Tablice jednowymiarowe w języku C++

Źródło: Roman Synkevych, domena publiczna.

> W tym e-materiale powtarzamy wiadomości ze szkoły podstawowej.

Często zdarza się, że w programie trzeba przechować wiele danych tego samego typu. Np. przemysłowe systemy pomiarowe potrafią dostarczyć w ciągu godziny olbrzymich ilości informacji na temat temperatury, ciśnienia, przepływu albo innych parametrów, które muszą zostać w jakiś sposób zapisane.

W momencie tworzenia programu często nie jesteśmy w stanie oszacować, ile dokładnie danych będzie on używał. Nawet jeżeli wiemy, że nasz program potrzebowałby tysiąca zmiennych tekstowych, tworzenie ich ręcznie byłoby czasochłonne i niepraktyczne. Ponad tysiąc zmiennych byłoby nam potrzebnych np. do przechowywania napisów, jakie nasz program ma wyświetlać na ekranie.

O wiele lepszym sposobem przechowywania danych są w takich przypadkach tablice, o których mówiliśmy w e-materiale Tablice jednowymiarowe. Mają one wiele zastosowań. Służą nie tylko do składowania informacji, ale również ułatwiają ich analizowanie.

Ciekawi cię, jak wyglądają implementacje w innych językach programowania? Możesz się z nimi zapoznać w dwóch pozostałych e-materiałach z tej serii:

- Tablice jednowymiarowe w języku Java,

- Tablice jednowymiarowe w języku Python.

Więcej zadań? Sięgnij do Tablice jednowymiarowe – zadania maturalne.

> **Twoje cele**
>
> - Prześledzisz wszystkie informacje o tablicach jednowymiarowych.
> - Przeanalizujesz rolę tablic w programach.
> - Samodzielnie rozwiążesz zadany problem, korzystając z tablic jednowymiarowych.

Przeczytaj

Problem 1

Napisz program znajdujący maksymalny niemalejący, spójny podciąg w ciągu liczb. Przetestuj jego działanie dla tablicy [6, 7, 1, 3, 4] .

Specyfikacja problemu:

Dane:

- `liczby` – tablica liczb naturalnych
- `rozmiarTablicy` – liczba naturalna

Wynik:

Na standardowym wyjściu program wyświetla maksymalny niemalejący spójny podciąg.

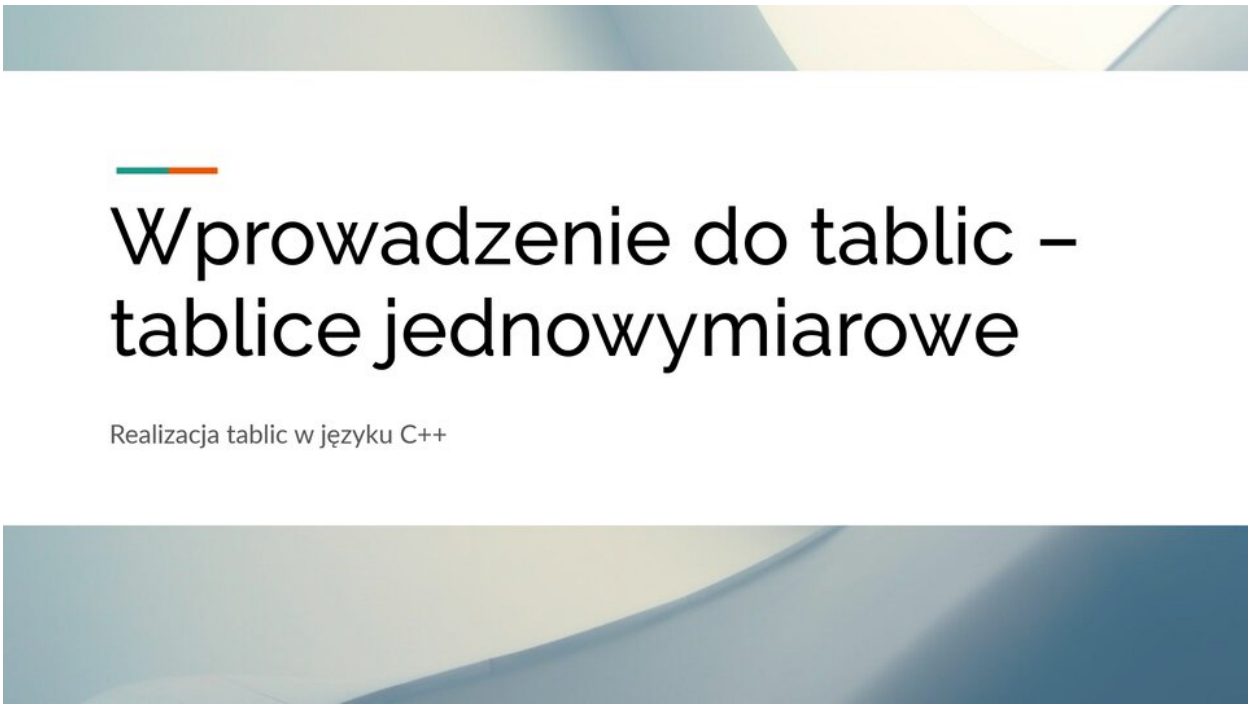
Kod:

```
#include <iostream>

using namespace std;

int main() {
    // tutaj dopisz kod
}
```

```
1
```

