## Podsumowanie

- Algorytm sita Eratostenesa pozwala wyznaczyć liczby pierwsze z określonego przedziału.
- Algorytm nie bada podzielności liczb, lecz usuwa (oznacza jako liczby złożone) wielokrotności kolejnych liczb zbioru, większe od tych liczb.
- Najmniejszą wielokrotnością, którą algorytm usuwa, jest kwadrat rozpatrywanej liczby.
- Algorytm kończy działanie, gdy kwadrat liczby, której krotności ma usunąć, jest większy od prawego końca przedziału.
- Liczba operacji wykonywanych przez algorytm sita Eratostenesa jest znacznie mniejsza niż podczas sprawdzania pierwszości każdej liczby niezależnie.

## Zadania

* **1** Napisz program, który wyznaczy liczby pierwsze w przedziale liczbowym podanym przez użytkownika.

* **2** Napisz program wypisujący pary liczb bliźniaczych nie większych od $n$, $2 \leqslant n \leqslant 1\,000\,000$. W rozwiązaniu wykorzystaj algorytm sita Eratostenesa.
Uwaga: Liczby bliźniacze to liczby pierwsze, których różnica wynosi 2. Liczbami bliźniaczymi są np. pary: 3 i 5, 5 i 7, 11 i 13.

** **3** Program utworzony w tym temacie zmodyfikuj tak, aby w tablicy była przechowywana wyłącznie informacja o liczbach nieparzystych (czyli rozmiar tablicy był dwukrotnie mniejszy).

** **4** Program wypisujący liczby półpierwsze nie większe od $n$, $2 \leqslant n \leqslant 1000$. Wypisywane liczby nie muszą być uporządkowane. W rozwiązaniu wykorzystaj algorytm sita Eratostenesa.
Uwaga: Liczba półpierwsza to liczba będąca iloczynem dwóch liczb pierwszych. Liczbami półpierwszymi są np.: 4, 6, 9, 10, 14, 15.

*** **5** Napisz program realizujący algorytm sita Eratostenesa z użyciem szablonu struktury danych **set** z biblioteki STL. Struktura **set** reprezentuje w pamięci komputera zbiór.
Wskazówka: Żeby móc korzystać z typu **set**, należy dodać dyrektywę `#include <set>`.
Do dodania i usuwania elementów zbioru możesz wykorzystać metody **insert** i **erase**:

```
set<int> zbior;
zbior.insert(liczba);
zbior.erase(liczba);
```

Do pobrania kolejnego elementu zbioru możesz wykorzystać instrukcję:

```
liczba=*zbior.upper_bound(liczba);
```

Do wypisania elementów zbioru należy użyć zmiennej typu **iterator**:

```
set<int>::iterator it;
for(it=zbior.begin();it!=zbior.end();it++) cout<<*it<<" ";
```



# 15. Szukamy różnych podciągów

Czasami zachodzi potrzeba znalezienia fragmentu danych spełniającego pewne warunki. Na przykład trener zapisuje wyniki zawodnika osiągane podczas treningów, aby się dowiedzieć, kiedy rezultaty są coraz lepsze, a kiedy następuje spadek formy. Dużą popularnością cieszą się aplikacje, za pomocą których możemy monitorować swoją aktywność (sen, ruch, wysiłek fizyczny itp.), a potem np. sprawdzać, jaki był najdłuższy okres regularnych treningów albo kiedy spaliśmy najkrócej. W tym temacie zajmiemy się poszukiwaniem danych spełniających określone warunki.

### Cele lekcji

- Obliczysz długość najdłuższego spójnego podciągu niemalejącego.
- Wyznaczysz najdłuższy spójny podciąg niemalejący.
- Poznasz i porównasz różne algorytmy znajdowania maksymalnej sumy elementów podciągu spójnego.
- Znajdziesz podciąg spójny o maksymalnej sumie elementów.

**Podciąg** to wybrane elementy ciągu wyjściowego zachowujące kolejność występowania. Na przykład liczby 3, 2, 5, 7 tworzą podciąg ciągu 3, 1, 2, 5, 7, 4. Jeśli elementy podciągu występują w ciągu wyjściowym obok siebie, to taki podciąg nazywamy **podciągiem spójnym**. Na przykład ciąg 1, 2, 5 jest podciągiem spójnym ciągu 3, 1, 2, 5, 7, 4. W tym temacie będziemy poszukiwać tylko podciągów spójnych.

