Temat C3. Stosowanie tablic w języku C++ i list w języku Python

1. Zmienne indeksowane
2. Deklarowanie tablicy w języku C++
3. Definiowanie listy w języku Python
4. Wprowadzanie elementów do tablicy lub listy i wyprowadzanie elementów na ekran
5. Wykonywanie operacji na elementach tablicy lub listy

Warto powtórzyć

1. Do czego służą funkcje w językach programowania?
2. Kiedy stosujemy funkcję zwracającą wartość, a kiedy niezwracającą wartości?
3. W jaki sposób definiujemy funkcje w wybranym języku programowania (C++ lub Python)?
4. Jak wywołujemy funkcję z parametrami, a jak funkcję bez parametrów?

1. Zmienne indeksowane

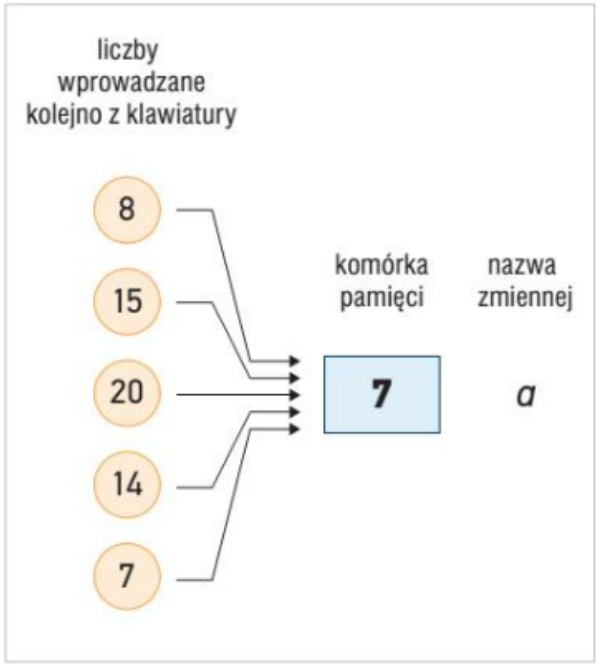
W funkcji *suma* utworzonej w temacie C2 wszystkie wprowadzane liczby komputer pamiętał kolejno w jednej zmiennej (o nazwie *a*), czyli nie zapamiętywał wielkości poszczególnych dostaw. W jaki sposób zapamiętać wszystkie wprowadzone dane?

Gdy z klawiatury wprowadzimy kolejno liczby, na przykład wielkości dostaw telewizorów ze zbioru {8, 15, 20, 14, 7}, każda wprowadzana liczba zastępuje poprzednią (rys. 1a).

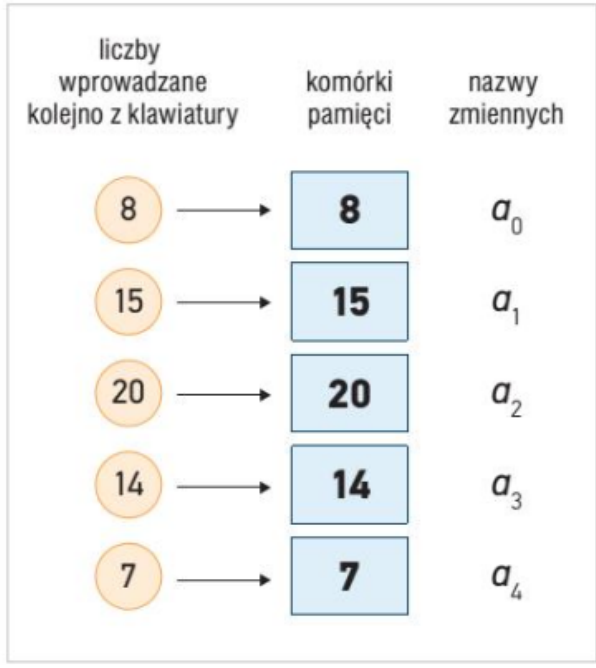
Aby zapamiętać wszystkie dane, musimy je różnie nazwać. Moglibyśmy użyć liter alfabetu, ale w przypadku dużej liczby danych zabrakłoby liter. Ponadto odwoływanie się do wielu zmiennych byłoby niewygodne. Dlatego stosujemy tzw. **zmienne indeksowane**, w których do nazwy zmiennej (np. *a*) dodaje się kolejny numer (**indeks**), np.: a_0, a_1, …, a_{n-1}. Wówczas dla każdej zmiennej komputer rezerwuje oddzielną komórkę pamięci, np. na poszczególne wielkości dostaw telewizorów z pięciu transportów (rys. 1b).