Polecenie 1

Porównaj swoje rozwiązanie z przedstawionym w filmie.



Film dostępny pod adresem /preview/resource/RL9cgcjkmLuUW

Film nawiązujący do treści materiału

Tablica jednowymiarowa – przypomnienie

Tablica jednowymiarowa jest uporządkowanym zbiorem elementów tego samego typu. Są one numerowane (indeksowane). Pierwszy element ma numer 0.

Tablica jednowymiarowa w języku C++

Oto deklaracja – i zarazem definicja – tablicy jednowymiarowej w języku C++:

```
1 int przykladowaTablica[10];
```

Definiowanie tablicy rozpoczynamy od wskazania typu danych, które będą w niej przechowywane – w tym przypadku są to liczby całkowite (`int`).

Następnie nadajemy nazwę tablicy – niech będzie to `przykladowaTablica`. Do nazwy tej będziemy się odwoływać, aby odczytać zawartość tablicy, wpisać konkretną liczbę w określonym miejscu albo wykonać operację na wybranym elemencie.

Za nazwą, w nawiasach kwadratowych, określamy rozmiar tablicy. Jest to liczba elementów, które da się zapisać w tablicy.

W języku C++ istnieją dwa rodzaje tablic. Jednym z nich są tablice statyczne, a drugim – tablice dynamiczne. Rozmiar tablicy statycznej jest ustalany podczas jej tworzenia i nie można go zmienić. W przypadku tablic dynamicznych jest inaczej – ich rozmiar zmienia się w zależności od liczby przechowywanych elementów. W tym e-materiale będziemy zajmować się tylko tablicami statycznymi.

> **Ciekawostka**
>
> Przy tworzeniu nowej tablicy, możemy od razu ją wypełnić. Robimy to w następujący sposób:
>
> ```
> 1 int liczby[3] = {1, 5, 9};
> ```
>
> Jeżeli liczba podanych elementów będzie mniejsza niż rozmiar tablicy, puste pola zostaną wypełnione zerami:
>
> ```
> 1 int liczby[4] = {3, 6, 9};
> ```
>
> Przedstawiona tablica będzie wyglądać następująco: [3, 6, 9, 0].
> **Ciekawostka**

> Warto zdefiniować zmienną służącą do przechowywania rozmiaru tablicy. Ułatwi to później operowanie na tablicy (np. w pętli `for`):
>
> ```
> int rozmiar = 10;
> int cyfry[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
> ```
>
> Jeżeli chcielibyśmy podać naszą zmienną `rozmiar` bezpośrednio w deklaracji tablicy, musiałaby ona być zadeklarowana jako stała, czyli `const` – jej rozmiar nie mógłby zostać zmieniony w trakcie działania programu.
>
> ```
> const int rozmiar = 10;
> int cyfry[rozmiar] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
> ```

Aby korzystać z tablic, musimy wiedzieć, jak umieszczać w nich dane i w jaki sposób odczytywać zapisane informacje. Przyjmijmy, że na pierwszym miejscu – w zdefiniowanej wcześniej tablicy `przykladowaTablica` – chcemy przechować liczbę 33. Robimy to, przypisując elementowi o indeksie 0 wartość 33:

```
przykladowaTablica[0] = 33;
```

Pamiętajmy, że elementy tablic są numerowane od zera. W przypadku naszej przykładowej tablicy pierwszy element ma indeks 0, a ostatni 9. A zatem po podaniu nazwy tablicy i wprowadzeniu w kwadratowych nawiasach indeksu 0 przypiszemy pierwszemu elementowi wartość 33.