● Podciąg

● Podciąg spójny

## 15.1. Szukamy długości najdłuższego spójnego podciągu niemalejącego

W **podciągu niemalejącym** każdy kolejny element jest większy lub równy poprzedniemu. Na przykład w ciągu 3, 1, 2, 5, 7, 4 możemy wyróżnić następujące spójne podciągi niemalejące:

- 1, 2
- 1, 2, 5
- 1, 2, 5, 7
- 2, 5
- 2, 5, 7
- 5, 7

● Podciąg niemalejący

W powyższym przykładzie najdłuższy spójny podciąg niemalejący tworzą liczby: 1, 2, 5, 7. Jego długość jest równa 4.

Zapiszmy specyfikację problemu znajdowania długości najdłuższego spójnego podciągu niemalejącego.

**Specyfikacja**

**Dane:** A[0..n–1] – tablica *n* liczb całkowitych.

**Wynik:** maks_dl – długość najdłuższego spójnego podciągu niemalejącego w tablicy A.

**Warto wiedzieć**

Długość najdłuższego podciągu niemalejącego będzie równa jeden, gdy ciąg wyjściowy jest malejący.

**Znajdujemy maksimum,** s. 146

Podciągi jednoelementowe potraktujemy jako podciągi niemalejące. Może się więc zdarzyć, że szukana długość będzie równa jeden. Może być też równa długości ciągu wyjściowego.

Algorytm poszukiwania długości najdłuższego spójnego podciągu niemalejącego jest podobny do **znajdowania maksimum w tablicy**. Długość aktualnie badanego fragmentu ciągu będziemy pamiętać w zmiennej akt_dl, a największą dotychczas znalezioną długość podciągu spójnego – w zmiennej maks_dl. Ponieważ każdy element ciągu tworzy podciąg jednoelementowy, początkowe wartości zmiennych akt_dl i maks_dl będą równe 1. Pętla główna przegląda kolejne elementy tablicy, zaczynając od drugiego. Jeśli dany element tablicy jest większy lub równy poprzedniemu, to powiększamy wartość zmiennej akt_dl o 1. Ponadto sprawdzamy, czy długość badanego podciągu jest większa od dotychczas znalezionej największej długości podciągu. Jeśli jest większa, zmieniamy wartość zmiennej maks_dl. Gdy przeglądany element tablicy jest mniejszy od poprzedniego, staje się on początkiem następnego podciągu, a więc wartość zmiennej akt_dl musimy ustawić ponownie na 1. Oto zapis algorytmu w pseudokodzie oraz kod źródłowy funkcji realizującej ten algorytm:

```
maks_dl ← 1
akt_dl ← 1
dla i ← 1, …, n – 1 wykonuj
    jeśli A[i] ≥ A[i–1] to
        akt_dl ← akt_dl + 1
        jeśli akt_dl > maks_dl to maks_dl ← akt_dl
    w przeciwnym przypadku akt_dl ← 1
```

Fragment kodu źródłowego programu poszukującego długości najdłuższego spójnego podciągu niemalejącego – definicja funkcji znajdującej tę długość

```
1.  int DNSPNM(int A[])
2.  // Długość Najdłuższego Spójnego Podciągu NieMalejącego
3.  {
4.      int maks_dl=1, akt_dl=1, i;
5.      for (i=1;i<N;i++)
6.          if (A[i]>=A[i-1])
7.          {
8.              akt_dl++;
9.              if (akt_dl>maks_dl) maks_dl=akt_dl;
10.         }
11.         else akt_dl=1;
12.     return maks_dl;
13. }
```





N oznacza w programie stałą określającą rozmiar tablicy. Do wylosowania i wypisania elementów tablicy można wykorzystać te same funkcje, które stosowaliśmy w **programach wyszukujących element w tablicy**.

Rysunek 15.1 przedstawia przykładowe wywołanie programu.

