

## Znajdowanie najmniejszego lub największego elementu w ciągu. [Proste algorytmy iteracyjne.]

Algorytm	Specyfikacja	Opis słowny	Podstawowa część kodu programu
Znajdowanie najmniejszego elementu w n-elementowym ciągu liczb [całkowitych].	<b>Dane:</b> Tablica T - n-elementowa tablica liczb całkowitych Type tab = array[1..n] of integer; T: tab; i - zmienna sterująca typu byte; <b>Wynik:</b> min – najmniejsza liczba w ciągu [w tablicy T] – zmienna typu integer	1. Wypełniamy tablicę T wartościami [danymi]. 2. Zmiennej min przypisujemy pierwszą wartość z tablicy. 3. Przechodzimy całą tablicę od drugiej wartości do końca za każdym razem porównując wartość przechowywaną w zmiennej min z wartością przechowywaną w danej komórce tablicy T, jeżeli jest ona mniejsza od min, to przypisujemy wartość tej komórki zmiennej min. 4. Po przejściu tablicy T mamy w zmiennej min najmniejszą wartość z tablicy i wypisujemy ją [ze zmiennej min].	Const n=7; Type tab = array[1..7] of integer; Var T: tab; i, n: byte; min: integer; Begin Randomize; For i:=1 to n do T[i]:=1+Random(31); min:=T[1]; for i:=2 to n do if min>T[i] then min:=T[i]; write('min=',min) End.
Znajdowanie największego elementu w n-elementowym ciągu liczb [rzeczywistych].	<b>Dane:</b> Tablica V - n-elementowa tablica liczb rzeczywistych Type tabl = array[1..n] of real; V: tabl; j - zmienna sterująca typu word; <b>Wynik:</b> max – największa liczba w ciągu [w tablicy V] – zmienna typu real	<p style="color: red;">Zmodyfikuj poniższe zapisy tak, aby dotyczyły algorytmu znajdowania największego elementu w ciągu. [Poniżej wersja już po modyfikacji.]</p> 1. Wypełniamy tablicę V wartościami [danymi]. 2. Zmiennej max przypisujemy pierwszą wartość z tablicy. 3. Przechodzimy całą tablicę od drugiej wartości do końca za każdym razem porównując wartość przechowywaną w zmiennej max z wartością przechowywaną w danej komórce tablicy V, jeżeli jest ona większa od max, to przypisujemy wartość tej komórki zmiennej max. 4. Po przejściu tablicy V mamy w zmiennej max największą wartość z tablicy i wypisujemy ją [ze zmiennej max].	Const n=50; Type tabl = array[1..33000] of real; Var V: tabl; j, n: word; max: real; Begin Randomize; For j:=1 to n do V[j]:=100*Random(); max:=V[1]; for j:=2 to n do if max<V[j] then max:=V[j]; write('max=',max) End.
Zmodyfikuj poniższe zapisy tak, aby dotyczyły algorytmu obliczania sumy elementów ciągu. [Poniżej wersja już po modyfikacji.]			
Obliczanie sumy wszystkich elementów k-elementowego ciągu liczb [całkowitych].	<b>Dane:</b> Tablica T - n-elementowa tablica liczb całkowitych Type tab = array[1..n] of word; T: tab; i - zmienna sterująca typu word; min - zmienna typu word; <b>Wynik:</b> suma – suma wszystkich elementów ciągu [w tablicy T]	1. Wypełniamy tablicę T wartościami [danymi]. 2. Zmiennej suma przypisujemy 0. 3. Przechodzimy całą tablicę od pierwszej wartości do końca za każdym razem dodając wartość przechowywaną w zmiennej suma do wartości przechowywanej w danej komórce tablicy T. 4. Po przejściu tablicy T mamy w zmiennej suma sumę elementów tablicy i wypisujemy ją [ze zmiennej suma].	Const n=50; Type tab = array[1..50] of word; Var T: tab; i, n: word; suma: LongInt; Begin Randomize; For i:=1 to n do T[i]:=Random(101); suma:=0; for i:=1 to n do suma:=suma+T[i]; write('suma=',suma) End.