Ćwiczenie 1. Wyjaśniamy, czym są zmienne indeksowane

Objaśnij, czym są zmienne indeksowane, korzystając z przygotowanych pomocy dydaktycznych, np. kartek z przykładowymi liczbami.



Rys. 1a. *Zapisywanie w pamięci komputera wartości danych wprowadzanych z klawiatury z użyciem jednej zmiennej*



Rys. 1b. *Zapisywanie w pamięci komputera wartości danych wprowadzanych z klawiatury z użyciem pięciu zmiennych indeksowanych*

2. Deklarowanie tablicy w języku C++

Aby wykorzystać zmienne indeksowane w języku C++, musimy utworzyć specjalną strukturę danych – **tablicę**.

Deklaracja tablicy	`opis_typu nazwa_tablicy [liczba_elementów_tablicy];`	C++

Tabela 1. *Ogólna postać deklarowania tablicy w języku C++*

Do elementów tablicy odwołujemy się, podając nazwę tablicy i indeks elementu umieszczony w nawiasach kwadratowych, np.: $a[0]$, $a[1]$, …, $a[n - 1]$ – dla tablicy n-elementowej o nazwie a. W języku C++ pierwszy indeks jest zawsze równy 0.

Liczbę elementów tablicy musimy określić wcześniej:

- wpisać w deklaracji tablicy jako konkretną wartość (liczbę naturalną większą od 0)

  lub
- podać za pomocą wcześniej zdefiniowanej stałej (`const`).

Nie można zmienić liczby elementów raz zadeklarowanej tablicy.

! Uwaga

Stałe w językach C++ i Python będziemy zapisywać wielkimi literami. W obydwu językach wielkie i małe litery są rozróżniane (np. *n* i *N* oznaczają dwie różne zmienne lub stałe).

Deklaracja tablicy	Opis	Sposób odwołania
`int sztuki[100];`	oznacza zadeklarowanie tablicy o nazwie *sztuki* składającej się ze stu liczb całkowitych o indeksach od 0 do 99	do elementów tablicy odwołujemy się przez zmienne: *sztuki*[0], *sztuki*[1], …, *sztuki*[99]
`const int N = 20;` `int a[N];`	oznacza zadeklarowanie tablicy o nazwie *a* składającej się z dwudziestu elementów typu całkowitego (**int**).	do elementów tablicy odwołujemy się przez zmienne: *a*[0], a[1], …, *a*[*N*-1]
`const int LICZBA_EL = 80;` `float tablica[LICZBA_EL];`	oznacza utworzenie tablicy o nazwie *tablica* składającej się z osiemdziesięciu elementów typu rzeczywistego (**float**)	do elementów tablicy odwołujemy się przez zmienne: *tablica*[0], *tablica*[1], …, *tablica*[*LICZBA_EL*-1]

Tabela 2. *Przykłady deklarowania tablicy w języku C++*

Początkowe wartości elementów tablicy można podać w momencie jej definiowania, zapisując je w nawiasach klamrowych.

Deklaracja tablicy	Opis
`int DOSTAWY[5] = {8,15,20,14,7};`	tablica *DOSTAWY* składająca się z pięciu liczb całkowitych zostanie wypełniona liczbami ze zbioru {8, 15, 20, 14, 7}
`int DOSTAWY[] = {8,15,20,14,7};`	jak wyżej, ale liczba elementów tablicy *DOSTAWY* (pięć elementów) zostanie ustalona na podstawie liczby danych początkowych
`int sztuki[100] = {0};`	tablica *sztuki*, składająca się ze stu liczb całkowitych, zostanie wypełniona liczbami 0

Tabela 3. *Przykłady ustalania początkowych wartości tablicy (inicjalizowania tablicy) w języku C++*

3. Definiowanie listy w języku Python

Aby wykorzystać zmienne indeksowane w języku Python, możemy zdefiniować specjalną strukturę danych – **listę**.