**Fragment kodu źródłowego programu wyszukującego element w tablicy,** s. 152

```
15 70 31 44 52 70 9 66 38 11
Dlugosc najdluzszego spojnego podciagu niemalejacego: 4
```

Rys. 15.1. Przykład wywołania programu znajdującego długość najdłuższego spójnego podciągu niemalejącego

**Ćwiczenie 1**

Napisz program znajdujący długość najdłuższego spójnego podciągu niemalejącego. Wykorzystaj w nim zdefiniowaną wcześniej funkcję DNSPNM.

## 15.2. Szukamy najdłuższego spójnego podciągu niemalejącego

Żeby znaleźć najdłuższy spójny podciąg niemalejący, wystarczy wyznaczyć indeks pierwszego elementu tego podciągu. Koniec podciągu jest wówczas wyznaczany przez indeks pierwszego elementu mniejszego od poprzedniego lub przez koniec ciągu. Jeśli w danym ciągu jest kilka najdłuższych podciągów, nie jest ważne, który z nich znajdziemy.

Aby wyznaczyć indeks pierwszego elementu podciągu, zmodyfikujemy algorytm wyznaczający długość najdłuższego spójnego podciągu niemalejącego. Będziemy pamiętać nie tylko długość aktualnie badanego podciągu i podciągu dotychczas najdłuższego, lecz także indeksy pierwszych elementów tych podciągów. Zapiszemy je odpowiednio w zmiennych akt_pocz i maks_pocz.

**A to ciekawe**

### Poszukiwanie podciągów w DNA

Podstawowym składnikiem DNA są nukleotydy. W DNA występują cztery rodzaje nukleotydów, a każdy z nich zawiera inną zasadę azotową: adeninę, guaninę, cytozynę lub tyminę. Nić DNA można więc opisać jako ciąg składający się z liter reprezentujących te nukleotydy: A, G, C i T. Dzięki poszukiwaniu powtarzających się podciągów we fragmentach DNA badacze potrafią np. ustalić pokrewieństwo osób lub wskazać sprawcę przestępstwa. Z kolei znalezienie w kodzie określonych genów pozwala określić predyspozycje do zachorowania na niektóre choroby.

Zmodyfikowany zapis algorytmu w pseudokodzie może być następujący:

```
maks_dl ← 1
akt_dl ← 1
maks_pocz ← 0
akt_pocz ← 0
dla i ← 1, …, n – 1 wykonuj
    jeśli A[i] ≥ A[i–1] to
        akt_dl ← akt_dl + 1
        jeśli akt_dl > maks_dl to
            maks_dl ← akt_dl
            maks_pocz ← akt_pocz
    w przeciwnym przypadku
        akt_dl ← 1
        akt_pocz ← i
```

Wartością początkową każdej ze zmiennych `maks_pocz` i `akt_pocz` jest zero (indeks pierwszego elementu tablicy). Początkiem kolejnego podciągu do zbadania jest pierwszy element, który jest mniejszy od elementu poprzedniego.

Zapisując powyższy algorytm w postaci funkcji, warto się zastanowić, co powinno być wartością funkcji: indeks pierwszego elementu znalezionego podciągu czy długość podciągu. Jeśli znamy indeks pierwszego elementu, będziemy mogli wypisać ten podciąg, podczas gdy informacja tylko o jego długości takiej możliwości nie daje. Jeśli wystąpi więcej niż jeden najdłuższy podciąg, funkcja znajdzie pierwszy taki podciąg i zwróci indeks jego pierwszego elementu.

Oto kod źródłowy funkcji realizującej podany algorytm:

**Fragment kodu źródłowego programu wyznaczającego najdłuższy spójny podciąg niemalejący – definicja funkcji NSPNM, której wynikiem jest indeks pierwszego elementu tego podciągu**

```
1.  int NSPNM(int A[])
2.  // Najdłuższy Spójny Podciąg NieMalejący
3.  // wynikiem jest indeks pierwszego elementu podciągu
4.  {
5.      int maks_dl=1, akt_dl=1, maks_pocz=0, akt_pocz=0, i;
6.      for (i=1;i<N;i++)
7.          if (A[i]>=A[i-1])
8.          {
9.              akt_dl++;
10.             if (akt_dl>maks_dl)
11.             {
12.                 maks_dl=akt_dl; maks_pocz=akt_pocz;
13.             }
14.         }
15.         else
16.         {
17.             akt_dl=1; akt_pocz=i;
18.         }
19.     return maks_pocz;
20. }
```





