Materiał 1:

```
#include <iostream>
#include <cstring>

using namespace std;

bool czy_palindrom(char tab[])
{
	int dl; //zmienna przechowuje długość wczytanego wyrazu
	dl = strlen(tab); //strlen zwraca ilość znaków w tablicy tab
	--dl; //zmniejszenie wartości zmiennej dl, w celu wskoczenia
				// na ostatni indeks tablicy
	for(int i=0;i<=dl/2;i++)
	  if(tab[i]!=tab[dl-i]) //jeśli odpowiednie litery nie będą się zgadzać
		return false;			//funkcja zwróci false

	return true; //gdy wszystkie litery będą się zgadzać
}

int main()
{
	int t; /*zmienna określa ilość zestawów danych*/

	char wyraz[101]; //zmienna przechowuje wczytany wyraz

	cin>>t; //wczytanie liczby testów

	while(t--) //pętla odpowiedzialna za wczytanie zestawów danych
	{
		cin>>wyraz;

		if(czy_palindrom(wyraz)) //lub if(czy_palindrom(wyraz == true)
		  cout<<"tak"<<endl;
		else
		  cout<<"nie"<<endl;

	}

	return 0;
}
```

Objaśnienie.

Rozpatrzmy słowo *kajak*. Popatrzmy jak przechowywane są kolejne litery tego wyrazu:

k - 0

a - 1

j - 2

a - 3

k - 4

Żeby słowo było palindromem pierwszy znak musi być równy ostatniemu, drugi przedostatniemu itd. Czyli:

tab[0] = tab[4]

tab[1] = tab[3]

dalej nie trzeba sprawdzać

Zauważmy, że gdy do zmiennej ***dl*** przypiszemy długość słowa **kajak** (liczbę 5) i zmniejszymy ją o 1, to będzie pasowała ona jako ostatni numer komórki tablicy. Teraz przy każdym przejściu pętli początek zwiększamy o wartość licznika ***i,*** a koniec zmniejszamy wskakując na odpowiednie litery do porównania (tab[i] = tab[dl-i]). Warto zauważyć, że pętla powinna wykonać się nie więcej niż połowę długości wyrazu o parzystej ilości znaków i w naszym przypadku

$$iloscznakow \; div \; 2$$

Materiał 2:

Rozwiązanie rozpocznę od przykładu. Przeanalizujmy wyraz:

$$ALAMALA$$

Ustawię teraz dwa liczniki, jeden na pierwszej komórce w tablicy:

$$i = 0$$

natomiast drugi na ostatniej komórce:

$$j = strlen(tab) - 1$$

Pamiętajmy o tym, że komórki tablicy numerujemy od

$$0$$

, a funkcja *strlen* zwróci nam ilość liter w wyrazie. Żeby prawidłowo ustawić licznik

$$j$$

na ostatnim znaku w tablicy, należy odjąć

$$1$$

od ilości liter ciągu.

Następnie poruszamy się licznikami w stronę środka wyrazu: licznikiem

$$i$$

w prawą stronę natomiast licznikiem

$$j$$

w lewą stronę porównując na bieżąco kolejne znaki. Jeśli jakaś para nie będzie sobie równa, oznacza to, że wyraz nie jest palindromem. Gdy liczniki spotkają się na środku i wszystkie pary będą zgodne to mamy palindrom.

Dla powyższego wyrazu mamy palindrom. Przeanalizujmy ten przypadek.

Wartość i	Wartość j	Litera na i-tej pozycji	Litera na j-tej pozycji	Status
$i = 0$	$j = 6$	A	A	ok
$i = 1$	$j = 5$	L	L	ok
$i = 2$	$j = 4$	A	A	ok

W następnej iteracji liczników spowoduje nieprawdziwość warunku

$$i < j$$

- dalej nie musimy sprawdzać. Przeanalizujmy jeszcze jeden przypadek dla wyrazu

$$BCDSCB$$

Wartość i	Wartość j	Litera na i-tej pozycji	Litera na j-tej pozycji	Status
$i = 0$	$j = 5$	B	B	ok
$i = 1$	$j = 4$	C	C	ok
$i = 2$	$j = 3$	D	S	źle

Tu natomiast widzimy, że w ostatniej iteracji para liter *D* i *S* jest niezgodna, a więc wryaz nie jest palindromem.

Rozwiązanie 2:

```
#include<iostream>
#include<cstring>
using namespace std;

bool czy_palindrom(char tab[])
{
	//ustawiam liczniki "i" i "j" na pierwszy i ostatni znak wyrazu tab
	int i=0, j = strlen(tab)-1; //pamiętajmy, że indeksujemy tablicę od 0

	while(i<j) //pętla wykonuje się do momentu spotkania liczników
	{
		if(tab[i]!=tab[j]) //gdy znaki nie będą się zgadzać, to wyraz nie jest
palindromem
			return false;

		++i; //przejscie do następnej litery idąc w prawą stronę
		--j; //przejscie do poprzedniej litry idąc w lewą stronę
	}

	return true; //wyraz jest palindromem
}

int main()
{
	char tab[100];
	cout<<"Podaj wyraz: ";
	cin>>tab;

	if(czy_palindrom(tab)) //lub if(czy_palindrom(tab)==true) lub
if(czy_palindrom(tab)==1)
		cout<<"Wyraz "<<tab<<" jest palindromem"<<endl;
	else
		cout<<"Wyraz "<<tab<<" nie jest palindromem"<<endl;

	return 0;
}
```