## Jednoczesne znajdowanie najmniejszego i największego elementu w ciągu – zasada dziel i zwyciężaj.

Algorytm	Specyfikacja	Opis słowny	Podstawowa część kodu programu
Jednoczesne znajdowanie w ciągu elementu najmniejszego i największego - metoda "dziel i zwyciężaj"	<p><b>Dane:</b> Liczba naturalna <math>n</math> i zbiór <math>n</math> liczb dany w postaci ciągu <math>x_1, x_2, \dots, x_n</math>.</p> <p><b>Wynik:</b> Najmniejszy element <math>min</math> i największy element <math>max</math> wśród liczb <math>x_1, x_2, \dots, x_n</math>.</p>	<p><b>Krok 1.</b> Podział zbioru na 2 podzbiory: <math>N</math> - zbiór kandydatów na minimum oraz <math>M</math> - zbiór kandydatów na maksimum. Na początku zbiory są puste. Jeśli <math>n</math> jest liczbą parzystą, to dla <math>i=1, 3, \dots, n-1</math>, a jeśli <math>n</math> jest liczbą nieparzystą, to dla <math>i=1, 3, \dots, n-2</math> wykonaj: jeśli <math>x_i \leq x_{i+1}</math>, to dołącz <math>x_i</math> do <math>N</math>, a <math>x_{i+1}</math> do <math>M</math>, a w przeciwnym wypadku dołącz <math>x_i</math> do <math>M</math>, a <math>x_{i+1}</math> do <math>N</math>. Jeśli <math>n</math> jest liczbą nieparzystą, to dołącz <math>x_n</math> do obu zbiorów <math>M</math> i <math>N</math>.</p> <p><b>Krok 2.</b> Znajdź <math>min</math> w zbiorze <math>N</math> stosując algorytm znajdowania wyrazu minimalnego w ciągu.</p> <p><b>Krok 3.</b> Znajdź <math>max</math> w zbiorze <math>M</math> stosując algorytm znajdowania wyrazu maksymalnego w ciągu.</p>	

## Porządkowanie przez wybór – iteracja algorytmu [znajdowania minimum lub maksimum wciągu].

Algorytm	Specyfikacja	Opis słowny	Podstawowa część kodu programu
<p>Porządkowanie ciągu elementów - sortowanie przez wybieranie [rosnące – od elementu najmniejszego do największego]</p>	<p><b>Dane:</b> Liczba naturalna <math>n</math> [liczba elementów ciągu]; Ciąg liczb <math>a_1, a_2, \dots, a_n</math> [Tablica A]</p> <p><b>Wynik:</b> Ciąg uporządkowany w kolejności rosnącej.</p>	<p><b>Krok 1:</b> Dla <math>i = 1, 2, \dots, n-1</math> wykonaj kroki 2, 3.  <b>Krok 2:</b> Znajdź <math>k</math> takie, że <math>x_k</math> jest najmniejszym elementem w podciągu <math>x_i, \dots, x_n</math>.  <b>Krok 3:</b> Zamień miejscami elementy <math>x_i</math> oraz <math>x_k</math>.</p>	<pre> Const n=80; Type  tablica = array[1..80] of integer; Var    A: tablica;        i, j, n: byte;        min, pomocnik: integer;  Begin     Randomize;     For i:=1 to n do A[i]:=Random(778);     begin         For i:=1 to n-1 do             begin                 min:=A[i];                 for j:=i+1 to n do                     if min&gt;A[j] then                         begin                             min:=A[j];                             poz:=j                         end;                 pomocnik:=A[i];                 A[i]:=min;                 A[poz]:=pomocnik             end;         For i:=1 to n do Write(A[i]:8);         Readln     end. </pre>
<p>Porządkowanie ciągu elementów - sortowanie przez wybieranie [malejące – od elementu największego do najmniejszego]</p>	<p><b>Dane:</b> Liczba naturalna <math>n</math> [liczba elementów ciągu]; Ciąg liczb <math>a_1, a_2, \dots, a_n</math> [Tablica A]</p> <p><b>Wynik:</b> Ciąg uporządkowany w kolejności malejącej.</p>	<p><b>Krok 1:</b> Dla <math>i = 1, 2, \dots, n-1</math> wykonaj kroki 2, 3.  <b>Krok 2:</b> Znajdź <math>k</math> takie, że <math>x_k</math> jest największym elementem w podciągu <math>x_i, \dots, x_n</math>.  <b>Krok 3:</b> Zamień miejscami elementy <math>x_i</math> oraz <math>x_k</math>.</p>	<pre> Const n=80; Type  tablica = array[1..80] of integer; Var    A: tablica;        i, j, n: byte;        max, pomocnik: integer;  Begin     Randomize;     For i:=1 to n do A[i]:=Random(778);     begin         For i:=1 to n-1 do             begin                 max:=A[i];                 for j:=i+1 to n do                     if max&lt;A[j] then                         begin                             max:=A[j];                             poz:=j                         end;                 pomocnik:=A[i];                 A[i]:=max;                 A[poz]:=pomocnik             end;         For i:=1 to n do Write(A[i]:8);         Readln     end. </pre>

			<div>end; For i:=1 to n do Write(A[i]:8); Readln End.</div>
--	--	--	---