Funkcja `main` programu znajdującego najdłuższy spójny podciąg niemalejący powinna wypisywać elementy tablicy od elementu znajdującego się na pozycji, która jest wartością funkcji `NSPNM`, do momentu, kiedy napotka element nienależący do podciągu lub koniec tablicy. Kod źródłowy funkcji `main` może wyglądać następująco:

**Fragment kodu źródłowego programu wyznaczającego najdłuższy spójny podciąg niemalejący – funkcja `main`**

```
1.  int main()
2.  {
3.      int i;
4.      int A[N];
5.      srand(time(NULL));
6.      Losuj(A);
7.      Wypisz(A);
8.      cout<<"Najdluzszy spojny podciag niemalejacy:";
9.      cout<<endl;
10.     i=NSPNM(A);
11.     do
12.     {
13.         cout<<A[i]<<" ";
14.         i++;
15.     }
16.     while (i<N && A[i]>=A[i-1]);
17.     return 0;
18. }
```

**Dobra rada**

Pamiętaj, że jeśli w pętli powtarzanych jest kilka instrukcji, należy je ująć w nawiasy klamrowe, tworzące blok instrukcji.

Ponieważ zostanie wypisany co najmniej jeden element tablicy, wygodnie będzie użyć pętli **`do while`**, która sprawdza warunek powtarzania po wykonaniu instrukcji, a nie przed wykonaniem (linie 11–16). Zwróć uwagę na warunek pętli w linii 16 – najpierw musimy sprawdzić, czy indeks jest z zakresu tablicy, a potem dopiero odwołać się do elementu tablicy o tym indeksie.

### Ćwiczenie 2

Napisz program wypisujący najdłuższy spójny podciąg niemalejący. Wykorzystaj w nim zdefiniowaną w temacie funkcję `NSPNM`.

Można także zapisać funkcję `NSPNM` tak, aby przekazywała do programu obydwie wyliczone wartości: indeks pierwszego elementu podciągu i długość znalezionego podciągu. Jedna z tych wartości może być wynikiem funkcji, a druga może być **przekazana jako parametr przez referencję** lub obydwie mogą być przekazane jako parametry (wówczas funkcja powinna być typu **`void`**).

Przekazanie parametru przez referencję, s. 161

Zmodyfikowany nagłówek funkcji NSPNM może być następujący:

Fragment kodu źródłowego programu wyznaczającego najdłuższy spójny podciąg niemalejący – zmodyfikowany nagłówek funkcji NSPNM

```
int NSPNM(int A[], int &maks_dl)
// Najdłuższy Spójny Podciąg NieMalejący
// wynikiem jest indeks pierwszego elementu podciągu
// długość podciągu jest zwracana za pośrednictwem
// parametru
```

Zwróć uwagę, że zmienna maks_dl jest teraz parametrem, więc nie może być już deklarowana jako zmienna lokalna. Pozostała treść funkcji pozostaje bez zmian. Znaleziony podciąg można wypisać wówczas w funkcji main za pomocą pętli **for**. Oto kod źródłowy funkcji main:

Fragment kodu źródłowego programu wyznaczającego najdłuższy spójny podciąg niemalejący – zmodyfikowana funkcja main

```
int main()
{
    int i, p, dl;
    int A[N];
    srand(time(NULL));
    Losuj(A);
    Wypisz(A);
    cout<<"Najdluzszy spojny podciag niemalejacy:";
    cout<<endl;
    p=NSPNM(A,dl);
    for (i=p;i<p+dl;i++) cout<<A[i]<<" ";
    return 0;
}
```

**Warto wiedzieć**

Do zwrócenia pary liczb można wykorzystać strukturę przechowującą dwie liczby. O strukturach pisaliśmy w temacie 7.

Żeby znaleźć najdłuższy spójny podciąg niemalejący, program przegląda tablicę jeden raz, więc złożoność czasowa tego algorytmu jest liniowa: O(*n*).

### Ćwiczenie 3

Napisz i przetestuj program wypisujący najdłuższy spójny podciąg niemalejący. Wykorzystaj w nim funkcję NSPNM, w której:

**a.** indeks pierwszego elementu podciągu będzie wartością funkcji, a długość znalezionego podciągu będzie zwracana do programu jako parametr przekazany przez referencję,

**b.** indeks pierwszego elementu podciągu i długość znalezionego podciągu będą zwracane do programu jako parametry przekazywane przez referencję.

