3. Porządek ma znaczenie, czyli sortujemy liczby

Porządkowanie jest ważną częścią ludzkiej aktywności. Zwykle wysiłek w nie włożony opłaca się, bo później wiele zadań można wykonać szybciej i łatwiej. Są sytuacje, w których porządek ma szczególne znaczenie. Na przykład jeśli domy są ponumerowane w zrozumiały sposób, w razie pożaru strażacy sprawniej dotrą na miejsce. W świecie oprogramowania porządkowanie odbywa się dzięki algorytmom sortowania. W tym temacie przyjrzymy im się bliżej.

Cele lekcji

- Uświadomisz sobie, jak ważne jest porządkowanie danych.
- Lepiej zrozumiesz algorytmy wyszukiwania (liniowego i binarnego).
- Poznasz algorytmy sortowania: bąbelkowego i przez wstawianie.
- Dowiesz się, jak w języku C++ używać tablic w argumentach funkcji.

3.1. Do czego służy sortowanie?

Sortowanie (porządkowanie) danych

Sortowanie danych (ang. *sorting*), zwane też **porządkowaniem danych**, to ustawianie danych w zadanej kolejności według określonego kryterium.

Zwykle gdy mówimy o sortowaniu liczb, mamy na myśli uporządkowanie ich w kolejności malejącej albo rosnącej. W rzeczywistości najczęściej jest to kolejność nierosnąca lub niemalejąca, ponieważ sortowane wartości mogą się powtarzać. Wyrazy z reguły ustawiamy w kolejności alfabetycznej (od A do Z) albo odwróconej kolejności alfabetycznej (od Z do A). Natomiast daty i godziny porządkuje się zwykle chronologicznie, czyli od najstarszych do najnowszych (ewentualnie w odwróconej chronologii – od najnowszych do najstarszych).

Uporządkowane informacje, zwłaszcza obszerne zbiory danych, o wiele łatwiej czytać, analizować oraz przeszukiwać.

Rysunek 3.1 przedstawia przykładową tabliczkę przystankową. Zauważ, że dane są uporządkowane na dwóch poziomach: kierunki – alfabetycznie, a godziny dla każdego kierunku – chronologicznie.

LGOTA WIELKA

ROZKŁAD ODJAZDÓW

Kierunek	Przez	Godziny odjazdu		
BOGUMIŁOWICE	SULMIERZYCE	**14:50** S		
ŁÓDŹ	BEŁCHATÓW	**6:58** dnw	**12:08** dnw	**17:03** dnw
PAJĘCZNO	SULMIERZYCE	**09:20** Dm		
RADOMSKO		**05:11** 6s	**07:01** S	**07:09** S
		07:11 S	**08:15** S	**08:15** Dm
		12:05 Dm	**14:35** dnw	**18:05** dnw
		23:18 dnw		

Rys. 3.1. Przykład tabliczki przystankowej

Ćwiczenie 1

Z jakimi uporządkowanymi zbiorami danych masz do czynienia na co dzień? Wymień co najmniej pięć konkretnych przykładów.

Szukanie hasła w słowniku lub skorowidzu

W encyklopediach, słownikach i leksykonach obowiązuje porządek alfabetyczny. Gdy korzystasz z tradycyjnego słownika, np. polsko-angielskiego, zwykle nie szukasz na chybił trafił, lecz intuicyjnie stosujesz bardzo rozsądną strategię poszukiwania słowa. Podobnie postępujesz podczas szukania hasła w indeksie (skorowidzu), umieszczonym zwykle na końcu książki (rys. 3.2).

Rys. 3.2. Fragment indeksu podręcznika *Informatyka na czasie 2*

Jeśli chcesz znaleźć słowo „algorytm", to prawdopodobnie szukasz go wśród początkowych haseł, a jeśli „mediana" – mniej więcej w środku. Następnie szukasz wyrazów zawierających dwie pierwsze litery – w naszej sytuacji będą to odpowiednio „al" i „me". Za każdym razem zawężasz obszar poszukiwań, pomijając strony, na których dane hasło na pewno nie wystąpi. Posortowanie haseł przez twórców słownika pozwala więc znacznie skrócić czas przeszukiwania.

Warto wiedzieć

Porządek alfabetyczny nazywa się też leksykograficznym. W tym uporządkowaniu spacje (i inne tego typu znaki, np. dwukropek) są umieszczane przed literami alfabetu. Wynika to z kolejności znaków w kodzie ASCII. To dlatego hasło „kod źródłowy" w słowniku informatycznym będzie wcześniej niż np. hasło „kodowanie".

Warto wiedzieć

Metodę wyszukiwania, którą stosujemy intuicyjnie podczas korzystania ze słownika, nazywa się algorytmem wyszukiwania interpolacyjnego.

A to ciekawe

Algorytmy wyszukiwania

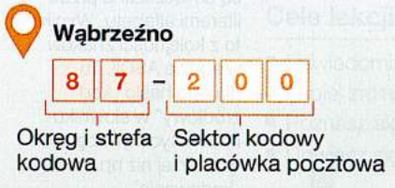
Znalezienie w sieci informacji, których potrzebujemy, byłoby praktycznie niemożliwe bez ich uporządkowania. Systemy rankingowe wyszukiwarek internetowych przeglądają indeksy wyszukiwania, stosując wiele różnych algorytmów, analizujących takie aspekty jak: treść zapytania, przydatność stron, wiarygodność źródła, czas zamieszczenia informacji, lokalizacja użytkownika. Gdyby wykorzystywane algorytmy nie działały bardzo szybko dla dużych zestawów danych, na wyniki dla pojedynczego zapytania trzeba byłoby czekać nawet kilkanaście minut lub dłużej.

Porządek wokół nas

Zwykle nie zwracamy uwagi na to, jak wiele obiektów wokół nas jest uporządkowanych. Ważne jest jednak nie tylko samo uporządkowanie, lecz także jego funkcjonalność, która znacznie ułatwia nam codzienne życie.

