**TABLICE TEKSTOWE. ALGORYTMY NA TEKSTACH**

**Tablice tekstowe**

W języku C++ teksty przechowywane są w tablicach o elementach typu **char**. Tablicę taka deklarujemy następująco:

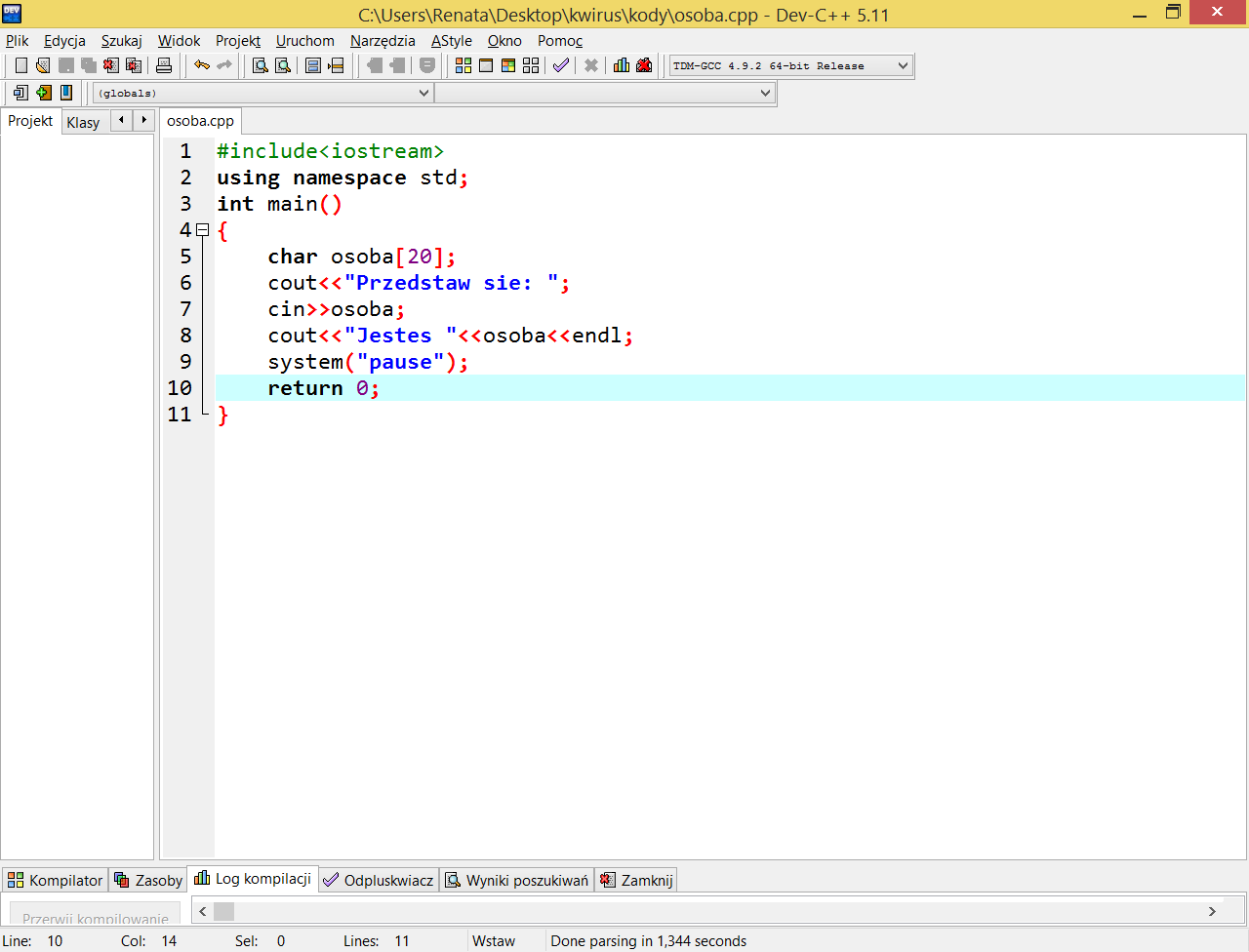
**char nazwa\_tablicy[ilość\_elementow];**

Możemy podobnie jak w przypadku tablic liczbowych zadeklarować i jednocześnie zainicjować w tej samej linii tablicę następująco:

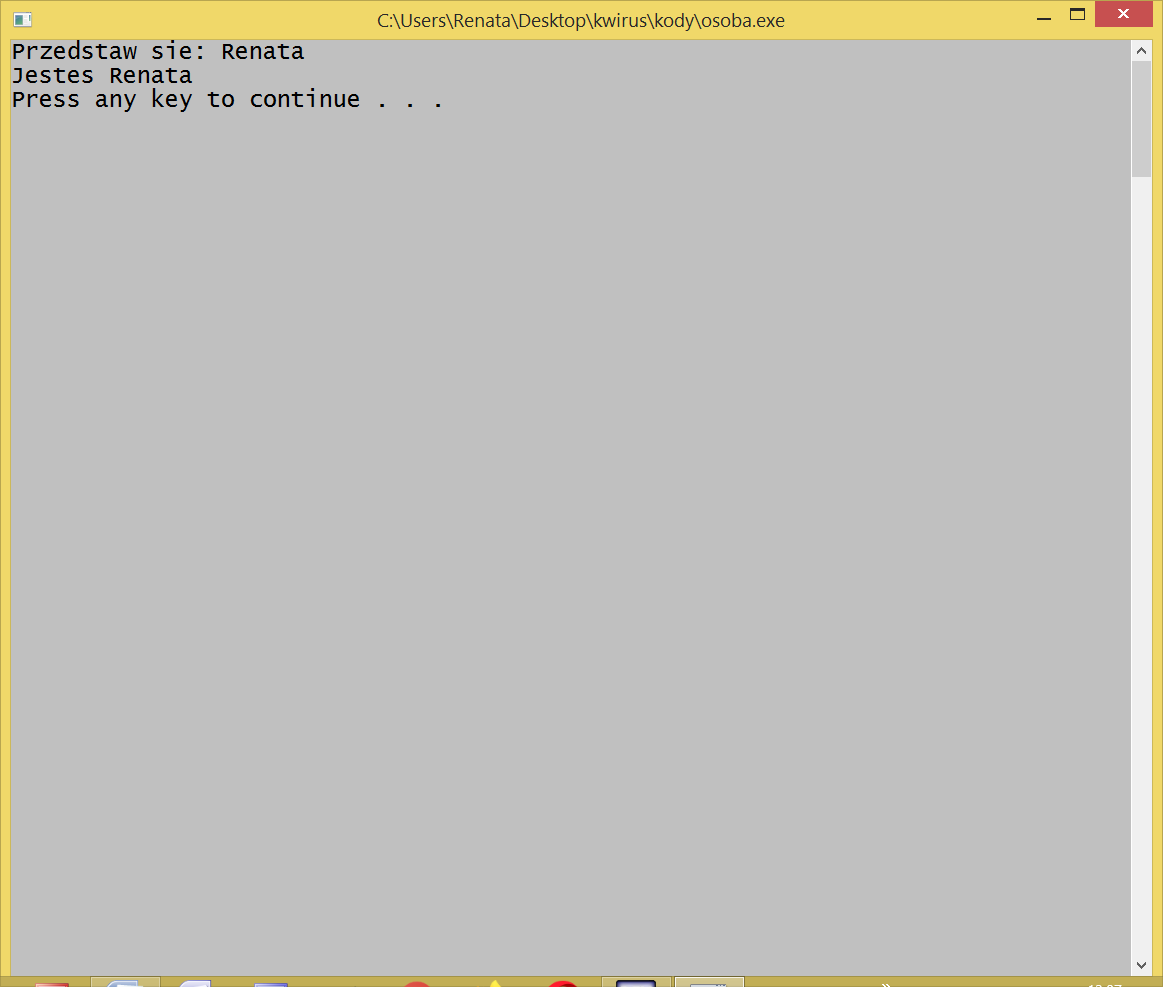
**char imie[] = „Oktawian”;**

Zadeklarowana w ten sposób tablica będzie 9-elementowa, bo Oktawian składa się z 8 liter, a jako ostatni znak jest automatycznie dodawany znacznik końca tekstu \0 ( zero, null). Dlatego, tablica która ma przechowywać tekst n-znakowy musi mieć rozmiar n+1, aby zmieścił się znak null.