Definicja listy	`nazwa_listy = [element1, element2, …, element_n]`	Python

Tabela 4. *Ogólna postać definiowania listy w języku Python*

Do elementów listy odwołujemy się, podając nazwę listy i indeks elementu umieszczony w nawiasach kwadratowych, np. $a[0], a[1], \ldots, a[n-1]$ – dla listy n-elementowej o nazwie a. W języku Python pierwszy indeks jest zawsze równy 0.

Lista (w odróżnieniu od tablicy) może zawierać elementy różnego typu, dlatego w języku Python nie określamy typu elementów listy.

Zarówno typ zmiennych przechowywanych w liście, jak i liczba elementów listy mogą się zmienić w trakcie działania programu.

Wartości elementów listy możemy:

- podać od razu w definicji listy,
- przypisać od razu taką samą wartość wszystkim elementom listy.

Liczbę elementów listy możemy przypisać zmiennej.

Uwaga

Liczbę przykładów definiowania list w języku Python ograniczyliśmy do kilku przydatnych do dalszych ćwiczeń.

Definicja listy	Opis	Sposób odwołania
`DOSTAWY = [8,15,20,14,7]`	oznacza zdefiniowanie listy o nazwie *DOSTAWY*, składającej się z pięciu elementów o indeksach od 0 do 4	do elementów listy odwołujemy się przez zmienne: *DOSTAWY*[0], *DOSTAWY*[1], …, *DOSTAWY*[4]
`lista_elementow = [0] * 20`	oznacza zdefiniowanie listy o nazwie *lista_elementow* składającej się z dwudziestu elementów, z których każdy jest równy zero (wyzerowanie wszystkich elementów listy)	do elementów listy odwołujemy się przez zmienne: *lista_elementow* [0], *lista_elementow*[1], …, *lista_elementow*[19]
`N = 50` `a = [0] * N`	oznacza zdefiniowanie listy o nazwie *a* składającej się z pięćdziesięciu elementów o wartości początkowej zero	do elementów listy odwołujemy się przez zmienne: *a*[0], *a*[1], …, *a*[N-1]

Tabela 5. *Przykłady definiowania listy w języku Python*

4. Wprowadzanie elementów do tablicy lub listy i wyprowadzanie elementów na ekran

Do wprowadzania danych do tablicy (C++) i listy (Python) oraz do wyprowadzania elementów możemy zastosować funkcję niezwracającą wartości bez parametrów lub z parametrami.

Przykład 1. Deklarowanie i stosowanie tablicy w języku C++

Zadanie: Wprowadź *N* liczb całkowitych do tablicy o nazwie *a*, następnie wyprowadź w kolumnie elementy tablicy na ekran. Zdefiniuj dwie funkcje niezwracające wartości bez parametrów *wprowadz_dane()* i *wyprowadz_dane()*, które wywołaj w programie głównym.

Dane: stała wartość *N*, tablica liczb *a*[*N*].

Wyniki: wyświetlone na ekranie w kolumnie elementy tablicy *a*: *a*[0], a[1], …, *a*[*N*-1].

