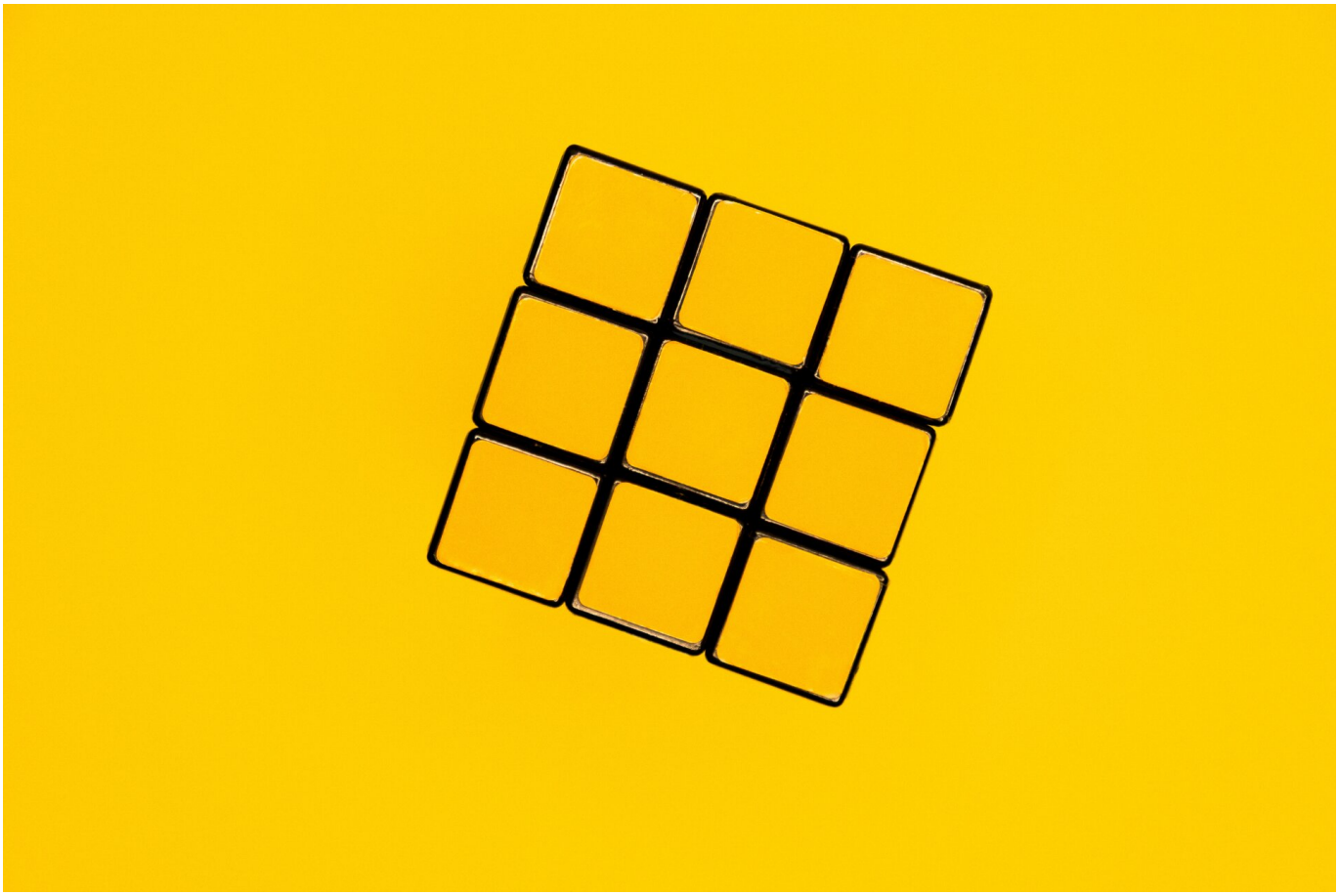


Jak rozwiązywać problemy programistyczne?

Źródło: Daniele Franchi, domena publiczna.

Informatyk w swojej pracy często spotyka się z problemami, które wymagają dodatkowej wiedzy i umiejętności. Dlatego też nieustannie musi doszkalać się w zakresie nowych technologii, korzystać z kursów oraz fachowej literatury. Wiele informacji można też znaleźć w internecie. W tym e-materiale omówimy najbardziej przydatne serwisy.

Więcej na temat rozwiązywania problemów informatycznych znajdziesz w e-materiałach:

- Rozwiązywanie problemów informatycznych – strategia „dziel i zwyciężaj”
- Praca z repozytorium Git
- Repozytorium kodu
- Zwinne metodyki tworzenia oprogramowania
- Projekt informatyczny: jak nim zarządzać?

Twoje cele

- Przedstawisz sposób przeszukiwania dowolnej strony internetowej za pomocą wybranej wyszukiwarki.
- Zapoznasz się z przykładowymi narzędziami, które wspierają pracę programisty.
- Rozwiążesz problemy informatyczne w kodzie za pomocą debuggera.