**Zapamiętaj**

Podciąg to wybrane elementy danego ciągu zachowujące kolejność występowania. W podciągu spójnym elementy występują obok siebie tak jak w ciągu wyjściowym. Żeby wyznaczyć najdłuższy niemalejący podciąg spójny, wystarczy przejrzeć ciąg raz.





## 15.3. Szukamy maksymalnej sumy podciągu spójnego

Kolejnym problemem, którym się zajmiemy, będzie znalezienie największej sumy wyrazów wśród podciągów spójnych danego ciągu. Gdyby w tablicy występowały tylko liczby dodatnie, rozwiązanie byłoby oczywiste – największą sumę wyrazów ma podciąg taki sam jak ciąg wyjściowy. Problem staje się trudniejszy, gdy w tablicy występują także liczby ujemne. Weźmy pod uwagę na przykład ciąg liczb:
23, –49, 6, 23, 24, –42, 40, 4, –48, 39

Szukana suma wynosi 55. Jest ona sumą wyrazów podciągu złożonego z liczb: 6, 23, 24, –42, 40, 4.

Oto specyfikacja problemu znajdowania maksymalnej sumy podciągu:

**Specyfikacja**

**Dane:** A[0..n–1] – tablica *n* liczb całkowitych, co najmniej jedna liczba w tablicy jest nieujemna.

**Wynik:** maks_suma – maksymalna suma elementów podciągu spójnego w tablicy A.

Rozważymy różne algorytmy znajdowania maksymalnej sumy podciągu spójnego. Założyliśmy, że w tablicy znajduje się co najmniej jedna liczba nieujemna. Jeśliby takiej liczby nie było, rozwiązaniem problemu byłaby największa liczba w ciągu wyjściowym.

### Znajdowanie maksymalnej sumy podciągu spójnego – algorytm 1

Najprostszym rozwiązaniem wydaje się liczenie sum wyrazów we wszystkich podciągach spójnych. Jeśli aktualnie obliczona suma będzie większa od dotychczas znalezionej, zmieniamy wartość maksymalnej sumy. Powinniśmy zliczać sumę wszystkich podciągów rozpoczynających się od pierwszego elementu tablicy, następnie wszystkich podciągów rozpoczynających się od drugiego elementu itd. Wymaga to użycia potrójnie zagnieżdżonej pętli. Aktualnie liczoną sumę podciągu zapamiętamy w zmiennej akt_suma, a największą znalezioną sumę – w zmiennej maks_suma. Algorytm zapisany w pseudokodzie może wyglądać następująco:

```
maks_suma ← 0
dla i ← 0, …, n – 1 wykonuj
    dla j ← i, …, n – 1 wykonuj
        akt_suma ← 0
        // liczymy sumę podciągu od i do j
        dla k ← i, …, j wykonuj
            akt_suma ← akt_suma + A[k]
        jeśli akt_suma > maks_suma to
            maks_suma ← akt_suma
```

Oto kod źródłowy funkcji realizującej algorytm:

Fragment kodu źródłowego programu znajdującego maksymalną sumę podciągu spójnego – funkcja znajdująca tę sumę (algorytm 1)

```
1.  int MSPS(int A[])
2.  // Maksymalna Suma Podciągu Spójnego
3.  {
4.      int maks_suma=0, akt_suma, i, j, k;
5.      for  (i=0;i<N;i++)
6.          for (j=i;j<N;j++)
7.          {
8.              akt_suma=0;
9.              for (k=i;k<=j;k++) akt_suma+=A[k];
10.             if (akt_suma>maks_suma)
11.                 maks_suma=akt_suma;
12.         }
13.     return maks_suma;
14. }
```

W programie realizującym podany algorytm potrzebna jest funkcja losująca tablicę liczb, w której co najmniej jedna liczba będzie nieujemna. Kod źródłowy takiej funkcji może być następujący:

Fragment kodu źródłowego programu znajdującego maksymalną sumę podciągu spójnego – funkcja losująca tablicę *N* liczb całkowitych z przedziału [–50; 49]

```
1.  void Losuj(int A[])
2.  {
3.      for (int i=0;i<N;i++)
4.          A[i]=rand()%100-50;
5.      A[rand()%N]=rand()%50;
6.  }
```