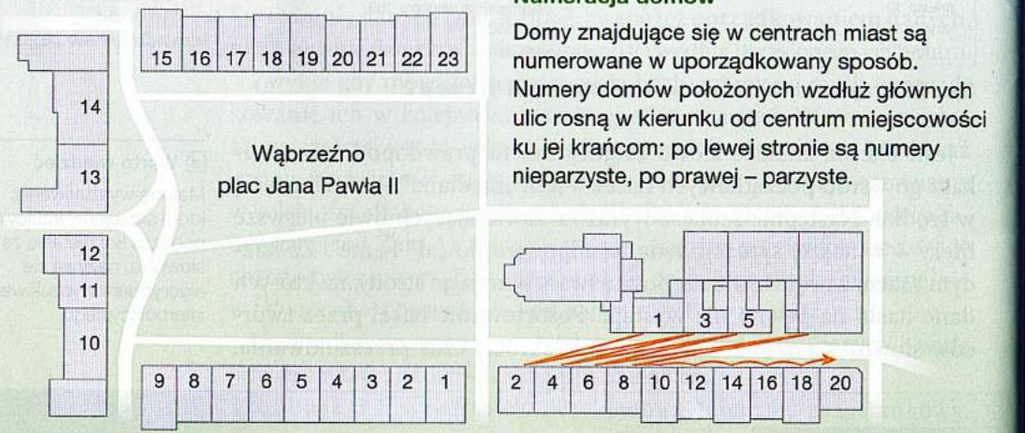
Kody pocztowe

Wprowadzenie kodów pocztowych znacząco usprawniło sortowanie i skróciło czas dostarczania listów. Wcześniej pracownicy poczty musieli sami odszyfrowywać nazwy miejscowości i sprawdzać ich lokalizacje.



Numeracja domów

Domy znajdujące się w centrach miast są numerowane w uporządkowany sposób. Numery domów położonych wzdłuż głównych ulic rosną w kierunku od centrum miejscowości ku jej krańcom: po lewej stronie są numery nieparzyste, po prawej – parzyste.



Porządek w lesie

Lasy w Polsce są podzielone na oddziały (prostokątne fragmenty), które są identyfikowane przez tzw. słupki oddziałowe. Na słupkach znajdują się liczby oznaczające numery oddziałów leśnych. Słupki oddziałowe pomagają określić położenie, wyznaczyć kierunek oraz ułatwiają prowadzenie akcji ratowniczej.

Klasyfikacja zawodników w pięcioboju nowoczesnym

Przyjrzyjmy się bliżej zasadom ustalania kolejności w pięcioboju nowoczesnym. Składa się on z: szermierki, pływania, jazdy konnej oraz konkurencji łączonej – biegu na przełaj i strzelania z pistoletu. Wyniki w poszczególnych konkurencjach są przeliczane na punkty wielobojowe. W tabeli 3.1 przedstawiono wyniki pierwszych dziesięciu zawodniczek w klasyfikacji końcowej Letnich Igrzysk Olimpijskich w Rio de Janeiro w 2016 r. W rywalizacji brało udział 36 zawodniczek.

Warto wiedzieć

Na igrzyskach w Rio de Janeiro w 2016 r. polska pięcioboistka Oktawia Nowacka z wynikiem 1349 punktów wywalczyła brązowy medal.

			Szermierka			Pływanie			Jazda konna		Bieg ze strzelaniem			Suma
			Wygrane	Miejsce	Pkt	Czas	Miejsce	Pkt	Miejsce	Pkt	Czas	Miejsce	Pkt	
1.	Chloe Esposito	AUS	19	13.	215	00:02:12	7.	303	19.	284	00:12:10	2.	570	1372
2.	Élodie Clouvel	FRA	21	7.	227	00:02:09	2.	315	11.	293	00:12:59	17.	521	1356
3.	Oktawia Nowacka	POL	27	1.	264	00:02:17	16.	290	9.	293	00:13:18	25.	502	1349
4.	Annika Schleu	GER	17	16.	202	00:02:19	21.	282	12.	293	00:12:22	3.	559	1336
5.	Kate French	GBR	17	19.	202	00:02:06	15.	292	1.	300	00:12:43	8.	537	1331
6.	Natalya Coyle	IRL	19	12.	215	00:02:17	18.	288	4.	300	00:12:58	16.	522	1325
7.	Alice Sotero	ITA	20	9.	221	00:02:13	8.	303	23.	279	00:13:00	18.	520	1323
8.	Samantha Murray	GBR	14	31.	192	00:02:11	4.	308	22.	279	00:12:59	7.	542	1321
9.	Jelena Potapienko	KAZ	17	20.	202	00:02:12	5.	306	7.	293	00:13:07	22.	513	1314
10.	Tamara Vega	MEX	15	26.	190	00:02:17	17.	290	5.	300	00:12:49	11.	531	1311

Tabela 3.1. Wyniki pięcioboju nowoczesnego kobiet – igrzyska w Rio de Janeiro w 2016 r.

Ćwiczenie 2

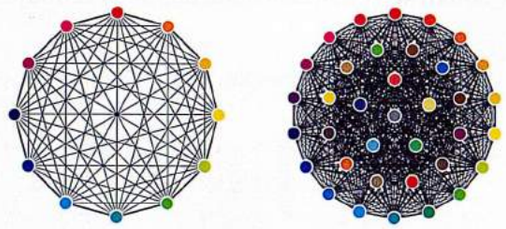
a. Według jakiego kryterium uporządkowano zawodniczki w tabeli 3.1?
b. Czy klasyfikacje dla każdej pojedynczej konkurencji są tworzone na podstawie wyników uporządkowanych malejąco?