Przeczytaj

Wprowadzenie

Istnieje wiele dobrych praktyk dotyczących programowania i rozwiązywania problemów oraz mnóstwo sposobów i narzędzi, które umożliwiają wykrywanie błędów. W celu ułatwienia pracy programistom powstały specjalne edytory kodów źródłowych z różnymi funkcjami, np. analizatorami składniowymi oraz narzędziami do edycji kodu w kilku miejscach naraz. Dzięki temu wykrycie i naprawa błędu składniowego (literówka w nazwie zmiennej) czy semantycznego (odczytanie niezainicjowanej zmiennej) staje się znacznie prostsze. Ponadto zawsze możemy skorzystać z serwisów internetowych, które pomogą wyjaśnić problemy programistyczne.

Stack Overflow

Stack Overflow to jedno z najbardziej popularnych for informatycznych. Powstało w celu udzielania porad i korzystania z pomocy społeczności podczas wytwarzania oprogramowania. Każdy zarejestrowany użytkownik może zadać pytanie, dodając fragment kodu (demo) obrazujący dany problem. Strona cieszy się tak wielką popularnością, że można ją traktować jako zbiór rozwiązanych problemów programistycznych. Przy umiejętnym użyciu wyszukiwarki internetowej, istnieje duże prawdopodobieństwo, że to właśnie na tej stronie znajdziemy interesujące nas rozwiązanie.

> **Ciekawostka**
>
> Nazwę serwisu można przetłumaczyć jako „przepełnienie stosu". Oznacza to błąd programistyczny, zwykle spowodowany uruchomieniem nieskończonej rekurencji. Programy w systemie ładowane i uruchamiane są w pamięci RAM w strukturze nazywanej stosem. Każde wywołanie funkcji tworzy pewną liczbę danych na stosie. W momencie uruchomienia nieskończonej rekurencji następuje przepełnienie stosu, czyli wykroczenie poza przydzielony przez system obszar pamięci. W takiej sytuacji następuje błąd krytyczny i działanie programu zostaje przerwane przez system operacyjny.

Aby zobrazować zalety serwisu Stack Overflow, spróbujemy przeanalizować propozycje programów napisanych w językach: C++, Java oraz Python, pobierających dane od użytkownika.

Załóżmy, że dopiero rozpoczęliśmy naukę programowania. Jedną z rzeczy, którą na pewno wielokrotnie będziemy musieli zaprogramować, będzie pobranie danych od użytkownika.

Posłużmy się wyszukiwarką Google w celu przeszukania strony Stack Overflow, aby odnaleźć sposób na wczytanie danych od użytkownika. Wyszukiwarka Google umożliwia nam określenie, z jakiej strony chcielibyśmy wyświetlić wyniki. W tym celu do zapytania należy dopisać angielskie słowo „site:" (czyli: strona), a po dwukropku podać adres strony. W omawianym przypadku wpiszemy frazę: „site:stackoverflow.com" .

> **Ważne!**
>
> Zalecamy wpisywanie fraz w języku angielskim. Znacząco zwiększa to szansę odnalezienia odpowiedzi na szukane pytanie.

Język C++

Sprawdźmy zatem, co wyświetli się po zgłoszeniu wyszukania „c++ read input from user site:stackoverflow.com". Pierwszą pozycją, jaka wyświetli się w przeglądarce (dostęp: 06.05.2022 r.), jest strona zawierająca następujący kod programu:

Formatted I/O; taken from *Baby's First C++*:

14

```
#include <string>
#include <iostream>

int main()
{
  std::string name;
  std::cout << "Enter your name: ";
  std::getline(std::cin, name);
  std::cout << "Thank you, '" << name << "'." << std::endl;
}
```

This isn't quite satisfactory, as many things can (and thus will) go wrong. Here's a slightly more watertight version:

```
int main()
{
  std::string name;
  int score = 0;

  std::cout << "Enter your name: ";

  if (!std::getline(std::cin, name)) { /* I/O error! */ return -1; }

  if (!name.empty()) {
    std::cout << "Thank you, '" << name << "', you passed the test." << std::endl;
    ++score;
  } else {
    std::cout << "You fail." << std::endl;
    --score;
  }
}
```

Using `getline()` means that you might read an empty line, so it's worthwhile checking if the result is empty. It's also good to check for the correct execution of the read operation, as the user may pipe an empty file into stdin, for instance (in general, never assume that any particular circumstances exist and be prepared for anything). The alternative is token extraction, `std::cin >> name`, which only reads one word at a time and treats newlines like any other whitespace.

Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Otrzymaliśmy zatem informację, w jaki sposób możemy wczytać od użytkownika pewien napis, a także jak wykryć powstałe błędy.

Język Java

Dla zapytania „java read input from user site:stackoverflow.com" również otrzymujemy kilka wyników w wyszukiwarce. Pozycje położone najwyżej to strony z pytaniami najbardziej zbliżonymi do naszego oraz najchętniej odwiedzane. Wejdźmy w pierwszą z nich:

You can use any of the following options based on the requirements.