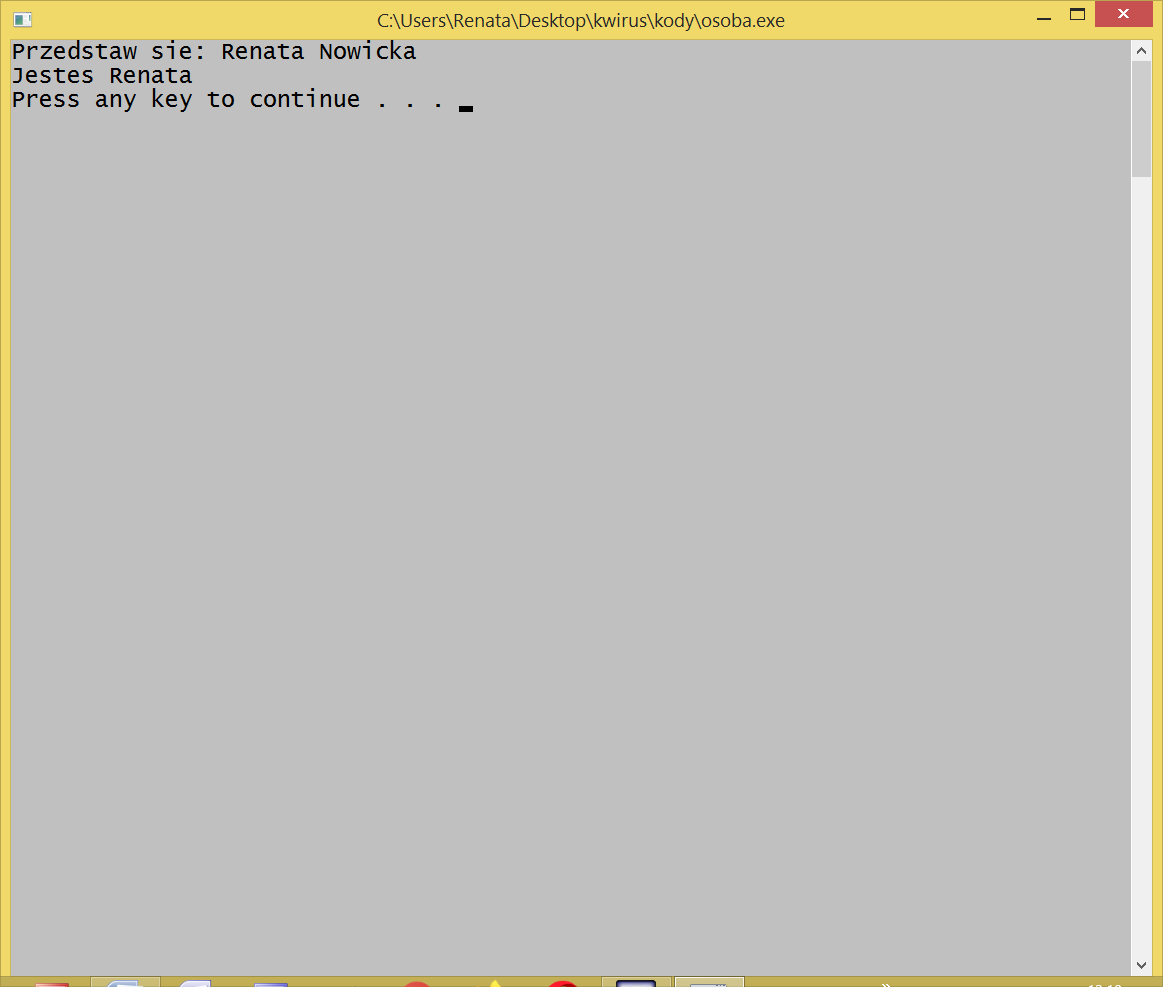
Przykład prostego algorytmu ze zmienną tablicową przechowującą tekst:



Wynik działania:



Wydaje się, że program działa poprawnie, ale tak nie jest. Co się stanie, gdy ktoś przedstawi się za pomocą imienia i nazwiska? Sprawdźmy. Wynik działania:



Zmienna osoba nie przyjęła wartości „Renata Nowicka”, tylko „Renata”. Dlaczego? W miejsce zmiennej zostanie podstawiony tylko fragment do pierwszej spacji.

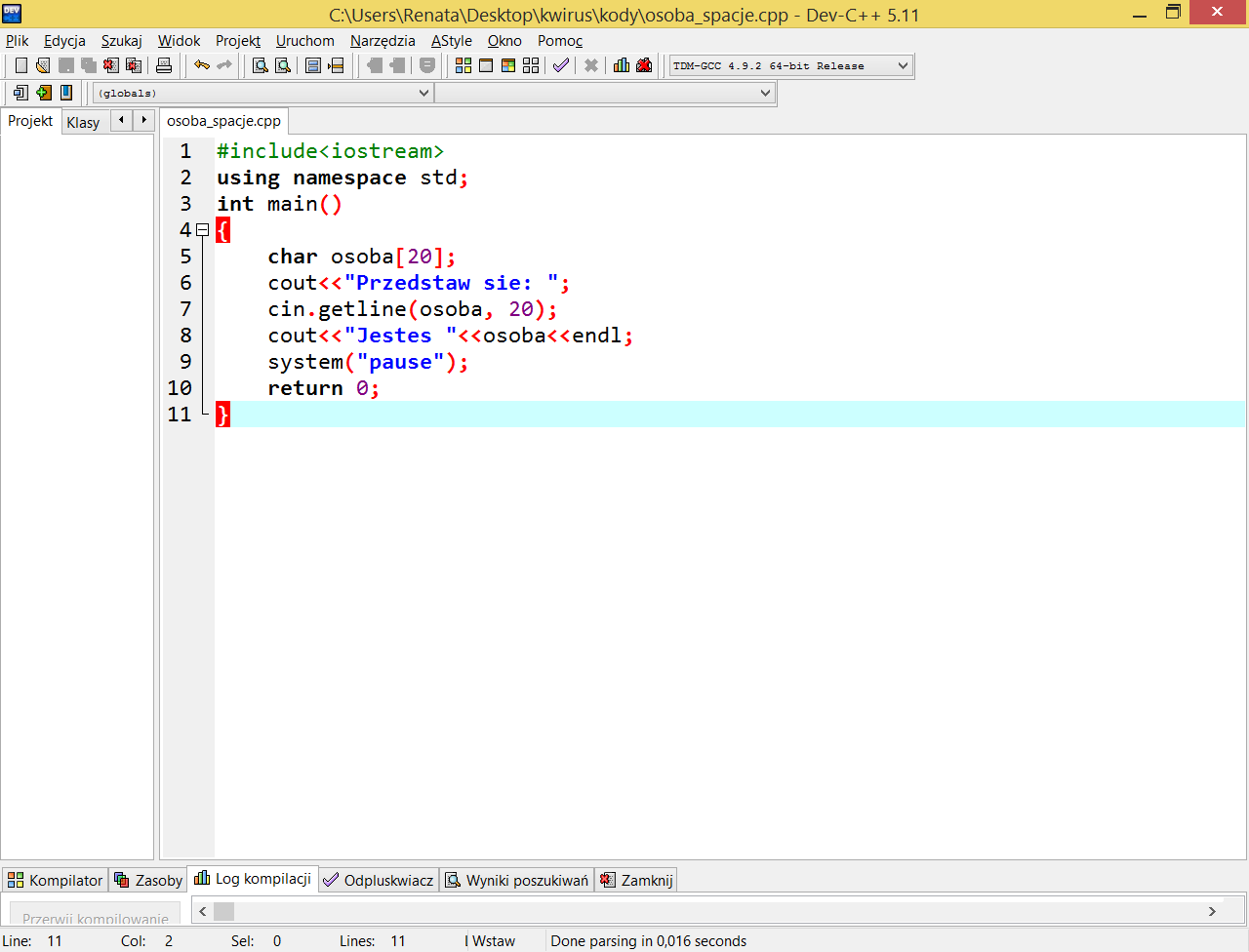
Za pomocą wyrażenia **cin>>osoba , gdzie osoba to tablica znaków, można pobrać tekst tylko do pierwszej spacji, reszta znaków jest pominięta.**