Ustalenie kolejności zawodników podczas zawodów w wieloboju jest szczególnie ważne przed rozegraniem ostatniej konkurencji (strzelanie i bieg). Różnice punktowe w klasyfikacji przeliczane są bowiem na sekundy. Lider startuje jako pierwszy, a kolejni z opóźnieniem wyliczonym na podstawie dotychczasowych wyników.

Warto wiedzieć

W języku angielskim system rywalizacji „każdy z każdym" nazywa się *round-robin tournament* (dosł. turniej karuzelowy). Z informatycznego punktu widzenia jest to algorytm sortowania porównawczo-zliczeniowego. Każdy element zbioru danych liczbowych jest porównywany ze wszystkimi pozostałymi, co daje informację, od ilu elementów jest on większy.

Systemy wyłaniania zwycięzców wcześniejszych konkurencji muszą być sprawiedliwe. Na przykład w szermierce stosuje się system „każdy z każdym", zwany systemem kołowym. Przy dużej liczbie rywalizujących jest on jednak bardzo czasochłonny. Dla 36 zawodniczek (każda walczy 35 razy) należy przeprowadzić aż $\frac{1}{2} \cdot 36 \cdot 35 = 630$ pojedynków. Dla porównania – przy 12 zawodniczkach odbyłoby się 66 walk. Różnicę tę pokazuje rysunek 3.3.



Rys. 3.3. Ilustracja liczby pojedynków w rywalizacji „każdy z każdym" dla 12 i 36 zawodniczek

Sortowanie danych w komputerze

Sortowanie zbiorów danych jest jedną z operacji najczęściej wykonywanych przez komputery podczas przetwarzania informacji. Oto kilka przykładów zastosowań sortowania, od prostszych do mniej oczywistych:

- wyświetlanie listy plików w katalogu (folderze) na dysku – przed wyświetleniem zawartości katalogu system operacyjny sortuje pliki zgodnie z określonym kryterium,
- przygotowanie do wydruku danych z arkusza kalkulacyjnego lub bazy danych – często stosujemy sortowanie, aby ułatwić czytanie danych zawartych w tabelach (nadać im odpowiednie znaczenie),
- wyszukiwarki internetowe – zasoby zgromadzone na serwerach wyszukiwarek internetowych muszą być zindeksowane, aby wyszukiwarka mogła zwrócić wynik możliwie szybko,
- grafika komputerowa – obiekty w grze komputerowej lub np. w symulatorze lotu są ułożone na warstwach, więc aby program mógł wyświetlić obiekty na ekranie w dobrej kolejności, może wymagać ich wcześniejszego posortowania (względem odległości od obserwatora).

Warto wiedzieć

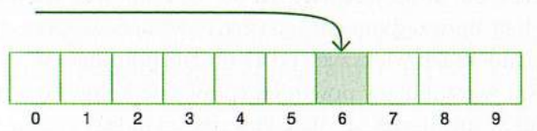
Przykłady problemów, do których rozwiązania potrzebne jest sortowanie danych, to:
- usuwanie duplikatów,
- sprawdzanie, czy wszystkie elementy zbioru danych są różne,
- znajdowanie dwóch wartości o najmniejszej różnicy,
- wyznaczanie mediany, dominanty i innych statystyk,
- odtwarzanie pierwotnego porządku elementów zbioru danych.

3.2. Sortowanie a wyszukiwanie

Uporządkowanie danych ma zasadnicze znaczenie podczas przeszukiwania ich zbioru. Prześledźmy to na przykładzie wyszukiwania w pamięci komputera.

Przyjmijmy, że po uruchomieniu jakiegoś programu w pamięci komputera mamy zapisaną tablicę jako uporządkowany ciąg komórek pamięci RAM, zaadresowanych kolejnymi liczbami całkowitymi.

Jeśli procesor zna położenie danych w pamięci, to odwołuje się do nich bezpośrednio, poprzez adres komórki, w której się znajdują (rys. 3.4).



Rys. 3.4. Dostęp bezpośredni

Aby odczytać dane zapisane w siódmej komórce fragmentu pamięci, procesor nie odczytuje wszystkich danych w kolejności, w jakiej zostały zapisane, lecz pobiera dane z właściwej komórki (rys. 3.4).

Współczesne algorytmy komputerowe często wymagają, aby czas dostępu do danych z pamięci operacyjnej był jak najkrótszy. Wtedy kluczowe znaczenie – oprócz aspektu technicznego (budowy pamięci) – ma właśnie uporządkowanie danych. Jego brak znacznie spowalnia wyszukiwanie informacji, nawet jeśli korzysta się z szybkich procesorów.

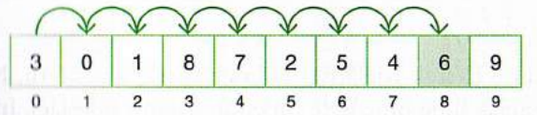
Warto wiedzieć

Czas dostępu do danych zależy nie tylko od ich uporządkowania, lecz także od rodzaju pamięci. Różny czas osiągają rejestry procesora, pamięć podręczna, pamięć RAM czy pamięć zewnętrzna (np. flash, dysk twardy).

Wyszukiwanie sekwencyjne

Kiedy szukamy informacji w zbiorach danych nieuporządkowanych, nie mamy wielkiego wyboru: albo zastosujemy metodę „na chybił trafił", albo algorytm **wyszukiwania sekwencyjnego** (ang. *sequential search*), tj. przeglądanie zbioru danych w kolejności, w jakiej są zapisane (np. według adresów w pamięci). Rysunek 3.5 pokazuje sekwencyjne wyszukiwanie liczby 6 w zbiorze nieuporządkowanym.