350

`Scanner` class

```
import java.util.Scanner;
//...
Scanner scan = new Scanner(System.in);
String s = scan.next();
int i = scan.nextInt();
```

`BufferedReader` and `InputStreamReader` classes

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
//...
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
String s = br.readLine();
int i = Integer.parseInt(s);
```

`DataInputStream` class

```
import java.io.DataInputStream;
//...
DataInputStream dis = new DataInputStream(System.in);
int i = dis.readInt();
```

The `readLine` method from the `DataInputStream` class has been *deprecated*. To get String value, you should use the previous solution with BufferedReader

`Console` class

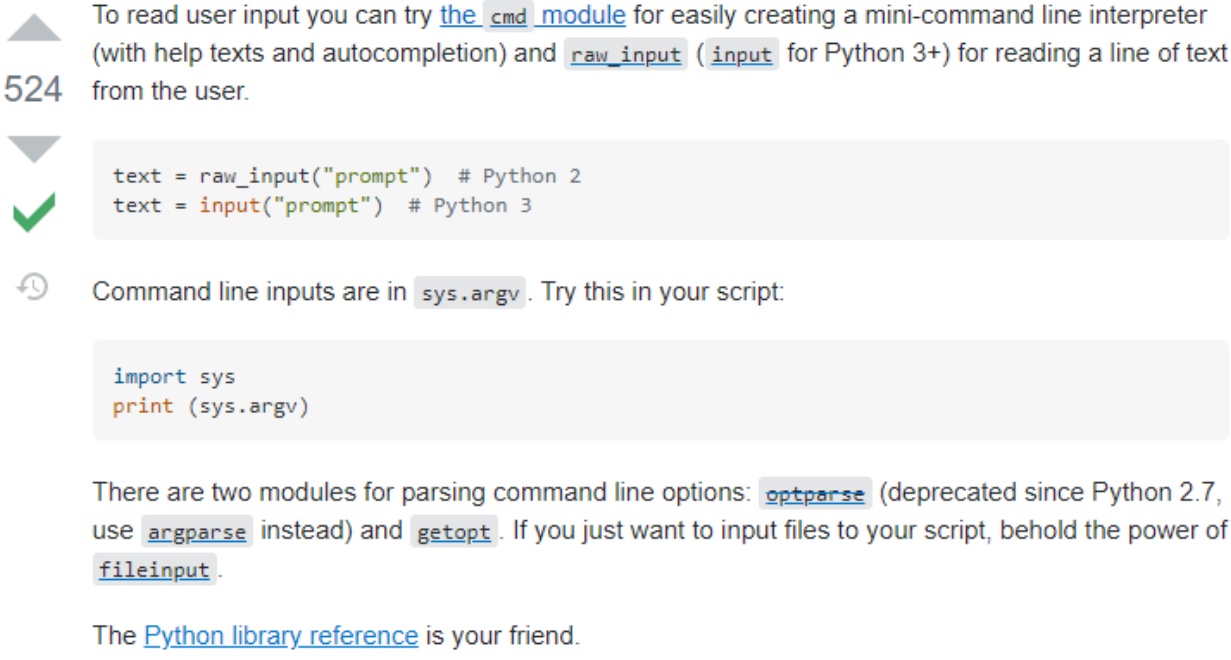
```
import java.io.Console;
//...
Console console = System.console();
String s = console.readLine();
int i = Integer.parseInt(console.readLine());
```

Apparently, this method does not work well in some IDEs.

Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Język Python

Dla języka Python zapytanie „python read input from user site:stackoverflow.com” zwróciło rezultat:



524

To read user input you can try the `cmd` module for easily creating a mini-command line interpreter (with help texts and autocompletion) and `raw_input` (`input` for Python 3+) for reading a line of text from the user.

```
text = raw_input("prompt")  # Python 2
text = input("prompt")  # Python 3
```

Command line inputs are in `sys.argv`. Try this in your script:

```
import sys
print (sys.argv)
```

There are two modules for parsing command line options: `optparse` (deprecated since Python 2.7, use `argparse` instead) and `getopt`. If you just want to input files to your script, behold the power of `fileinput`.

The Python library reference is your friend.

Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

W tym przypadku otrzymaliśmy również informację, w jaki sposób zrealizować oczekiwaną operację w dwóch wersjach języka Python. Ponieważ zadane pytanie składało się z dwóch części, uzyskana została odpowiedź na dwa pytania.

Dokumentacje

Ważnym elementem nauki każdego języka programowania jest zapoznanie się z jego dokumentacją. Przygotowana w odpowiedni sposób dokumentacja może sprawiać, że nauka będzie prosta, a wyszukiwanie nowych funkcji języka – bardzo sprawne.

Jak wiemy, na egzaminie maturalnym można posługiwać się jednym z następujących języków programowania: C++, Java i Python. Poniższej znajdziesz informacje na temat dotyczących ich dokumentacji:

C++:

- Cppreference – dokumentacja w języku angielskim (kompletna)
- Cplusplus – inna dokumentacja w języku angielskim (nieco bardziej przyjazna dla początkujących użytkowników)

Java:

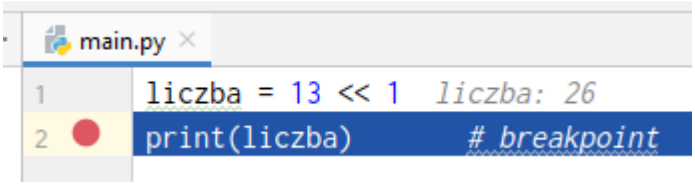
- Oracle – oficjalna dokumentacja języka Java

Python:

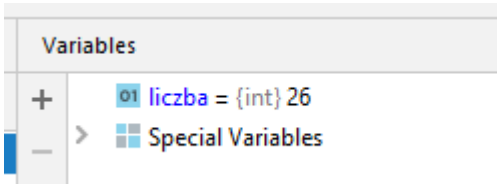
- Python – oficjalna dokumentacja języka Python

Debugger

Debugger to ważna część środowisk programistycznych, stosowana do wyszukiwania błędów, przede wszystkim semantycznych i logicznych, w tworzonym systemie (błędy składniowe wskazywane są przez kompilator). Zazwyczaj debugger dostarczany jest wraz z IDE. Uruchomienie programu w trybie debugowania pozwala na wykonanie instrukcji krok po kroku, podgląd wartości przechowywanych w pamięci w danym czasie, zatrzymywanie programu w momencie oznaczonym jako breakpoint.



W trybie debugowania w środowisku PyCharm w edytorze przy danej linijce kodu, wyświetlają się wartości, które przyjmują poszczególne zmienne.



Również w oknie debuggera możemy podejrzeć, jaką wartość przyjmuje zmienna w danym momencie.

Rozważmy następujący problem:

Dana jest tablica liczb całkowitych. Sprawdź, ile z tych liczb ma ostatnią cyfrę 2 i jest podzielnych przez 7. Znajdź tę liczbę.

Specyfikacja problemu:

Dane:

- `tablica` – tablica liczb całkowitych

Wynik:

- liczba liczb z tablicy dane podzielnych przez 7 oraz o ostatniej cyfrze równej 2

Trzech początkujących programistów napisało swoje rozwiązania w językach C++, Java oraz Python. Okazało się, że w każdej implementacji wystąpił błąd, który spowodował niepoprawne działanie programu. W celu zdiagnozowania problemu posłużymy się debuggerem.

> **Ważne!**
>
> Przedstawione kroki mogą różnić się w zależności od konfiguracji środowiska programistycznego.

C++

Kod napisany w języku C++ wygląda następująco:

```cpp
#include <iostream>

using namespace std;

int main() {
    int dane[9] = { 1, 5, 49, 212, 735, 231, 532, 103, 147 };

    int i = 0, k = 0;

    while (i < 9) {
        if (dane[i] % 10 == 2 && dane[i++] % 7 == 0) {
            k = k + 1;
        }
    }

    cout << k << endl;
```

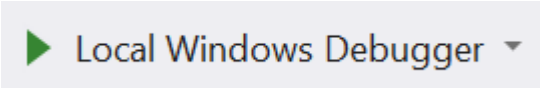
```
17 }
```

Po uruchomieniu programu zaczyna on wykonywać się w nieskończoność. Początkujący programista może nie wiedzieć, co jest przyczyną tego problemu. Spróbujmy zatem prześledzić problem w debuggerze. Ponieważ program wykonuje się w nieskończoność, łatwo stwierdzić, że błąd jest najprawdopodobniej spowodowany niepoprawnie skonstruowanym warunkiem końca. Ustawmy zatem pierwszy *breakpoint* w linijce 9. W środowisku Microsoft Visual Studio *breakpoint* ustawiamy, klikając lewym przyciskiem myszy na linijce przy numerze wiersza. Aby go usunąć, musimy kliknąć lewym przyciskiem myszy na czerwoną kropkę na pasku przerwań.