Aby wypisać dane z wybranego elementu tablicy, należy podać jej nazwę wraz z odpowiednim indeksem w nawiasach kwadratowych:

```
std::cout << przykladowaTablica[0] << std::endl;
```

> **Ważne!**
>
> `<<< std::endl;` to manipulator strumienia; wstawia znak nowej linii i wymusza zapisanie danych z bufora.

Tablica jednowymiarowa – przykład

Wykonajmy proste zadanie. Z tablicy jednowymiarowej, wypełnionej liczbami, wypiszmy tylko te elementy, które są parzyste. Użyjemy operatora modulo oraz instrukcji warunkowej.

Oto szkic rozwiązania w pseudokodzie:

```
tablica = [3, 6, 9, 4, 8]
rozmiarTablicy = 5

dla i = 0, 1, ..., rozmiarTablicy - 1 wykonuj:
    jeżeli tablica[i] % 2 = 0 wykonaj:
        wypisz tablica[i]
```

Odczytujemy kolejne elementy tablicy, korzystając z pętli `for` i za pomocą instrukcji warunkowej sprawdzamy, czy reszta z dzielenia poszczególnych liczb przez 2 wynosi zero. Jeżeli tak właśnie jest, wypisujemy sprawdzany element na ekranie:

```
int tablica[5] = {3, 6, 9, 4, 8};
int rozmiarTablicy = 5;

for (int i = 0; i < rozmiarTablicy; i++) {
        if (tablica[i] % 2 == 0) {
            std::cout << tablica[i] << std::endl;
        }
}
```

Słownik

> **instrukcja warunkowa**
>
> element sterujący, który w zależności od spełnienia określonego warunku pozwala dokonać wyboru między różnymi zestawami poleceń
> **operator modulo**
>
> zwraca resztę z dzielenia liczb całkowitych; w języku C++ jest zapisywany za pomocą symbolu %

Prezentacja multimedialna

Polecenie 1

Napisz program znajdujący maksymalny niemalejący, spójny podciąg w n-elementowym ciągu liczb. Przetestuj jego działanie dla następującej tablicy: [5, 8, 3, 5, 9, 233, 5, 645, 543, 43, 23, 54, 54, 67, 87, 564, 543, 43, 54, 34].

Specyfikacja problemu:

Dane:

- `glownyCiag` – *n*-elementowa tablica składająca się z dodatnich liczb całkowitych

Wynik:

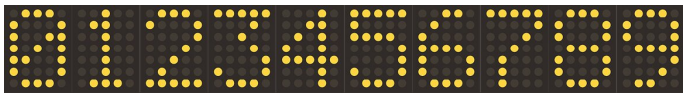
Program wyświetla maksymalny niemalejący spójny podciąg.

```
#include <iostream>

using namespace std;

int main() {
    // miejsce na twój kod
}
```

```
1
```

Polecenie 2

Porównaj swoje rozwiązanie z przedstawionym w prezentacji.



1

Niemalejącym nazywamy taki ciąg liczbowy, w którym każdy jego wyraz – z wyjątkiem pierwszego – jest nie mniejszy od poprzedniego
Źródło: Pixabay [dostęp 01.12.2022 r.], CC 0.

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P12p0ryiU

Przypomnijmy sobie treść zadania:

W otrzymanym zestawie liczb musimy znaleźć najdłuższy spójny podciąg niemalejący. Innymi słowy, w ciągu danych mamy wskazać najdłuższy fragment, w którym każdy kolejny element jest równy poprzedniemu lub od niego większy.

2

W omawianym programie korzystamy z tablicy statycznej – podczas jej tworzenia podaje się rozmiar, którego nie można później zmienić.
Źródło: Roth Melinda, unsplash.com [dostęp 01.12.2022 r.], domena publiczna.