Wyszukiwanie sekwencyjne



Rys. 3.5. Wyszukiwanie sekwencyjne

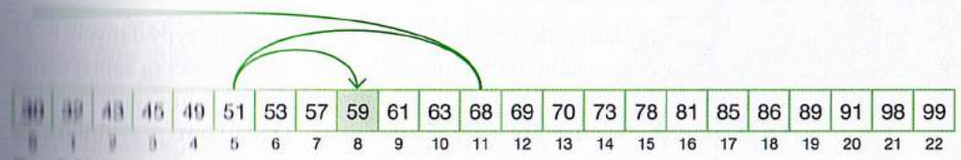
Warto wiedzieć

Metodę „na chybił trafił" nazywa się czasami wyszukiwaniem głupim (ang. *bogo sort*).

Wyszukiwanie binarne

Wyszukiwanie informacji można znacznie przyspieszyć, gdy zbiór danych jest uporządkowany. Stosujemy wówczas strategię połowienia zbioru, czyli zmniejszania liczby jego elementów o połowę w każdej kolejnej próbie. Nazywamy ją **wyszukiwaniem binarnym (połówkowym)**. Algorytm ten, zastosowany do poszukiwania liczby 59 w zbiorze składającym się z 23 elementów, przedstawiono na rysunku 3.6.

Wyszukiwanie binarne (wyszukiwanie połówkowe)



Rys. 3.6. Wyszukiwanie binarne

Warto wiedzieć

Porównanie dwóch liczb jest jedną z podstawowych operacji wykonywanych przez procesor komputerowy. Każdy język programowania musi obsługiwać tę operację.

Wyszukiwanie liczby 59 zaczęło się od sprawdzenia środkowej komórki obszaru pamięci. Jest to komórka o indeksie 11, przechowująca liczbę 68, która jest większa od 59. Ponieważ wiemy, że dane w pamięci są uporządkowane, mamy pewność, że przeszukiwanie komórek o indeksach większych od 11 można pominąć. W drugiej próbie program wyszukujący powinien sprawdzić komórkę o indeksie 5. Znajduje się w niej liczba 51, dlatego wartości należy szukać w komórkach o indeksach od 6 do 10. W kolejnym kroku sprawdzamy komórkę o adresie 8 – znajdziemy w niej poszukiwaną liczbę.

Ćwiczenie 3

Ile sprawdzeń (porównań dwóch liczb) wystarczy, aby rozstrzygnąć, czy w ciągu komórek z rysunku 3.6 znajduje się liczba 80? Zapisz indeksy oraz wartości komórek, które kolejno będą sprawdzane.

3.3. Algorytmy sortowania

Warto wiedzieć

Algorytmy sortowania można podzielić na dwie grupy: te, które działają na zasadzie porównywania elementów zbioru parami (np. sortowanie przez wybór), oraz nieposługujące się tą operacją (np. sortowanie przez zliczanie).

Spójrzmy na porządkowanie jako na problem algorytmiczny i programistyczny. W ostatnich kilkudziesięciu latach odkryto i opisano co najmniej kilkanaście użytecznych algorytmów sortowania danych. Okazuje się, że nie ma algorytmu idealnego o ogólnym przeznaczeniu, najszybszego w każdej sytuacji. Najbardziej uniwersalne są metody hybrydowe, łączące zalety dwóch lub trzech algorytmów sortowania, stosowanych na różnych etapach procesu porządkowania.

Ćwiczenie 4

Przygotuj 8 kartek wielkości połowy kartki z zeszytu. Na każdej z nich zapisz imię oraz datę imienin (dzień i miesiąc). Imiona nie mogą się powtarzać. Następnie odwróć kartki i ułóż je w losowej kolejności.

a. Zaproponuj metodę sortowania informacji na kartkach (alfabetycznie lub chronologicznie). Przyjmij, że kartki trzeba porównywać parami i nie wolno odkryć więcej niż dwóch naraz.

b. W grupie 2- lub 3-osobowej przedyskutujcie zastosowane wcześniej metody porządkowania kartek. Porównajcie czas sortowania 8 i 16 kartek wybraną przez was metodą. Ile razy dłużej trwa porządkowanie 16 kartek?

Programy komputerowe mogą porządkować różne typy danych: liczby, zbiory liter i słów, daty i inne. Wszystkie te wartości są zapisywane cyfrowo w pamięci komputera. Dla uproszczenia ograniczymy się więc do porządkowania tablic liczb całkowitych. Dodatkowo przyjmijmy, że będziemy sortować w porządku niemalejącym.

Oto specyfikacja naszego problemu sortowania:

Specyfikacja

Dane: liczba naturalna `N` i `N`-elementowa tablica `A` liczb całkowitych.

Wynik: tablica `A` po uporządkowaniu: $A[0] \leqslant A[1] \leqslant \ldots \leqslant A[N-1]$.

3.4. Sortowanie bąbelkowe

Jednym z algorytmów porządkowania jest **sortowanie bąbelkowe** (ang. *bubble sort*). W tym algorytmie na każdym etapie porównuje się parę elementów sąsiednich, zaczynając od pierwszej pary w tablicy. Jeżeli podczas porównywania elementy są w niewłaściwej kolejności, to zamienia się je miejscami.

Sortowanie bąbelkowe

Operacja prostej zamiany

Sortowanie bąbelkowe nazywa się też czasami sortowaniem przez **prostą zamianę**, rozumianą jako zamianę miejscami sąsiednich elementów w tablicy.

Operacja prostej zamiany

Podobną operację stosuje się w oprogramowaniu niektórych telewizorów. Pozwala ono za pomocą pilota uszeregować listę kanałów. Są trzy możliwości: zamiana kanałów miejscami (•), przejście w dół (▽) i przejście w górę listy (△). Rysunek 3.7 przedstawia zastosowanie operacji prostej zamiany podczas ustawiania listy kanałów.