**Dobra rada**

Możesz zmienić przedział losowania liczb na inny. Żeby wylosować liczbę z przedziału [*a*; *b*], skorzystaj z wyrażenia: `rand()%(b-a+1)+a`

Powyższa funkcja losuje liczby całkowite z przedziału [–50; 49]. W linii 5 w losowym miejscu tablicy dodatkowo wstawiana jest losowa liczba nieujemna. Dzięki temu mamy pewność, że w tablicy nie znajdą się same liczby ujemne.

### Ćwiczenie 4

Napisz i przetestuj program znajdujący największą sumę podciągu spójnego z wykorzystaniem powyższych funkcji `MSPS` i `Losuj`.

### Znajdowanie maksymalnej sumy podciągu spójnego – algorytm 2

Zwróć uwagę, że dla podciągów rozpoczynających się od tego samego elementu nie musimy liczyć każdej sumy od początku. Wystarczy po porównaniu aktualnej sumy z największą dotychczas znalezioną dodać do aktualnej sumy kolejny element tablicy. Na przykład dla kolejnych podciągów: 23, –49, 6 oraz 23, –49, 6, 23 wystarczy dodać do policzonej już sumy wyrazów pierwszego podciągu (–20) wartość kolejnego elementu (23). Pozwoli to usprawnić algorytm poprzez usunięcie jednej pętli.





Zmodyfikowany algorytm zapisany w pseudokodzie może wyglądać następująco:

```
maks_suma ← 0
dla i ← 0, …, n – 1 wykonuj
    akt_suma ← 0
    dla j ← i, …, n – 1 wykonuj
        akt_suma ← akt_suma + A[j]
        jeśli akt_suma > maks_suma to
            maks_suma ← akt_suma
```

Oto kod źródłowy funkcji realizującej zapisany w pseudokodzie algorytm:

Fragment kodu źródłowego programu znajdującego maksymalną sumę podciągu spójnego – funkcja znajdująca tę sumę (algorytm 2)

```
1.  int MSPS(int A[])
2.  // Maksymalna Suma Podciągu Spójnego
3.  {
4.      int maks_suma=0, akt_suma, i, j;
5.      for (i=0;i<N;i++)
6.      {
7.          akt_suma=0;
8.          for (j=i;j<N;j++)
9.          {
10.             akt_suma+=A[j];
11.             if (akt_suma>maks_suma)
12.                 maks_suma=akt_suma;
13.         }
14.     }
15.     return maks_suma;
16. }
```

### Ćwiczenie 5

Napisz i przetestuj program znajdujący maksymalną sumę podciągu spójnego z wykorzystaniem powyższej funkcji.

### Znajdowanie maksymalnej sumy podciągu spójnego – algorytm 3

Zastanówmy się, czy trzeba sprawdzać wszystkie podciągi spójne danego ciągu. Jeśli np. pierwszy element będzie ujemny, nie ma sensu rozpatrywać podciągów rozpoczynających się od tego elementu, ponieważ suma każdego podciągu zaczynającego się od drugiego elementu będzie większa. Ogólnie, gdyby aktualna suma miała przyjąć wartość ujemną lub zero, można rozpocząć badanie następnego podciągu od kolejnego elementu ciągu.

Zapis algorytmu w pseudokodzie oraz kod źródłowy funkcji realizującej ten algorytm są następujące.

Łatwiej będzie wypisać znaleziony podciąg, gdy będziemy znali także indeks jego ostatniego elementu. W tym celu można dodać jeszcze jeden parametr (maks_kon) przekazywany przez referencję, określający tę wartość. Nagłówek funkcji PSMS będzie wówczas następujący:

Fragment kodu źródłowego programu znajdującego podciąg spójny o maksymalnej sumie – zmodyfikowany nagłówek funkcji znajdującej ten podciąg

```
1. void PSMS(int A[], int &maks_suma, int &maks_pocz,
2.           int &maks_kon)
3. // Podciąg Spójny o Maksymalnej Sumie
4. // maks_suma – maksymalna suma podciągu spójnego
5. // maks_pocz – indeks pierwszego elementu podciągu
6. // o maksymalnej sumie
7. // maks_kon – indeks ostatniego elementu podciągu
8. // o maksymalnej sumie
```

Aktualizując informację o znalezionym podciągu, należy także określić wartość parametru maks_kon. Końcem jest indeks aktualnie oglądanego elementu (wartość zmiennej sterującej pętli **for**, czyli i).