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P12p0ryiU

Zacznijmy od zdefiniowania tablicy zawierającej dane do analizy.

```
1 int glownyCiag[20] = {5,
  8, 3, 5, 9, 233, 5, 645,
  543, 43, 23, 54, 54, 67,
  87, 564, 543, 43, 54, 34};
2 int glownyCiagDlugosc =
  20;
```

Do zbadania mamy ciąg złożony z 20 liczb. Dla wygody zdefiniowaliśmy zmienną `glownyCiagDlugosc`. Przechowuje ona informację o długości naszej tablicy.

3

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P12p0ryiU

Potrzebne są nam jeszcze inne zmienne:

```
1 int najdluzszyIndeks = -1;
```

```
int najdluzszaDlugosc =
-1;
int indeksTeraz = 0;
int dlugoscTeraz = 1;
```

Będziemy przechowywać informacje o długości i indeksie pierwszego elementu najdłuższego spójnego, niemalejącego podciągu. Posłużymy się zmiennymi `najdluzszyIndeks` oraz `najdluzszaDlugosc`.

Z kolei `indeksTeraz` oraz `dlugoscTeraz` będą zawierały informacje na temat sprawdzanego obecnie podciągu.

4

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P12p0ryiU

Następnym etapem jest zastosowanie pętli `for`, która pozwoli nam zbadać poszczególne elementy tablicy `glownyCiag`:

```
for (int i = 1; i <
glownyCiagDlugosc; i++) {
    // Instrukcje wewnątrz
pętli for
}
```

Jako początkową wartość zmiennej sterującej `i` wybieramy 1, a nie `0` (jak zazwyczaj się dzieje). Pierwszy element tablicy na pewno będzie częścią pierwszego podciągu, więc można go pominąć. Dzięki użyciu zmiennej `glownyCiagDlugosc` nasz kod jest bardziej czytelny oraz łatwiejszy do zmodyfikowania.

Materiał audio dostępny pod adresem:

5

https://zpe.gov.pl/b/P12p0ryiU

Zmienna sterująca wskazuje kolejne elementy tablicy `glownyCiag`. Aby stwierdzić, czy w następnej sekwencji pętli badany podciąg nadal jest niemalejący, musimy użyć instrukcji warunkowej.

```
for (int i = 1; i < glownyCiagDlugosc; i++) {
    if (glownyCiag[i] < glownyCiag[i - 1]) {
        // Kolejny element jest mniejszy od poprzedniego
    } else {
        // Kolejny element jest niemniejszy od poprzedniego
    }
}
```

Jeżeli poprzedni element jest większy od elementu o indeksie `i`, podciąg przestał być niemalejący. Co zrobić w takim przypadku?

6

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P12p0ryiU

Jeżeli podciąg przestał być niemalejący, musimy sprawdzić, czy jest on dłuższy niż wcześniej sprawdzone podciągi. Jeżeli tak jest, zapamiętujemy go jako najdłuższy do tej pory; jeżeli nie, ustawiamy wartość zmiennej `indeksTeraz` na `i` (będzie ona wskazywała miejsce, gdzie zakończył się poprzedni podciąg, a rozpocznie kolejny).

Ponieważ zaczynamy liczyć długość nowego podciągu, wartość zmiennej `dlugoscTeraz` musi zostać zmieniona na 1:

```
if (dlugoscTeraz > najdluzszaDlugosc) {
    najdluzszaDlugosc = dlugoscTeraz;
    najdluzszyIndeks = indeksTeraz;
}

indeksTeraz = i;
dlugoscTeraz = 1;
```

Umieśćmy ten fragment kodu w programie.

7

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P12p0ryiU

```
for (int i = 1; i < glownyCiagDlugosc; i++) {
    if (glownyCiag[i] < glownyCiag[i - 1]) {

        if (dlugoscTeraz > najdluzszaDlugosc) {

najdluzszaDlugosc = dlugoscTeraz;

najdluzszyIndeks = indeksTeraz;
        }

        indeksTeraz = i;
        dlugoscTeraz = 1;
    } else {
        // Kolejny element jest niemniejszy od poprzedniego
    }
}
```

Trzeba teraz rozpatrzyć przypadek, w którym kolejny element tablicy nie jest mniejszy od poprzedniego. W takiej sytuacji podciąg pozostaje niemalejący. Co należy zrobić?