TV 3	•	TV 0	▽	TV 0		TV 0		TV 0		TV 0
TV 0		TV 3	•	TV 1	▽	TV 1		TV 1		TV 1
TV 1		TV 1		TV 3	▽	TV 3		TV 3	•	TV 2
TV 6		TV 6		TV 6	▽	TV 6	•	TV 2	△	TV 3
TV 7		TV 7		TV 7	•	TV 2	△	TV 6		TV 6
TV 2		TV 2		TV 2		TV 7		TV 7		TV 7
TV 5		TV 5		TV 5		TV 5		TV 5		TV 5
TV 4		TV 4		TV 4		TV 4		TV 4		TV 4

Rys. 3.7. Pierwsze kroki porządkowania kanałów telewizyjnych przez prostą zamianę

A to ciekawe

Sortowanie w tańcu

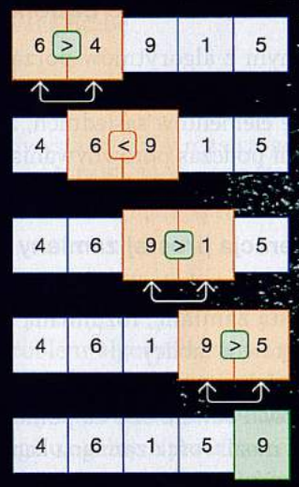
Na kanale AlgoRythmics w serwisie YouTube można znaleźć filmy objaśniające działanie różnych algorytmów sortowania. Porządkowanie liczb jest zaprezentowane w tańcu przez członków węgierskiego zespołu folkowego Marosi Művészegyüttes. Można tam zobaczyć również m.in. wyszukiwanie binarne.

Algorytm sortowania bąbelkowego

Porządkowanie elementów tą metodą pokażemy na przykładzie pięcioelementowej tablicy, w której na kolejnych pozycjach znajdują się liczby: 6, 4, 9, 1, 5. Elementy uporządkujemy niemalejąco. Pierwszy etap ilustruje, jak umieścić właściwy element na ostatniej pozycji. W kolejnych etapach postępujemy analogicznie.

Etap 1

- Porównujemy pierwszy element z drugim. 6 > 4, więc zamieniamy liczby miejscami.
- Porównujemy drugi element z trzecim. 6 < 9, więc nie zamieniamy liczb miejscami.
- Porównujemy trzeci element z czwartym. 9 > 1, więc zamieniamy liczby miejscami.
- Porównujemy czwarty element z piątym. 9 > 5, więc zamieniamy liczby miejscami.
- W wyniku czterech porównań największy element tablicy (liczba 9) znalazł się na ostatniej pozycji.



W drugim etapie umieścimy właściwy element na przedostatniej pozycji. Przy porównaniach pomijamy ostatni element tablicy, ponieważ jest już na właściwym miejscu. W trzecim etapie przy porównaniach pomijamy dwa ostatnie elementy tablicy, w czwartym – trzy ostatnie itd. Poniżej przedstawione są wyniki otrzymane w kolejnych etapach.

Etap 2 Porównujemy pary liczb: 4 i 6, 6 i 1 oraz 6 i 5. Liczba 6 znalazła się na właściwej, przedostatniej pozycji.

Etap 3 Porównujemy pary liczb 4 i 1 oraz 4 i 5. W wyniku tych operacji liczba 5 jest na właściwej pozycji.

Etap 4 Porównujemy parę liczb 1 i 4. Znalezienie drugiego elementu ciągu wskazuje jednocześnie najmniejszy element ciągu.

Wynikiem działania algorytmu jest tablica:

Ćwiczenie 5

Dana jest tablica A = [3, 0, 1, 7, 5]. Sprawdź, ile prostych zamian potrzeba, by uporządkować ją algorytmem sortowania bąbelkowego:

a. rosnąco,
b. malejąco.

Algorytm porządkowania metodą bąbelkową można zapisać jako **pętle zagnieżdżone**. To oznacza, że na jeden etap algorytmu (cykl pętli zewnętrznej) może przypadać wiele cykli pętli wewnętrznej.

Pętle zagnieżdżone, s. 26

Każdy etap algorytmu kończy się ustawieniem na docelowym miejscu elementu, który w analizowanym fragmencie był tym o największej wartości. Liczba porównań w kolejnym etapie zmniejsza się więc o 1.

Oto zapis algorytmu w postaci listy kroków:

1. Wczytaj tablicę liczb A.
2. Dla wartości i równych kolejno 1, 2, …, N-1 powtarzaj kroki 3 oraz 4.
3. Dla wartości j równych kolejno 0, 1, …, N-i-1 powtarzaj krok 4.
4. Jeśli A[j]>A[j+1], to zamień te elementy miejscami.

Liczba powtórzeń zewnętrznej pętli wynosi N–1. Liczba powtórzeń wewnętrznej pętli (tzw. pętli sortującej) jest zależna od wartości zmiennej i, sterującej zewnętrzną pętlą.

Kod źródłowy funkcji o nazwie Babelkowe w programie *Sortowanie bąbelkowe* może wyglądać następująco:

```
1. void Babelkowe (int A[])
2. {
3.     for (int i=1; i<N; i++)
4.         for (int j=0; j<N-i; j++)
5.             if (A[j]>A[j+1])
6.                 swap(A[j],A[j+1]);
7. }
```

Kod źródłowy funkcji Babelkowe w programie *Sortowanie bąbelkowe*

Zauważ, że funkcja Babelkowe jako parametr przyjmuje tablicę A. W ten sposób przekazuje się do funkcji informację o adresie tablicy, czyli położeniu pierwszej komórki tablicy w pamięci. Wszystkie operacje są wykonywane na tej tablicy, a zmiany są w niej zapamiętane nawet po zakończeniu działania funkcji.

W liniach 3 oraz 4 znajduje się odwołanie do wartości N, określającej rozmiar tablicy do posortowania. Wartość ta jest stałą globalną, zdefiniowaną przed definicją funkcji, która z niej korzysta. Dla tablicy o 10 elementach odpowiedni wiersz kodu będzie miał postać: const int N = 10;.

Dobra rada

Aby uporządkować elementy nierosnąco, zmień w instrukcji warunkowej znak > na <.