```
1   #include <iostream>
2
3   int main() {
4       int dane[9] = { 1, 5, 49, 212, 735, 231, 532, 103, 147 };
5
6       int i = 0, k = 0;
7
8       while (i < 9) {
9           if (dane[i] % 10 == 2 && dane[i++] % 7 == 0) {
10              k = k + 1;
11          }
12      }
13
14      std::cout << k << std::endl;
15  }
```

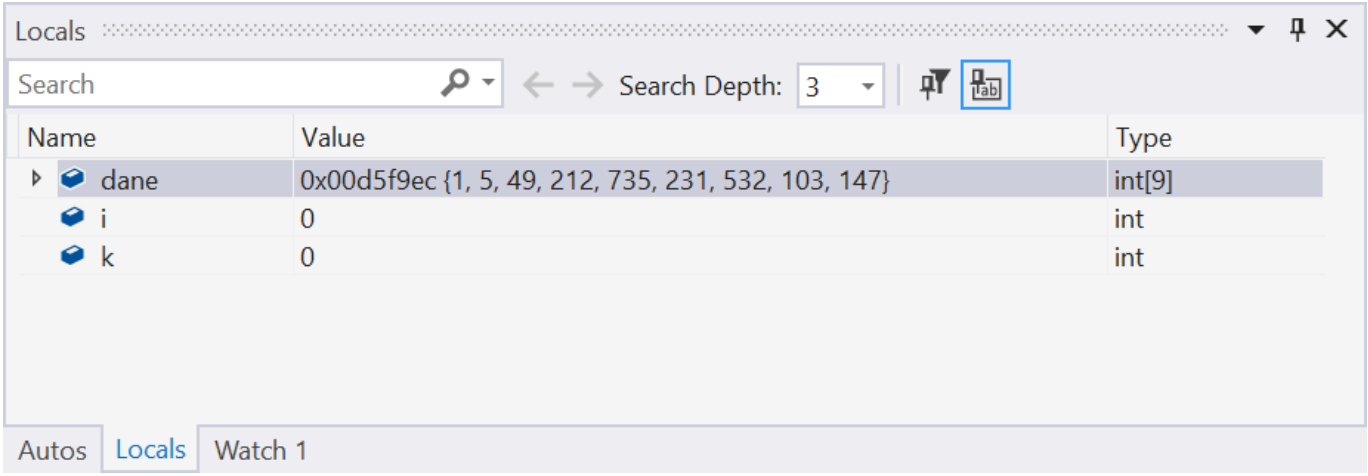
Poprawnie ustawiony *breakpoint* w środowisku Microsoft Visual Studio.

Uruchomimy program w trybie debugowania. W tym celu wybieramy zielony trójkąt z napisem `Local Windows Debugger` w górnej części programu Visual Studio lub przyciskamy klawisz `F5` na klawiaturze.



Przycisk uruchamiający program w trybie debugowania w środowisku Microsoft Visual Studio.

Po uruchomieniu program zatrzyma się tuż przed wykonaniem linijki 9. W dolnej części środowiska możemy znaleźć kartę o nazwie `Locals`, w której wyświetlony jest obecny stan zmiennych zawartych w programie.

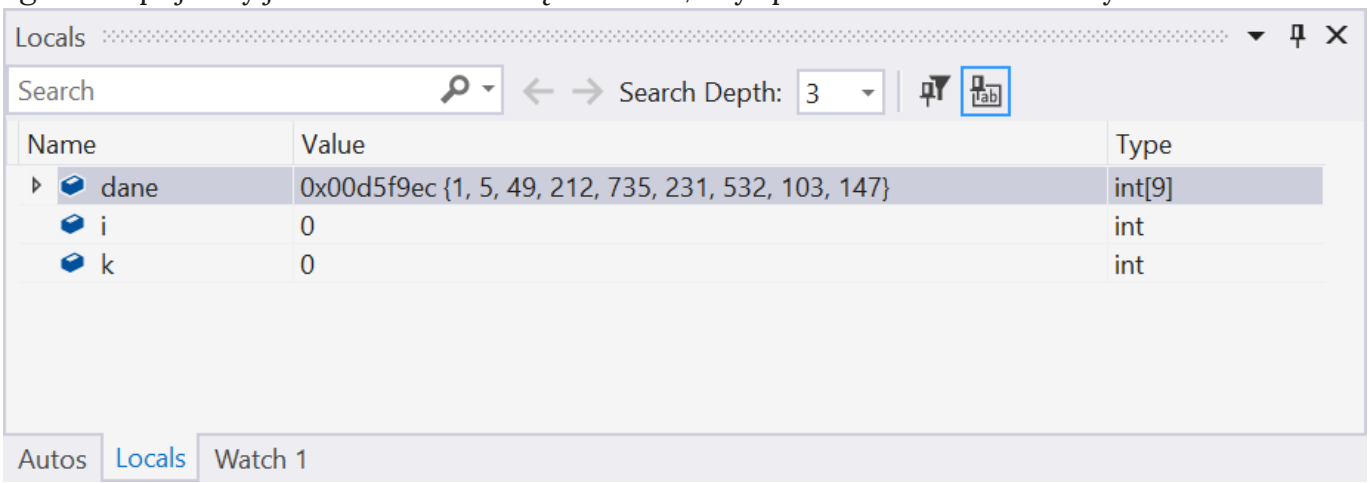


Karta z podglądem stanu zmiennych w środowisku Microsoft Visual Studio.

Na razie nie widzimy błędu. Spróbujmy zatem przejść krok dalej. Z górnego paska programu wybierzmy opcję `Step Over`. W tym celu możemy również nacisnąć klawisz `F10` na klawiaturze.

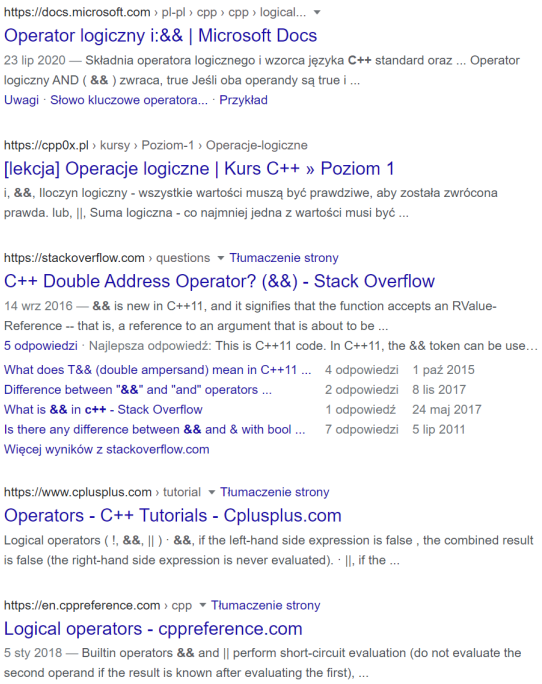
Opcja "Step Over" w środowisku Microsoft Visual Studio.

Po wybraniu opcji, debugger przeniósł nas od linijki 12. Oznacza to, że nie weszliśmy do instrukcji warunkowej, ponieważ warunek nie został spełniony. Na razie wszystko się zgadza. Spójrzmy jeszcze raz na kartę `Locals`, aby sprawdzić stan zmiennych.



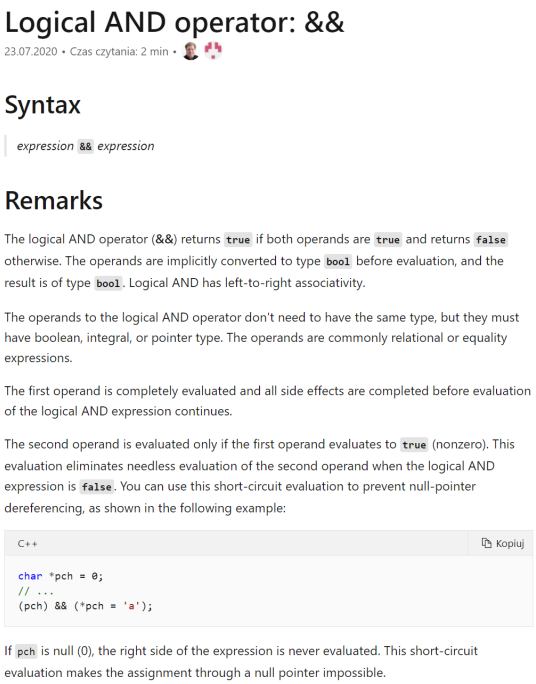
Widzimy, że wartości zmiennych nie zostały zmienione. Nie zgadza się to jednak z zamysłem autora kodu, gdyż w następnym wykonaniu pętla wykona się w dokładnie w ten sam sposób i zmienne pozostaną w tym samym stanie. Można zatem przypuszczać, że nie wykonała się instrukcja `i++` z linijki 9.

Spróbujmy poszukać przyczyny w wyszukiwarce Google. W tym wypadku wymyślenie odpowiedniego zapytania może być trudne, jednak zacznijmy od hasła „C++ &&”.



Wyniki wyszukiwania dla hasła "C++ &&" w wyszukiwarce Google z dnia pisania materiału.
Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Łatwo zauważyć, że trzecia pozycja raczej nie będzie stanowiła przydatnej informacji. Skupmy się zatem na pierwszym oraz piątym wyniku, ponieważ stanowią one dokumentację języka C++. W przypadku piątej pozycji odpowiedź możemy poznać już z poziomu wyszukiwarki Google. Pierwszy wynik to dokumentacja języka C++ według firmy Microsoft. Ponieważ strona została przetłumaczona na język polski z wykorzystaniem maszyny tłumaczącej, aby uniknąć błędów językowych, wyświetlmy ją w oryginalnym języku, tj. w angielskim. Dokonamy tego poprzez włączenie opcji „Przeczytaj w języku angielskim" w prawym górnym rogu.



Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

W czwartym akapicie znajduje się interesująca nas informacja: w przypadku operatora logicznego && drugi operand wykonywany jest tylko w momencie, gdy wartość pierwszego jest równa `true`. Skoro zatem nie jest spełniony warunek `dane[i] % 10 == 2`, nie zostanie sprawdzony warunek `dane[i++] % 7 == 0`, a co za tym idzie – nie zwiększy się wartość zmiennej `i`. Poprawny kod może wyglądać na przykład tak:

```cpp
#include <iostream>

using namespace std;

int main() {
    int dane[9] = { 1, 5, 49, 212, 735, 231, 532, 103, 147 };

    int i = 0, k = 0;

    while (i < 9) {
        if (dane[i] % 10 == 2 && dane[i] % 7 == 0) {
            k = k + 1;
        }
        i++;
    }

    cout << k << endl;
}
```

Java

Kod omawianego programu w języku Java wygląda następująco:

```java
public class Project {
    public static void main(String[] args) {
        int[] dane = {1, 5, 49, 212, 735, 231, 532, 103, 147};

        int i = 0, k = 0;

        while (i < 9) {
            if (dane[i] % 10 == 2 && dane[i++] % 7 == 0) {
                k = k + 1;
            }
        }
```

```
12
13          System.out.println(k);
14      }
15  }
```

Okazuje się, że program nie działa prawidłowo. Sprawdźmy za pomocą debuggera, w czym tkwi problem. Ustawmy breakpoint w linijce 8. W środowisku Eclipse możemy to osiągnąć przez wybranie opcji `Toogle breakpoint`, po kliknięciu prawym przyciskiem myszy na pasku przerwań przy interesującej nas linijce. Usunięcie breakpointa odbywa się w ten sam sposób, jednak należy wybrać opcję `Disable breakpoint`.