8

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P12p0ryiU

Powinniśmy dodać jeden do długości bieżącego podciągu, czyli do zmiennej `dlugoscTeraz`:

```
dlugoscTeraz++;
```

Musimy również wziąć pod uwagę sytuację, w której podciąg nie przestaje rosnąć aż do końca tablicy. Czynności, które wykonujemy, gdy podciąg przestaje być niemalejący, należy powtórzyć, gdy kończy się tablica.

Uzupełnijmy pętlę `for` o brakujące instrukcje.

9

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P12p0ryiU

```
for (int i = 1; i < glownyCiagDlugosc; i++) {
    if (glownyCiag[i] < glownyCiag[i - 1]) {

        if (dlugoscTeraz > najdluzszaDlugosc) {

najdluzszaDlugosc = dlugoscTeraz;

najdluzszyIndeks = indeksTeraz;
```

```
        }

        indeksTeraz = i;
        dlugoscTeraz = 1;
    } else {
        dlugoscTeraz++;

        if (i ==
glownyCiagDlugosc - 1) {

            if
(dlugoscTeraz >
najdluzszaDlugosc) {

najdluzszaDlugosc =
dlugoscTeraz;

najdluzszyIndeks =
indeksTeraz;
            }
        }
    }
}
```

Analityczna część programu jest gotowa. Przedstawmy zatem wyniki na ekranie.

10

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P12p0ryiU

Gdy aplikacja zakończy działanie, przedstawi indeks elementu rozpoczynającego najdłuższy podciąg oraz poda jego długość. Możemy wykorzystać te informacje i zaprezentować rekordowy podciąg.

```
for (int i = 1; i <
glownyCiagDlugosc; i++) {
    if (glownyCiag[i] <
glownyCiag[i - 1]) {

```

```
4         if (dlugoscTeraz > najdluzszaDlugosc) {
5 
              najdluzszaDlugosc = dlugoscTeraz;
6 
              najdluzszyIndeks = indeksTeraz;
7         }
8 
9         indeksTeraz = i;
10        dlugoscTeraz = 1;
11    } else {
12        dlugoscTeraz++;
13 
14        if (i == glownyCiagDlugosc - 1) {
15 
16            if (dlugoscTeraz > najdluzszaDlugosc) {
17 
                  najdluzszaDlugosc = dlugoscTeraz;
18 
                  najdluzszyIndeks = indeksTeraz;
19            }
20        }
21    }
22 }
23 
24 for (int i = najdluzszyIndeks; i < najdluzszaDlugosc + najdluzszyIndeks; i++) {
25     cout << glownyCiag[i] << " ";
26 }
27 
28 cout << endl;
```

W ostatnim okienku pokazujemy końcową wersję kodu.

```cpp
#include <iostream>

using namespace std;

int main() {
    int glownyCiag[20] = {5, 8, 3, 5, 9, 233, 5, 645, 543, 43, 23, 54, 54, 67, 87, 564, 543, 43, 54, 34};
    int glownyCiagDlugosc = 20;

    int najdluzszyIndeks = -1;
    int najdluzszaDlugosc = -1;
    int indeksTeraz = 0;
    int dlugoscTeraz = 1;

    for (int i = 1; i < glownyCiagDlugosc; i++) {
        if (glownyCiag[i] < glownyCiag[i - 1]) {

            if (dlugoscTeraz > najdluzszaDlugosc) {

najdluzszaDlugosc = dlugoscTeraz;

najdluzszyIndeks = indeksTeraz;
            }

            indeksTeraz = i;
            dlugoscTeraz = 1;
        } else {
```

```
25
                dlugoscTeraz++;
26
27              if (i ==
                glownyCiagDlugosc - 1) {
28
29                  if
                    (dlugoscTeraz >
                    najdluzszaDlugosc) {
30
                        najdluzszaDlugosc =
                        dlugoscTeraz;
31
                        najdluzszyIndeks =
                        indeksTeraz;
32                  }
33              }
34          }
35      }
36
37      for (int i =
        najdluzszyIndeks; i <
        najdluzszaDlugosc +
        najdluzszyIndeks; i++) {
38          cout <<
            glownyCiag[i] << " ";
39      }
40
41      cout << endl;
42  }
```


Polecenie 3

Zapisz notatkę, w której podsumujesz najważniejsze informacje przedstawione w prezentacji multimedialnej.