Warto wiedzieć

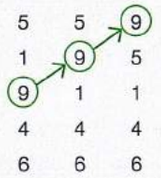
W nagłówku definiowanej funkcji, np. void Funkcja (int A[]), nazwa tablicy nie musi być identyczna z nazwą tablicy, dla której wywołamy funkcję w programie głównym. Funkcję można wywołać dla np. tablicy B i zapisać Funkcja(B).

Dobra rada

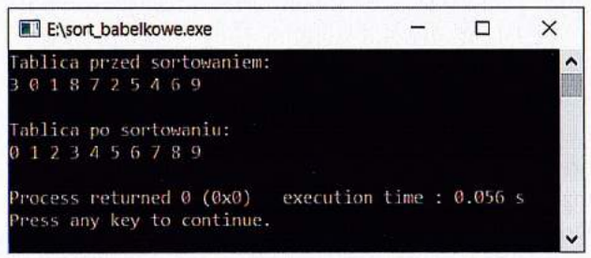
Zamiast wprowadzać dane do tablicy bezpośrednio w kodzie źródłowym, możesz napisać funkcję, która pozwoli wczytać liczby podane przez użytkownika.

Warto wiedzieć

Nazwa „sortowanie bąbelkowe" powstała przez analogię do zjawiska wypływania na powierzchnię wody bąbelków powietrza: ten o największej objętości wypływa jako pierwszy.



Rysunek 3.8 przedstawia przykładowe wywołanie programu *Sortowanie bąbelkowe*.



Rys. 3.8. Przykładowe wywołanie programu *sort_babelkowe.exe*

Ćwiczenie 6

Napisz kod źródłowy programu o nazwie *Sortowanie bąbelkowe*, w którym wykorzystasz funkcję Babelkowe, a dane do sortowania wprowadzisz bezpośrednio w kodzie źródłowym. Skompiluj kod i sprawdź działanie programu.

Zapamiętaj

Kluczową operacją w wielu algorytmach sortowania danych jest porównywanie danych parami, by ustalić ich porządek.

3.5. Sortowanie przez wstawianie

Sortowanie przez wstawianie

Omówimy teraz kolejny algorytm porządkowania danych – **sortowanie przez wstawianie** (ang. *insertion sort*). Metoda ta polega na przeglądaniu kolejnych elementów tablicy od lewej i przenoszeniu ich, w razie potrzeby, w odpowiednie miejsce już uporządkowanego fragmentu tablicy po lewej stronie. Działanie algorytmu przedstawiono na s. 57.

Sortowanie przez wstawianie przypomina układanie kart przez grających w gry karciane: karta, którą podnosimy ze stołu, jest wstawiana we właściwe miejsce do kart już uporządkowanych (rys. 3.9).

Warto wiedzieć

Doświadczeni gracze w pokera lub brydża nie sortują swoich kart. W ten sposób bowiem dostarczyliby przeciwnikom informacji o tym, jakie karty mają w ręku.

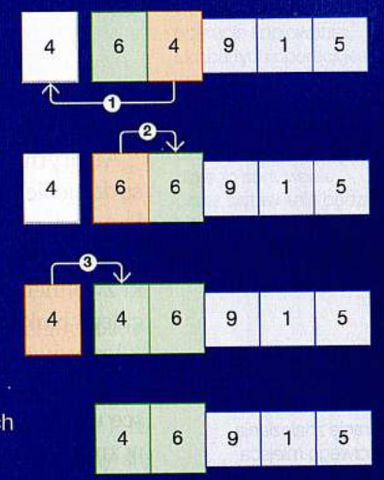
Rys. 3.9. Układanie kart metodą przez wstawianie

Algorytm sortowania przez wstawianie

Porządkowanie elementów tą metodą pokażemy na przykładzie pięcioelementowej tablicy, w której na kolejnych pozycjach znajdują się liczby: 6, 4, 9, 1, 5. Elementy uporządkujemy niemalejąco. Pierwszy etap ilustruje, jak uporządkować niemalejąco dwa elementy tablicy. Liczby przy strzałkach oznaczają kolejność wykonywania operacji.

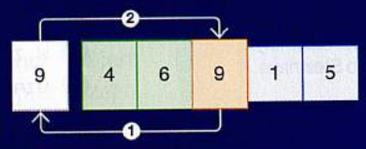
Etap 1

- Przyjmujemy, że pierwszy element tablicy (liczba 6) jest na właściwym miejscu. Drugi element, liczbę 4, zapamiętujemy w zmiennej pomocniczej.
- Liczba 4 powinna się znaleźć przed liczbą 6, dlatego aby zrobić dla niej miejsce, przesuwamy liczbę 6 o jedną pozycję w prawo.
- Liczbę 4 ze zmiennej pomocniczej zapisujemy jako pierwszy element tablicy.
- Wynikiem jest uporządkowanie niemalejąco dwóch pierwszych elementów wyjściowej tablicy.

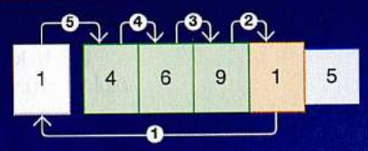


Postępując analogicznie, w drugim etapie uporządkujemy niemalejąco trzy elementy tablicy, w następnym – cztery itd. Poniżej znajduje się skrócony opis kolejnych etapów.

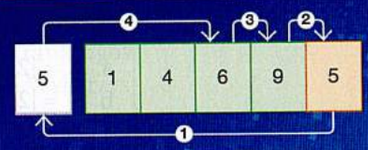
Etap 2 Zapamiętujemy liczbę 9 w zmiennej pomocniczej. Liczba 9 powinna się znaleźć za liczbami 4 i 6, dlatego nie trzeba zmieniać położenia tych liczb, a liczba 9 wraca na swoją pozycję.