Jak w takim razie wprowadzić do tablicy znakowej tekst ze spacjami?

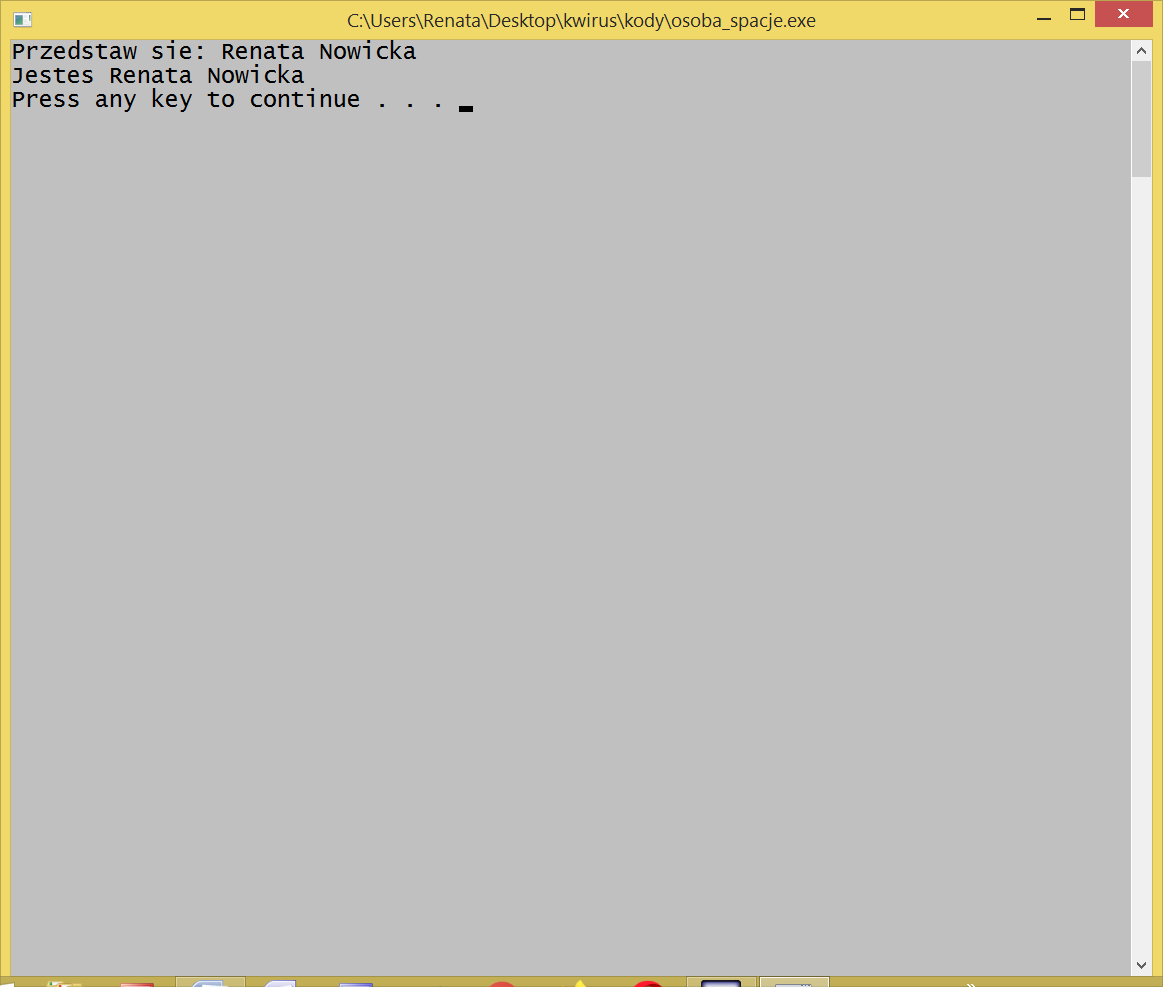
Zamiast **cin>>osoba;** należy użyć konstrukcji **cin.getline(osoba, 20);**