Sprawdź się

Pokaż ćwiczenia: 

Ćwiczenie 1

Napisz program, który wypisuje co drugi element tablicy (bez spacji oraz bez przechodzenia do nowych linii) o rozmiarze `rozmiarTablicy`. Swój program przetestuj dla tablicy składającej się z 10 elementów, której zawartość to [2, 5, 8, 9, 4, 5, 7, 3, 5, 7].

Specyfikacja problemu:

Dane:

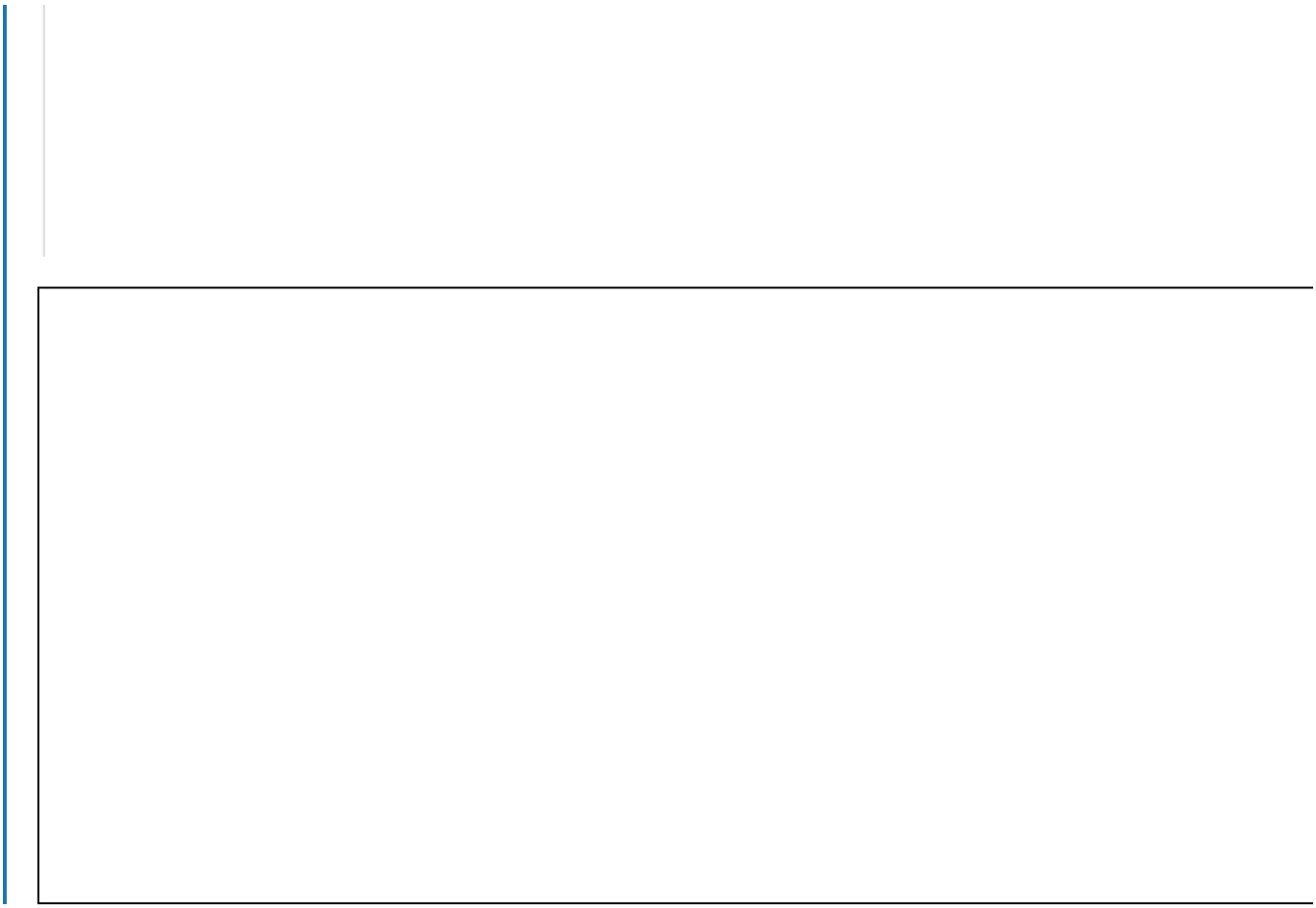
- `tablica` – tablica liczb naturalnych
- `rozmiarTablicy` – liczba naturalna

Wynik:

Na standardowe wyjście wyprowadzana jest co druga liczba umieszczona w tablicy.