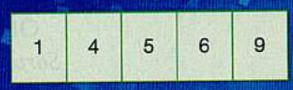
Etap 3 Zapamiętujemy liczbę 1 w zmiennej pomocniczej. Przesuwamy w prawo o jedną pozycję kolejno liczby: 9, 6 i 4. Zapisujemy liczbę 1 na pierwszym miejscu w tablicy.



Etap 4 Zapamiętujemy liczbę 5 w zmiennej pomocniczej. Przesuwamy liczbę 9, a następnie liczbę 6 o jedną pozycję w prawo. Wstawiamy liczbę 5 ze zmiennej pomocniczej w zwolnione miejsce.



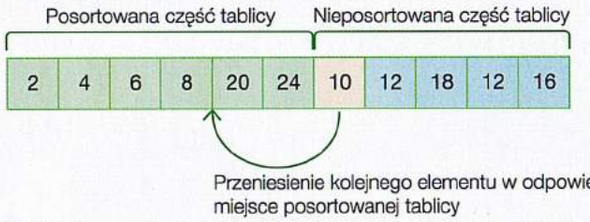
Wynikiem działania algorytmu jest tablica:



Warto wiedzieć

Sortowanie bąbelkowe i sortowanie przez wstawianie przeprowadzają operacje w miejscu, co oznacza, że poza pamięcią dla sortowanej tablicy potrzebują bardzo mało dodatkowej pamięci.

Algorytm dzieli tablicę danych na część posortowaną i nieposortowaną (rys. 3.10). Na każdym etapie algorytmu kolejny element z nieposortowanej części tablicy trafia na właściwą pozycję w obszarze posortowanym.



Rys. 3.10. Umieszczenie liczby 10 w częściowo uporządkowanej tablicy

Algorytmy sortowania bąbelkowego i sortowania przez wstawianie są do siebie podobne: pierwszy stopniowo przesuwa elementy zbioru ku końcowi, podczas gdy drugi przenosi elementy zbioru ku początkowi.

Wstawienie elementu tablicy na właściwą pozycję odbywa się dzięki zwolnieniu miejsca w tablicy poprzez przesunięcie, krok po kroku, całego bloku elementów w prawo o jedną pozycję.

Przesuwanie elementów jest realizowane przez serię operacji kopiowania. Element, który będzie docelowo wstawiony we właściwe miejsce uporządkowanej części tablicy, jest tymczasowo przechowywany w zmiennej pomocniczej. Porządkowanie rozpoczynamy od drugiego elementu (o indeksie 1).

Warto wiedzieć

Operację znalezienia właściwego miejsca dla danego elementu w ciągu uporządkowanym można realizować przez przeszukiwanie liniowe lub binarne. Bardziej efektywny jest drugi sposób. Analizował go już w latach 50. XX w. polski matematyk Hugo Steinhaus.

Oto zapis algorytmu w postaci listy kroków:

1. Wczytaj tablicę liczb A.
2. Dla wartości i równych kolejno 1, 2, …, N-1 powtarzaj kroki 3–8.
   3. W zmiennej pom zapamiętaj wartość zapisaną w A[i].
   4. Oznacz bieżącą wartość i-1 literą j.
   5. Dopóki j≥0 i A[j]>pom, wykonuj kroki 6–7.
      6. Skopiuj A[j] na miejsce A[j+1].
      7. Zmniejsz wartość j o 1.
   8. W komórce A[j+1] zapisz wartość pom.

Ćwiczenie 7

Prześledź działanie powyższej listy kroków dla następujących tablic.

a. A = [2, 3, 8, 5]
b. B = [2, 8, 3, 5]
c. C = [8, 2, 3, 5]

Oto kod źródłowy funkcji o nazwie `Wstawianie` w programie *Sortowanie przez wstawianie*.


```
1.  void Wstawianie(int A[])
2.  {
3.      int j, pom;
4.
5.      for (int i=1; i<N; i++)
6.      {
7.          pom = A[i];
8.          j = i - 1;
9.          while (j>=0 && A[j]>pom)
10.         {
11.             A[j+1] = A[j];
12.             j--;
13.         }
14.         A[j+1] = pom;
15.     }
16. }
```

Kod źródłowy funkcji `Wstawianie` w programie *Sortowanie przez wstawianie*

Warto wiedzieć

Podany algorytm sortowania przez wstawianie wykonuje operacje kopiowania. Liczba tych operacji jest w przybliżeniu równa liczbie porównań. Kopiowanie nie jest jednak tak czasochłonne jak zamiana, dlatego sortowanie przez wstawianie będzie kilkukrotnie szybsze od sortowania bąbelkowego.

Pętla zewnętrzna (linie 5–15) przegląda elementy z części nieposortowanej. W kolejnych iteracjach w zmiennej pomocniczej pom zapamiętujemy kolejne elementy z tej części (wiersz 7). Jeżeli element z lewej strony aktualnie analizowanego elementu jest większy (A[j]>pom), to przesuwamy elementy posortowanej części ciągu, które powinny się znaleźć za danym elementem, a jego wartość zapamiętaną w zmiennej pom wstawiamy na właściwe miejsce (linie 9–13).

Warunkiem w wierszu 9 jest **koniunkcja**, czyli złożone wyrażenie logiczne. Program po skompilowaniu i uruchomieniu będzie działał poprawnie, gdyż w działaniu kompilatora C++ stosuje się tzw. **leniwe wartościowanie** (ang. *lazy evaluation*). W naszym przypadku drugie porównanie się nie wykona, jeśli pierwszy warunek nie jest spełniony, tzn. gdy indeks j będzie mniejszy niż indeks pierwszego elementu tablicy.

Logika matematyczna i operatory logiczne, s. 242–244

Leniwe wartościowanie

Ćwiczenie 8

Zapisz kod źródłowy funkcji `Wstawianie` i napisz program, w którym tę funkcję zastosujesz. Sprawdź działanie programu.