C++ [*] Elementy_tablicy.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  const int N = 5;
5  int a[N];
6
7  void wprowadz_dane()
8  {
9      for(int i = 0; i < N; i++)
10     {
11         cout << "Podaj dana nr " << i << ": ";
12         cin >> a[i];
13     }
14 }
15
16 void wyprowadz_dane()
17 {
18     for(int i = 0; i < N; i++)
19         cout << "a[" << i << "] = " << a[i] << endl;
20 }
21
22 main ()
23 {
24     wprowadz_dane();
25     wyprowadz_dane();
26     return 0;
27 }
```

deklaracja stałej *N* o wartości 5

deklaracja tablicy o nazwie *a*

definicja funkcji *wprowadz_dane*

definicja funkcji *wyprowadz_dane*

wywołanie funkcji *wprowadz_dane*

wywołanie funkcji *wyprowadz_dane*

Przykład 2. Definiowanie i stosowanie listy w języku Python

Zadanie: Wprowadź *N* liczb całkowitych do listy o nazwie *a*, następnie wyprowadź w kolumnie elementy listy na ekran. Zdefiniuj dwie funkcje niezwracające wartości bez parametrów *wprowadz_dane()* i *wyprowadz_dane()*, które wywołaj w programie głównym.

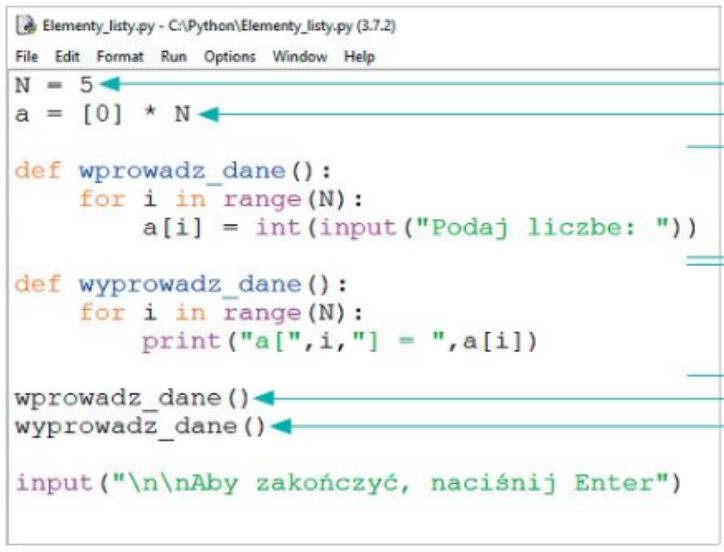
Dane: stała wartość *N*, lista liczb *a*[*N*].

Wyniki: wyświetlone w kolumnie elementy listy *a*: *a*[0], *a*[1], …, *a*[*N*-1].

Uwagi:

- Funkcja *wprowadz_dane* zmodyfikowała wartości istniejących zmiennych (które na początku były równe zero).
- Zapisany w języku Python program można również uruchomić, gdy zamkniemy okno z programem. Należy w tym celu z menu kontekstowego ikony programu (widocznej w Eksploratorze plików) wybrać polecenie:
  - **Otwórz** – w otwartym oknie zobaczymy uruchomiony program,
  - **Edit with IDLE** – program otworzy się w oknie powłoki Pythona (Python Shell) i będzie można go uruchomić.

Python



określenie liczby elementów listy

zdefiniowanie listy o nazwie *a* i wyzerowanie jej elementów

definicja funkcji *wprowadz_dane*

definicja funkcji *wyprowadz_dane*

wywołanie funkcji *wprowadz_dane*

wywołanie funkcji *wyprowadz_dane*

Ćwiczenie 2. Wprowadzamy dane do tablicy lub listy i wyprowadzamy je na ekran

1. Utwórz nowy plik źródłowy i przepisz program z przykładu 1. lub z przykładu 2. Zapisz program w pliku pod nazwą *Elementy_tablicy* lub *Elementy_listy*.
2. Uruchom i przetestuj program dla kilku różnych wartości elementów tablicy *a* lub listy *a*. Wyjaśnij znaczenie poszczególnych wierszy programu.

! Uwaga

W języku Python wszystkie zmienne, których nazwa jest zapisana wielkimi literami (i ewentualnie z podkreślnikami) będziemy traktować w programie jak stałe.

5. Wykonywanie operacji na elementach tablicy lub listy

Jeśli wszystkie wprowadzone dane pamiętamy w tablicy lub liście, możemy już po ich wprowadzeniu wykonywać na nich różne operacje, np. wyświetlić w innej kolejności, zsumować wybrane elementy czy uporządkować je (zastosowanie tablic i list w algorytmach porządkowania poznamy w temacie C4).

Ćwiczenie 3. Wyprowadzamy elementy z tablicy lub listy w odwrotnej kolejności

1. Otwórz program *Elementy_tablicy* lub *Elementy_listy* zapisany w ćwiczeniu 2. Zmodyfikuj program tak, aby dane wyświetlały się w odwrotnym porządku. Dodatkowo zwiększ liczbę danych do dziesięciu.
2. Zapisz program w pliku pod nazwą *Odwrotna_kolejnosc*.