Przy tablicy osoba[20]; możemy oczywiście pobrać o jeden znak mniej, czyli 19, bo jak pamiętamy ostatni znak w tablicy to znak NULL, oznaczający koniec tekstu.

Napiszmy program, dzięki któremu do tablicy osoba pobierzemy tekst ze spacją:



Wynik działania instrukcji cin.getline()



**Zapamiętaj!!!**

Teksty w których nie będzie białych znaków (spacji, tabulatorów), możesz pobierać do zmiennej tablicowej (tablicy) za pomocą składni:

**cin>>nazwa\_tablicy;**

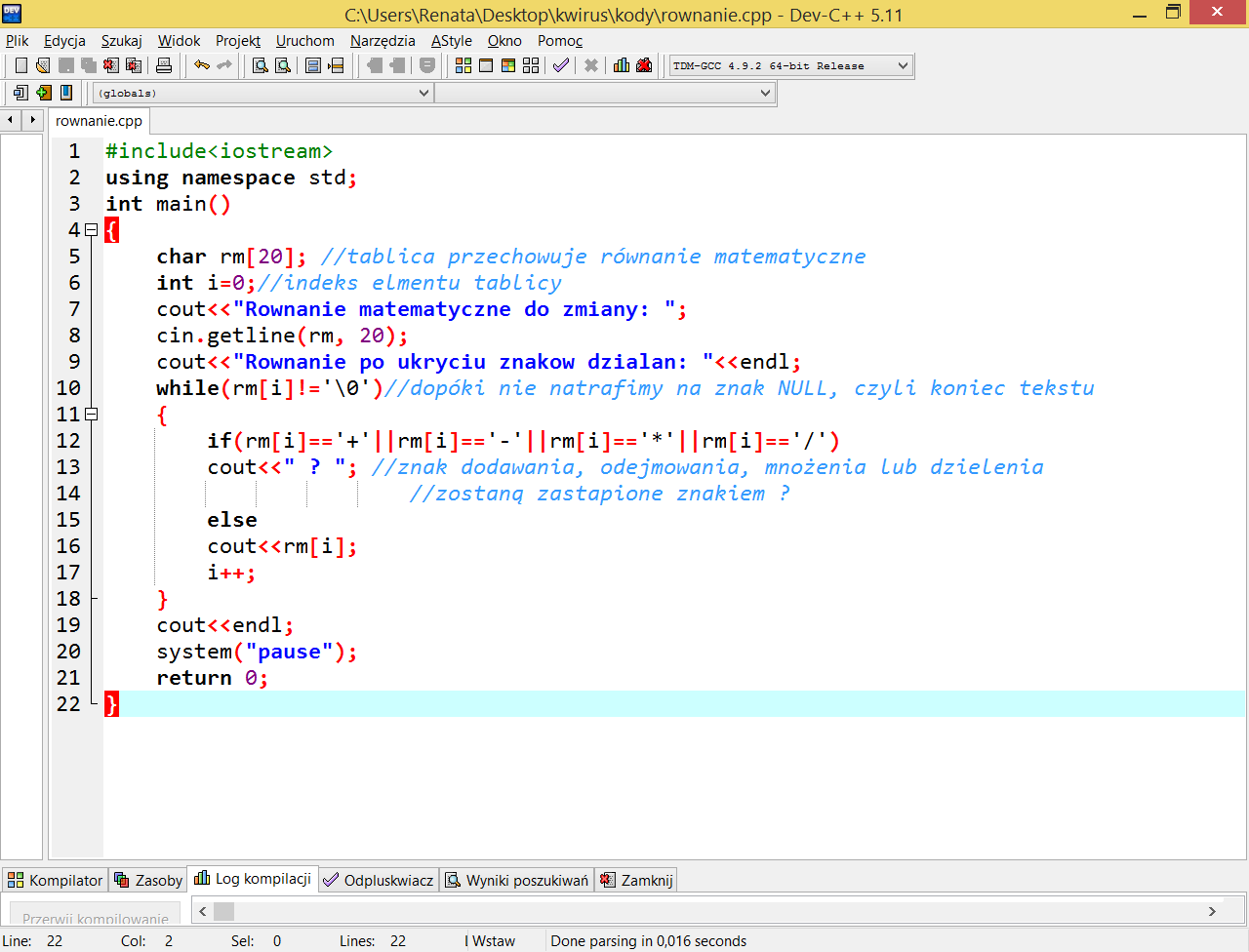
Kiedy w pobieranym ciągu znaków mogą pojawić się białe znaki, stosujemy:

**cin.getline( nazwa\_tablicy, rozmiar\_tablicy);**

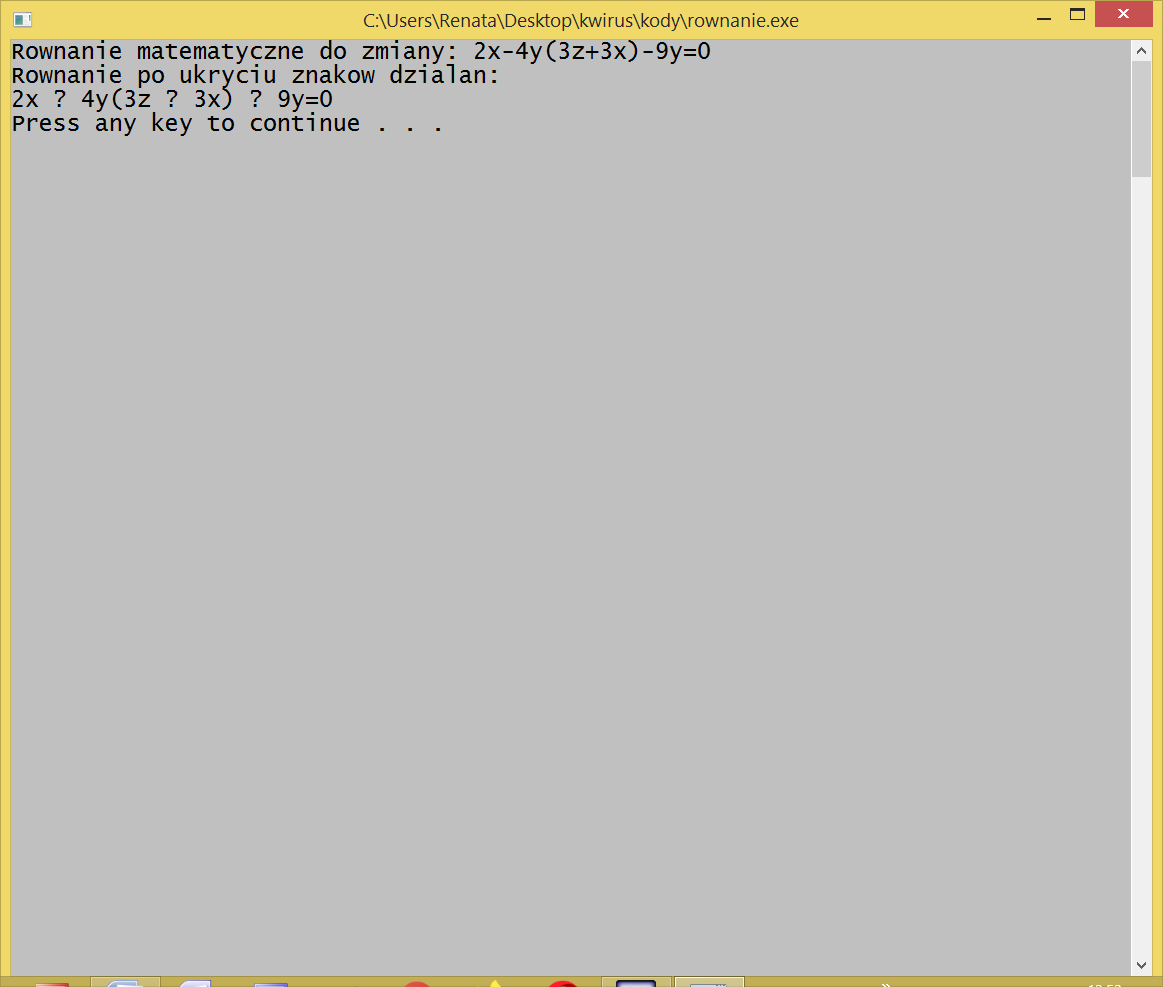
Wiemy już jak poprawnie pobierać tekst do tablicy, ale nie wiemy jeszcze jak ten tekst poprawnie odczytywać w sposób jaki my chcemy.

Jeśli chcę wypisać tablicę znak po znaku, to odczyt umieszczam w pętli, przy czym liczba obrotów pętli musi być równa liczbie wprowadzonych znaków. Po co nam odczytywanie znak po znaku? Jest potrzebne na Przykład do tego, aby każdy znak ustawić w nowej linii, zapisać tekst od tyłu (czy to palindrom), bądź dokonać operacji na znakach (np. jeśli w ciągu spotkam literę a to zamienię ją na znak +).