Algorytm sortowania przez wstawianie jest szczególnie efektywny przy sortowaniu zbiorów prawie uporządkowanych, np. rankingów aktualizowanych na bieżąco podczas zawodów sportowych. Wykorzystuje się go również w rozwiązaniach hybrydowych, m.in. w ostatniej fazie sortowania szybkiego (ang. *quicksort*), gdy do posortowania jest wiele podzbiorów o niewielkiej liczbie elementów.

Podsumowanie

- Sortowanie danych to ustawianie ich w zadanej kolejności według określonego kryterium.
- Uporządkowanie informacji ma ogromne znaczenie w wielu aspektach życia – zarówno w technologii komputerowej, jak i dziedzinach zupełnie z nią niezwiązanych.
- Sortowanie bąbelkowe polega na cyklicznym porównywaniu par elementów sąsiednich (zaczynając od pierwszej pary w tablicy) i zamianie miejscami tych elementów, które nie są w odpowiedniej kolejności.
- Sortowanie przez wstawianie polega na przenoszeniu jeden po drugim kolejnych elementów z nieuporządkowanego fragmentu tablicy w odpowiednie miejsce już uporządkowanego fragmentu tablicy.
- Aby funkcji zdefiniowanej w C++ przez użytkownika umożliwić dostęp do tablicy zadeklarowanej w części głównej programu (np. `int tabA[]`), w nagłówku funkcji jako parametr trzeba podać taki sam typ i dowolną nazwę tablicy, np. `Funkcja (int A[])`.
- Funkcję wywołuje się, podając jako parametr aktualny wyłącznie nazwę tablicy, np. `Funkcja (tabA)`. Wszystkie operacje są wykonywane przez funkcję na oryginalnej tablicy i zmiany są pamiętane po zakończeniu działania funkcji.

Zadania

* 1 Napisz program sprawdzający, czy tablica 10 liczb jest uporządkowana. Program powinien najpierw wczytać liczby do tablicy.

* 2 Napisz program, który po wczytaniu 10 liczb dodatnich całkowitych znajdzie wśród nich parę liczb o największej różnicy:
a. bez sortowania danych,
b. po posortowaniu danych.

* 3 Napisz program, który będzie porządkować malejąco tablicę 10 różnych liczb metodą:
a. bąbelkową,
b. przez wstawianie.

* 4 Przygotuj krótką prezentację na temat systemu „każdy z każdym", stosowanego w wielu dyscyplinach sportu. W prezentacji przedstaw zasady tworzenia par turniejowych dla kolejnych rund na przykładzie 8 zawodników (lub drużyn).

* 5 Przygotuj krótkie wystąpienie o historii powstania książki *Skafander i motyl*. Wyjaśnij, dlaczego jej autor posługiwał się w komunikacji ze światem alfabetem o zmienionym porządku liter: E S A R I N T U L O M D P C F B V H G J Q Z Y X K W.

** 6 Poszukaj informacji o algorytmie sortowania przez wybór. Napisz program, który będzie sortować liczby całkowite tą metodą.

** 7 Dodaj do programu *Sortowanie bąbelkowe* kod zliczający liczbę wykonanych prostych zamian. Podaj liczbę operacji zamian potrzebnych do uporządkowania zbioru:
a. 10-elementowego,
b. 20-elementowego,
c. 100-elementowego.
Dane do posortowania otrzymasz od nauczyciela w pliku tekstowym (np. o nazwie *posortuj_liczby.txt*).

** 8 Napisz program, który w tablicy zawierającej liczby nieujemne przesuwa na koniec tablicy wszystkie elementy o wartości 0. Porządek pozostałych elementów tablicy powinien zostać zachowany. Program nie powinien używać dodatkowej tablicy podczas przetwarzania danych.

** 9 Napisz program, który zamieni miejscami elementy znajdujące się w pierwszej i drugiej połowie tablicy o parzystej liczbie elementów.

Dane wejściowe:	3	1	4	5	2	8	0	3
Dane wyjściowe:	2	8	0	3	3	1	4	5

Program nie powinien używać dodatkowej tablicy pomocniczej.

** 10 Analizując sortowanie bąbelkowe zbioru [6, 2, 5, 4, 1], można zauważyć, że wartość 6 pojawi się na właściwym miejscu po stosunkowo niewielkiej liczbie operacji, a przeniesienie wartości 1 wymaga o wiele większej liczby operacji. Mówi się, że element o wartości 1 jest w tej sytuacji żółwiem, a element o wartości 6 – zającem. Jak zmodyfikować algorytm, żeby zaradzić tej asymetrii? Przedstaw ulepszoną wersję algorytmu, nazywaną czasami sortowaniem koktajlowym (ang. *shake sort*).

** 11 Poszukaj informacji o algorytmie sortowania bibliotecznego (ang. *library sort*), który został opublikowany w 2006 r. Przygotuj prezentację na temat tego algorytmu. Wyjaśnij pochodzenie jego nazwy.

** 12 Napisz program, który będzie porządkował wyrazy według ich długości. Wyrazy o tej samej długości należy ustawić w kolejności alfabetycznej.

*** 13 Napisz program sortujący metodą przez wstawianie z użyciem wartownika.
Wskazówka: Wartownika umieść jako dodatkowy element w pierwszej komórce tablicy zawierającej liczby do posortowania.

*** 14 W książce *Kalejdoskop matematyczny* Hugo Steinhaus opisał algorytm sortowania przez wstawianie z wyszukiwaniem binarnym. Przygotuj prezentację na temat tego algorytmu i wykorzystania w nim wagi szalkowej. Jakie znaczenie dla tego algorytmu ma ciąg 0, 1, 3, 5, 8, 11, 14, 17, 21, ...?

*** 15 Napisz program, który po wczytaniu 20 liczb dodatnich całkowitych znajdzie wśród nich parę liczb o najmniejszej różnicy.