Twoje zadania

1. Program wypisujący co drugi element tablicy (bez użycia spacji oraz bez przechodzenie do nowych linii).



Ćwiczenie 2

Napisz program, który znajdzie i wypisze elementy *n*-elementowej tablicy podzielne przez 3 oraz 2. Swój program przetestuj dla tablicy o zawartości [45, 643, 19, 21, 54, 654, 43, 76, 89, 66].

Specyfikacja problemu:

Dane:

- `tablica` – *n*-elementowa tablica liczb naturalnych

Wynik:

Program wypisuje elementy tablicy podzielne przez 3 oraz 2.

Twoje zadania

1. Program sprawdzający, które elementy tablicy są podzielne przez 3 oraz 2.

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Tablice jednowymiarowe w języku C++

Grupa docelowa:

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum, zakres podstawowy i rozszerzony

Podstawa programowa:

Cele kształcenia – wymagania ogólne

I. Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Treści nauczania – wymagania szczegółowe

I. Rozumienie, analizowanie i rozwiązywanie problemów.

Zakres podstawowy. Uczeń:

1) planuje kolejne kroki rozwiązywania problemu, z uwzględnieniem podstawowych etapów myślenia komputacyjnego (określenie problemu, definicja modeli i pojęć, znalezienie rozwiązania, zaprogramowanie i testowanie rozwiązania).

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.

Zakres podstawowy. Uczeń:

1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów, testuje poprawność programów dla różnych danych; w szczególności programuje algorytmy z punktu I.2);

2) do realizacji rozwiązań problemów prawidłowo dobiera środowiska informatyczne, aplikacje oraz zasoby, wykorzystuje również elementy robotyki;

I + II. Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

2) wykorzystuje znane sobie algorytmy przy rozwiązywaniu i programowaniu rozwiązań następujących problemów:

c) znajdowania w ciągu podciągów o różnorodnych własnościach, np. najdłuższego spójnego podciągu niemalejącego, spójnego podciągu o największej sumie,

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Prześledzisz wszystkie informacje o tablicach jednowymiarowych.
- Przeanalizujesz rolę tablic w programach.
- Samodzielnie rozwiążesz zadany problem, korzystając z tablic jednowymiarowych.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach.

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda;

- oprogramowanie dla języka C++, w tym kompilator GCC/G++ 4.5 (lub nowszej wersji) i Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio.

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Tablice jednowymiarowe w języku C++". Uczniowie zapoznają się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel wyświetla uczniom temat, wskazuje cele zajęć oraz ustala z uczestnikami zajęć kryteria sukcesu.
2. **Rozpoznanie wiedzy uczniów.** Nauczyciel prosi wybranego ucznia lub uczniów o przedstawienie sytuacji problemowej związanej z tematem lekcji.

Faza realizacyjna:

1. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Prezentacja multimedialna". Uczniowie w parach wykonują polecenie 1, a następnie porównują swoje rozwiązanie z przedstawionym w prezentacji.
2. **Ćwiczenie umiejętności.** Uczniowie wykonują ćwiczenia nr 1-2 z sekcji „Sprawdź się". Nauczyciel sprawdza poprawność wykonanych zadań, omawiając je wraz z uczniami.

Faza podsumowująca:

1. Wybrany uczeń podsumowuje zajęcia, zwracając uwagę na nabyte umiejętności, omawia ewentualne problemy podczas rozwiązania ćwiczeń z programowania w języku C++.

Praca domowa:

1. Uczniowie rozwiązują problem 1 z sekcji „Przeczytaj" i porównują swoje rozwiązanie z przedstawionym w filmie.

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla kompilatora GCC/G++ 4.5 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla języka C++.
- Oficjalna dokumentacja techniczna dla oprogramowania Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio.

Wskazówki metodyczne:

- Treści w sekcji „Prezentacja multimedialna” można wykorzystać na lekcji jako podsumowanie i utrwalenie wiedzy uczniów.