Przykład – Napiszmy program, który w pobranym od użytkownika równaniu matematycznym zamieni wszystkie znaki działań matematycznych na znaki zapytania i w tej postaci wypisze równanie na ekranie konsoli.



Wynik działania programu:



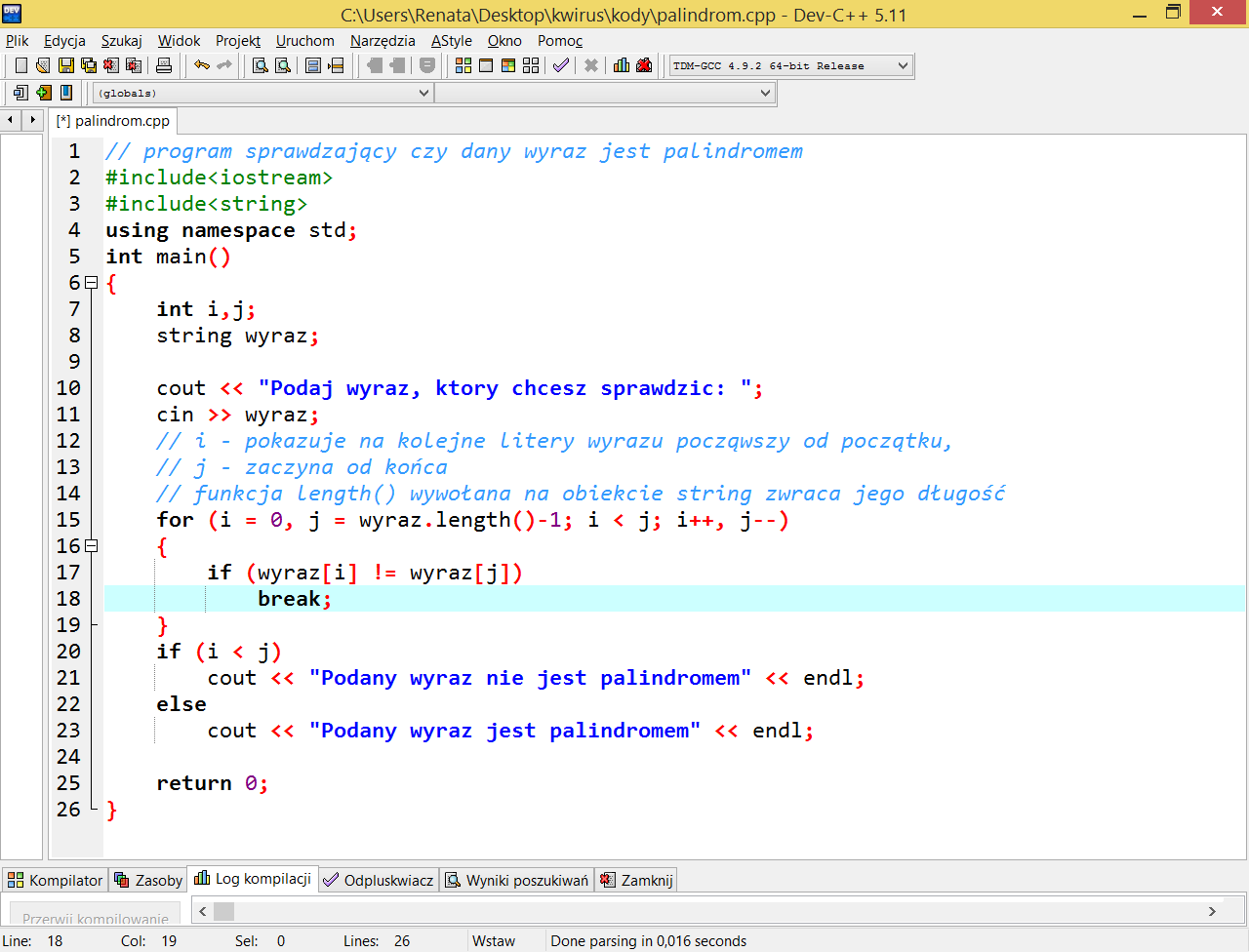
**Zapamiętaj!!!**

Jeśli zadeklarujesz tablicę znaków o długości n, to możesz do niej wprowadzić tylko n-1 znaków, gdyż ostatni z elementów tablicy to znacznik końca tekstu: \0, NULL