```
1  public class Project {
2      public static void main(String[] args) {
3          int[] dane = {1, 5, 49, 212, 735, 231, 532, 103, 147};
4
5          int i = 0, k = 0;
6
7          while (i < 9) {
8              if (dane[i] % 10 == 2 && dane[i++] % 7 == 0) {
9                  k = k + 1;
10             }
11         }
12
13         System.out.println(k);
14     }
15 }
```

Poprawnie utworzony breakpoint w środowisku Eclipse.

Następnie uruchomimy napisany program w trybie debugowania. W środowisku Eclipse możemy to osiągnąć za pomocą przycisku z ikoną robaka na górnym pasku narzędzi.

Ikona uruchamiająca program w trybie debugowania w środowisku Eclipse.

Po uruchomieniu program zatrzyma się przed wykonaniem sprawdzenia warunku instrukcji `if`. W prawej części środowiska wyświetli się karta z podglądem na stan zmiennych.

Variables | Breakpoints | Expressions

Name	Value
no method return value	
args	String[0] (id=20)
dane	(id=21)
[0]	1
[1]	5
[2]	49
[3]	212
[4]	735
[5]	231
[6]	532
[7]	103
[8]	147
i	0
k	0

Karta z podglądem stanu zmiennych w środowisku Eclipse.
Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

W tym momencie błąd nie jest jeszcze widoczny. Wybierzmy opcję `Step Over` z górnego paska narzędzi. Możemy ewentualnie nacisnąć klawisz `F6`.

Ikona funkcji "Step Over" w środowisku Eclipse.
Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

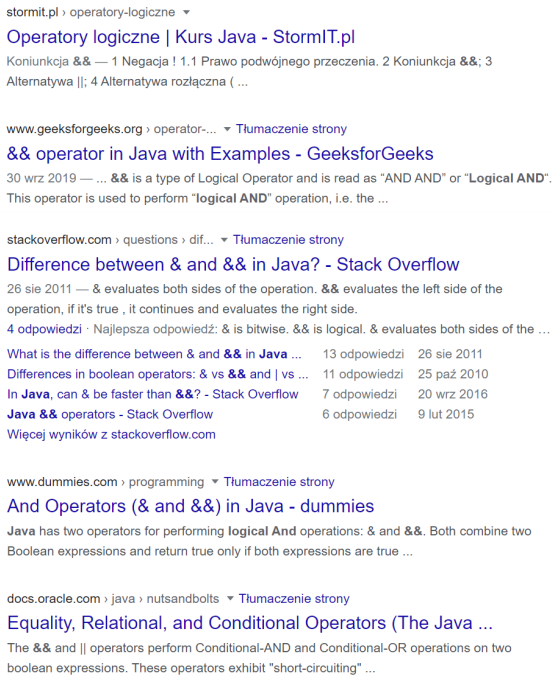
W tym momencie program wykona jedną instrukcję. Sprawdźmy aktualny stan zmiennych.

Variables | Breakpoints | Expressions

Name	Value
no method return value	
args	String[0] (id=20)
dane	(id=21)
[0]	1
[1]	5
[2]	49
[3]	212
[4]	735
[5]	231
[6]	532
[7]	103
[8]	147
i	0
k	0

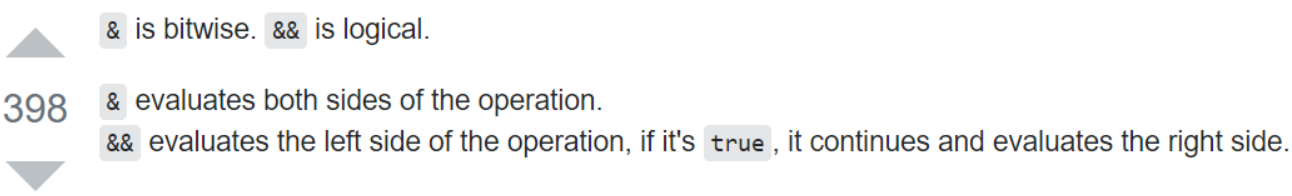
Karta z podglądem stanu zmiennych w środowisku Eclipse
Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Łatwo zauważyć, że nie została zwiększona wartość zmiennej `i`. Nie wykonała się zatem inkrementacja z drugiej części warunku. Kolejne próby wybrania opcji `Step Over` nadal nic nie zmieniają. Dlaczego? Odpowiedź na to pytanie spróbujmy znaleźć w wyszukiwarce Google. Możemy sprawdzić wyniki wyszukiwania dla hasła „Java &&”.



Wyniki wyszukiwania dla hasła "Java &&" w wyszukiwarce Google (dostęp: 30.03.2021)
Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Pierwsze dwie odpowiedzi nie wydają się zbyt pomocne. Zauważmy, że otrzymaliśmy wyniki wyszukiwania ze strony stackoverflow.com. Sprawdźmy zatem, czego możemy się dowiedzieć, wybierając trzeci wynik wyszukiwania. Rozwiązanie naszego problemu znajduje się w najlepszej odpowiedzi:



Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

W momencie, gdy lewa strona operatora && została uznana za `false`, prawa strona nie została wykonana. Z tego powodu nie zwiększyła się wartość zmiennej `i`, a w rezultacie pętla wykonywała się w nieskończoność. Poprawne rozwiązanie powinno być następujące:

