcie rozwiązania. Pozdrawiam. Sorka