Fragment kodu źródłowego programu znajdującego podciąg spójny o maksymalnej sumie – instrukcja wyznaczająca indeks początkowy i indeks końcowy oraz sumę podciągu

```
1. if (akt_suma>maks_suma)
2.     {
3.         maks_suma=akt_suma;
4.         maks_pocz=akt_pocz;
5.         maks_kon=i;
6.     }
```

Znaleziony podciąg można wówczas wypisać za pomocą pętli **for** przeglądającej elementy tablicy w zakresie znalezionych wartości maks_pocz i maks_kon.

### Ćwiczenie 9

Napisz i przetestuj program wypisujący podciąg spójny o maksymalnej sumie. Wykorzystaj zmodyfikowaną zgodnie z powyższym nagłówkiem funkcję PSMS.

**Zapamiętaj**

Żeby policzyć maksymalną sumę podciągu spójnego ciągu liczb całkowitych, w którym co najmniej jedna liczba jest nieujemna, nie trzeba liczyć sum wszystkich podciągów. Wystarczy raz przejrzeć elementy ciągu i podczas przeglądania sumować jego wyrazy. Jeśli obliczona suma będzie niedodatnia, sumowanie rozpoczynamy ponownie od następnego elementu.





## Podsumowanie

- Podciąg to wybrane wyrazy ciągu zachowujące kolejność występowania.
- Podciąg spójny to taki podciąg, którego wyrazy w ciągu wyjściowym występują obok siebie.
- Podciąg niemalejący to taki podciąg, w którym każdy kolejny element jest większy lub równy poprzedniemu.
- Żeby znaleźć spójny podciąg niemalejący, wystarczy raz przejrzeć elementy ciągu.
- Algorytmy znajdowania maksymalnej sumy podciągu analizujące wszystkie możliwe sumy podciągów mają złożoność czasową $O(n^3)$ lub $O(n^2)$.
- Istnieje algorytm liniowy $O(n)$, który znajduje maksymalną sumę podciągu spójnego. Nie analizuje on wszystkich podciągów. Kiedy suma aktualnie badanego podciągu nie jest dodatnia, rozpoczyna badanie kolejnego podciągu od następnego elementu.
- Jeśli funkcja ma zwrócić więcej niż jedną wartość typu prostego, warto zastosować przekazanie parametru przez referencję.

## Zadania

* **1** Napisz program wypisujący najdłuższy spójny podciąg malejący ciągu $n$ liczb całkowitych.

* **2** Napisz program, który wypisze najdłuższy spójny podciąg rosnący ciągu $n$ liczb całkowitych, a w przypadku występowania więcej niż jednego najdłuższego podciągu wypisze ostatni znaleziony.

** **3** Napisz program, który wypisze najdłuższy spójny podciąg rosnący ciągu $n$ liczb całkowitych, a w przypadku występowania więcej niż jednego najdłuższego podciągu wypisze wszystkie.

** **4** Napisz program wypisujący najdłuższy podciąg spójny złożony z liczb całkowitych dodatnich ciągu $n$ liczb całkowitych. Wykorzystaj w nim funkcję losującą tablicę tak, aby występowały w niej zarówno liczby dodatnie, jak i ujemne.

** **5** Napisz program znajdujący maksymalną sumę podciągu spójnego ciągu $n$ liczb całkowitych, w którym mogą wystąpić same liczby ujemne. Program napisz tak, żeby przejrzał tablicę co najwyżej dwa razy.

** **6** Napisz program wyznaczający w ciągu $n$ liczb całkowitych liczbę podciągów spójnych o podanej (wczytanej) sumie. Program napisz tak, aby wykonywał jak najmniej operacji.

*** **7** Napisz program wypisujący z ciągu $n$ liczb całkowitych wszystkie podciągi spójne o podanej (wczytanej) sumie. Program napisz tak, aby wykonywał jak najmniej operacji.