Wskazówki:

C++ W instrukcji iteracyjnej **`for`** użyj odpowiedniej wartości początkowej, warunku zakończenia pętli oraz operatora dekrementacji `i--` zamiast operatora inkrementacji `i++`.

Python W instrukcji iteracyjnej **`for`** użyj w funkcji `range` trzech argumentów: `range(początek, koniec, krok)`.

Ćwiczenie 4. Wyświetlamy na ekranie wybrany element tablicy lub listy

1. Otwórz program *Elementy_tablicy* lub *Elementy_listy* zapisany w ćwiczeniu 2. Zmodyfikuj program tak, aby na ekranie wyświetlać tylko element *k*-ty, gdzie *k* jest liczbą całkowitą wprowadzaną z klawiatury po uruchomieniu programu.
2. Zapisz program w pliku pod nazwą *Wybrany_element*.

Wskazówka: Należy sprawdzić, czy wprowadzona wartość *k* jest większa lub równa 0 oraz mniejsza od *N*.

Ćwiczenie 5. Porównujemy dwa elementy tablicy lub listy i wyświetlamy na ekranie większy z nich

1. Korzystając z funkcji *wprowadz_dane()* z programów *Elementy_tablicy* lub *Elementy_listy* (ćwiczenie 2.) oraz programu *Sumy_dostaw* (ćwiczenie 1., temat C2), napisz program sprawdzający dla dziesięciu dostaw, która dostawa telewizorów jest większa: pierwsza czy ostatnia. Zależnie od wyniku, program powinien wyprowadzać na ekran komunikat: „Wieksza jest pierwsza" lub „Wieksza jest ostatnia". W przypadku dostaw o tej samej liczbie telewizorów wyprowadź komunikat „Dostawy rowne".
2. Zapisz program w pliku pod nazwą *Wieksza*.

Warto zapamiętać

- Aby zaprogramować algorytm, w którym w trakcie działania programu musimy pamiętać wszystkie elementy zbioru, możemy zapisywać dane w tablicy (C++) lub na liście (Python).
- W języku C++ liczba elementów (rozmiar) tablicy musi być z góry określona, np. jako zdefiniowana wcześniej stała. Musimy również określić typ elementów.
- W języku Python nie określamy typu elementów listy. Interpreter określa typ elementów na podstawie rodzaju danych, jaki im przypiszemy.
- Do elementów tablicy (C++) i listy (Python), np. o nazwie *lista* i liczbie elementów *N*, odwołujemy się poprzez podanie nazwy i indeksu umieszczonego w nawiasach kwadratowych *lista*[0], *lista*[1], …, *lista*[*N* - 1].

Pytania i polecenia

1. Czym jest zmienna indeksowana?
2. W jakim celu definiujemy tablicę? Podaj przykład użycia tablicy.
3. Wyjaśnij na przykładach sposób deklarowania tablicy w języku C++.
4. W jakim celu definiujemy listę? Podaj przykład użycia listy.
5. Wyjaśnij na przykładach sposób definiowania listy w języku Python.

Zadania

1. Otwórz program *Elementy_tablicy* lub *Elementy_listy* zapisany w ćwiczeniu 2. Zmodyfikuj program tak, aby na ekranie wyświetlić tylko pierwszy i ostatni element. Zapisz program w pliku pod nazwą *Element_pierwszy_i_ostatni*.

2. Otwórz program *Element_pierwszy_i_ostatni* zapisany w zadaniu 1. Zmodyfikuj program tak, aby sprawdzać, który element jest większy: pierwszy czy drugi. Wyświetlaj tylko element większy oraz komunikat, który to element. W przypadku elementów równych wyświetlaj komunikat „Elementy rowne". Zapisz program w pliku pod nazwą *Element_wiekszy*.