**ALGORYTMY NA TEKSTACH**

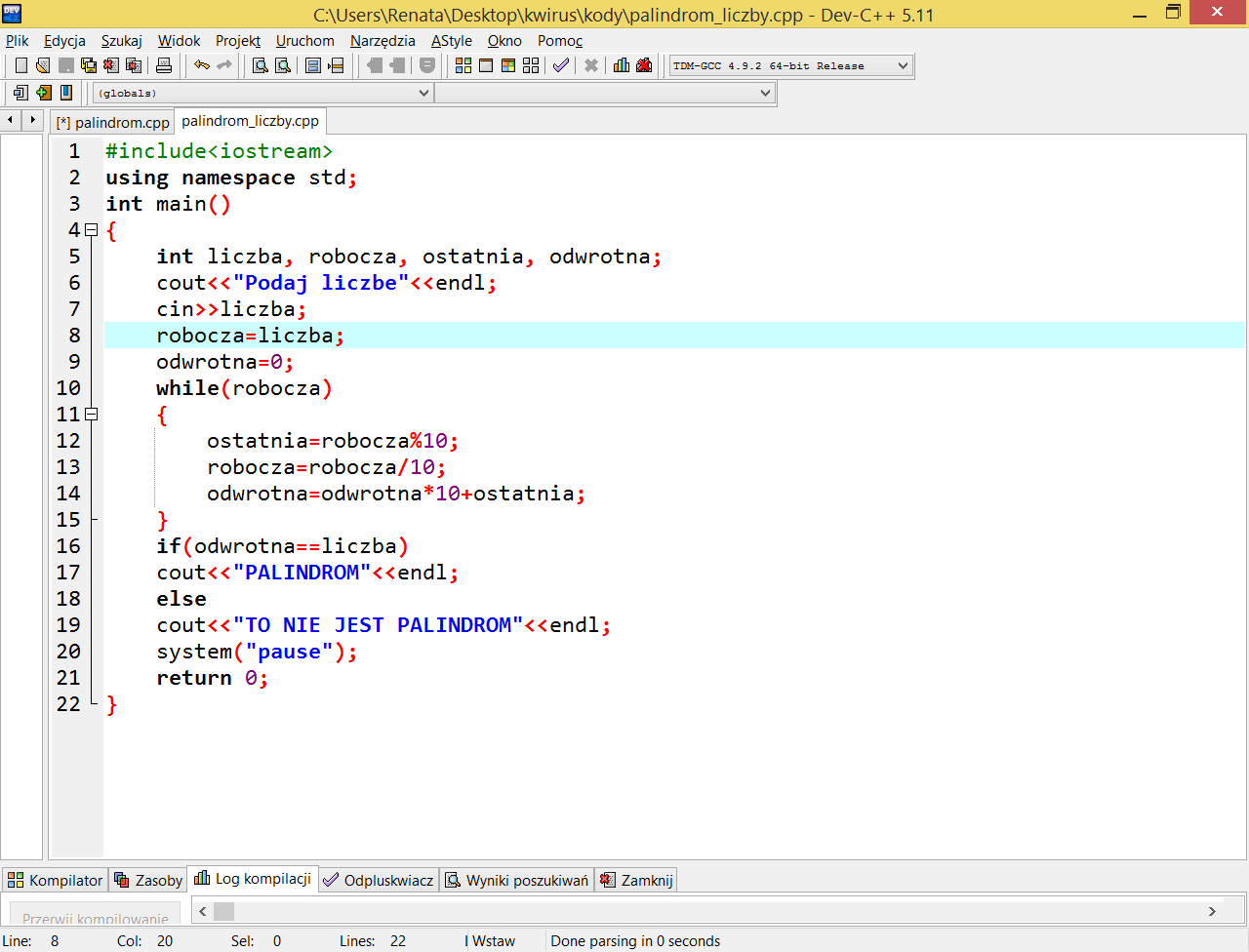
**Palindrom**

Palindrom to ciąg znaków, który czyta się tak samo od lewej do prawej jak i od prawej do lewej strony. Palindromami mogą być słowa, liczby czy dowolne ciągi znaków, np.: kajak, potop, zaraz, Ala, 12321, 565, gk@kg, kos-sok. Można sprawdzić kilkoma sposobami, czy dany ciąg znaków jest palindromem. My zrobimy następującą metodą.



**Na tablicach znakowych typu char pracujemy funkcją strlen(), a na string - length(). Obie zliczają ilość elementów, tylko na innych typach danych.**

**Czy liczba jest palindromem? Przykładowe palindromy: 121, 23432, 32123.**  W kodzie będziemy stosować dzielenie przez podstawę systemu liczenia, czyli przez 10, np. 121/10 aby uzyskać wartości cyfr na pierwszym, drugim i n-tym miejscu oraz reszty dzielenia przez 10. Aby uzyskać liczbę jedności w 121 musimy obliczyć resztę z dzielenia przez 10, czyli 121%10 to jest jeden, czyli mamy ostatnią cyfrę z liczby 121. Reszty z dzielenia dają nam ostatnią cyfrę, przedostatnią itd. Zobaczmy jak to będzie wyglądało w kodzie.



**Zadanie domowe – Przećwiczyć wszystkie kody źródłowe z tego materiału, czyli zapisać w Dev-C++ , skompilować i zobaczyć efekt. Nie przysyłajcie mi plików, to są pliki ćwiczeniowe, tak jak na lekcji.**