3. Otwórz program *Elementy_tablicy* lub *Elementy_listy* zapisany w ćwiczeniu 2. Dodaj do programu definicję funkcji (typu całkowitego w języku C++) *suma_danych*(), obliczającej sumę elementów tablicy lub listy i zwracającą tę wartość do programu głównego. Zapisz program w pliku pod nazwą *Suma_elementow*.
   Wskazówka: W treści funkcji umieść instrukcję: `suma += a[i]` odpowiadającą instrukcji `suma = suma + a[i]`.
4. Napisz program, który wczytuje liczby całkowite do ośmioelementowej tablicy lub listy i wypisuje na ekran indeks pierwszego elementu tablicy lub listy, którego wartość wynosi zero. Jeśli nie ma takiego elementu, program powinien wypisać komunikat „Brak elementu zero". Zapisz program w pliku pod nazwą *Indeks_elementu_zero*.
5. Zadeklaruj tablicę (C++) lub zdefiniuj listę (Python) składającą się z pięciu elementów, będących liczbami godzin lekcyjnych od poniedziałku do piątku. Utwórz program wypisujący w kolumnie elementy tablicy lub listy, czyli liczby lekcji każdego dnia. Zapisz program w pliku pod nazwą *Lekcje_w_tygodniu*.
   Wskazówka: Zastosuj sposób definiowania tablicy pokazany w pierwszym wierszu tabeli 3. lub listy pokazany w pierwszym wierszu tabeli 5.
6. Korzystając z tablicy lub z listy zdefiniowanej w zadaniu 5., utwórz program, który dla podanego z klawiatury numeru dnia tygodnia wyświetli liczbę godzin lekcji tego dnia. Zapisz program w pliku pod nazwą *Lekcje_w_dniu*.
7. Zmodyfikuj program *Sumy_dostaw* zapisany w ćwiczeniu 1. z tematu C2 tak, aby liczby dostaw telewizorów były pamiętane jako elementy tablicy lub listy *tv*[*T*], a liczby dostaw głośników jako elementy tablicy lub listy *glosniki*[*G*], gdzie *T* i *G* są stałymi równymi 5. Wyprowadzaj na ekran elementy obydwu tablic, obliczaj sumy dostaw telewizorów i głośników i wyprowadź je na ekran. Zapisz program w pliku pod nazwą *Sumy_dostaw_tablice* lub *Sumy_dostaw_listy*.

Dla zainteresowanych

8. Otwórz program *Suma_elementow* zapisany w zadaniu 3. Zwiększ wielkość tablicy (listy) do 10 ($N = 10$). Zmodyfikuj funkcję *suma*(), dodając parametr *ile* określający, ile początkowych elementów tablicy lub listy ma być zsumowanych. W programie głównym wywołaj funkcję z parametrem aktualnym *liczba_el* wprowadzanym z klawiatury. Sprawdź, czy jego wartość jest poprawna (czyli większa lub równa 0 i mniejsza lub równa *N*). Jeżeli nie jest, wyświetl komunikat „Wprowadzana wartosc danych jest spoza zakresu". Zapisz program w pliku *Suma_elementow_ile_z8*.
9. Otwórz program *Suma_elementow_ile_z8* zapisany w zadaniu 8. Zwiększ wielkość tablicy (listy) do 100 ($N = 100$). Zmodyfikuj funkcje wprowadzania i wyprowadzania danych, dodając do każdej parametr *ile* określający, ile elementów tablicy lub listy ma być wprowadzonych i wyprowadzonych. Wszystkie operacje wykonuj dla poprawnych danych. Zapisz program w pliku *Suma_elementow_ile_z9*.
   Wskazówka: W programie głównym sprawdzaj, czy wartość zmiennej *liczba_el* jest mniejsza lub równa *N*.
10. Zmodyfikuj program *Suma_elementow_ile_z9* zapisany w zadaniu 9. tak, aby zsumować oddzielnie elementy o indeksach parzystych i nieparzystych. Element o indeksie 0 dodaj do sumy elementów parzystych. Zapisz program w pliku *Suma_parzyste_i_nieprzyste*.