```
1 public class Project {
2     public static void main(String[] args) {
3         int[] dane = {1, 5, 49, 212, 735, 231, 532, 103, 147};
4 
5         int i = 0, k = 0;
```

```
6
7           while (i < 9) {
8               if (dane[i] % 10 == 2 && dane[i] % 7 == 0) {
9                   k = k + 1;
10              }
11              i++;
12          }
13
14          System.out.println(k);
15      }
16  }
```

Python

Program napisany w języku Python wygląda tak:

```
1  dane = [1, 5, 49, 212, 735, 231, 532, 103, 147]
2
3  i = 0
4  k = 0
5
6  while i < 9:
7      if dane[i] % 10 == 2 and dane[i] % 7 == 0:
8          k = k + 1
9
10 print(k)
```

Aby odnaleźć błąd w przedstawionym programie, ustawmy breakpoint wewnątrz pętli w linijce 7. Możemy tego dokonać, klikając lewym przyciskiem myszy na pasku przerwań. Usunięcie już istniejącego breakpointa można uzyskać w ten sam sposób, co w przypadku jego dodania.

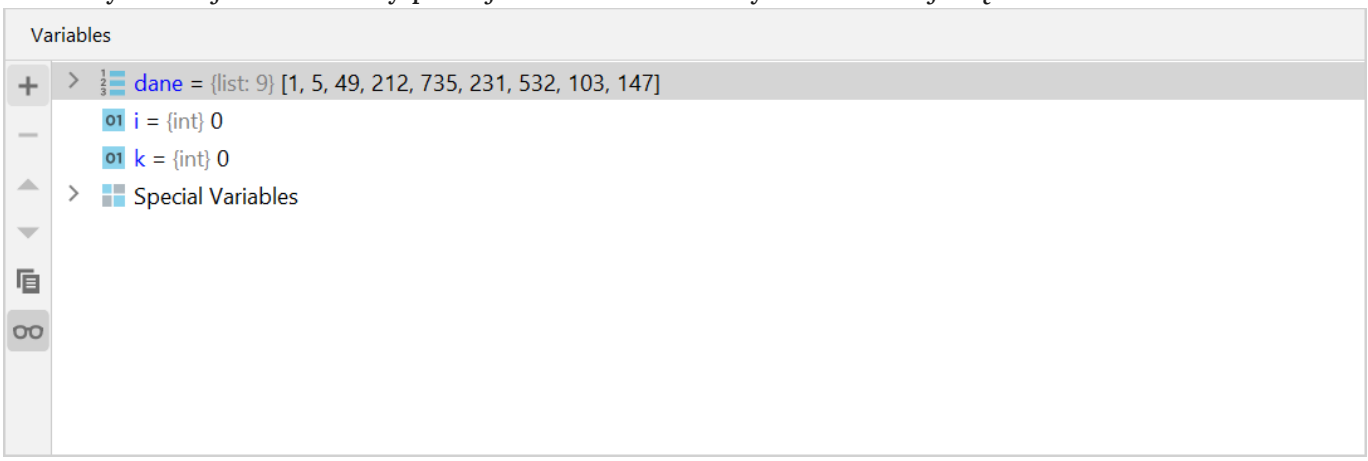
```
1   dane = [1, 5, 49, 212, 735, 231, 532, 103, 147]
2
3   i = 0
4   k = 0
5
6   while i < 9:
7       if dane[i] % 10 == 2 and dane[i] % 7 == 0:
8           k = k + 1
9
10  print(k)
```

Poprawnie ustawiony *breakpoint* w środowisku PyCharm.

Przejdźmy teraz do uruchomienia programu w trybie debugowania w środowisku PyCharm. W tym celu należy nacisnąć ikonę robaka w prawym górnym rogu lub kombinację klawiszy `Shift` + `F9` na klawiaturze.

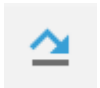
Przycisk uruchamiający program w trybie debugowania w środowisku PyCharm.

Już w tym miejscu możemy podejrzeć stan zmiennych w dolnej części środowiska.



Karta z podglądem stanu zmiennych w środowisku PyCharm.

Jeżeli nie widzimy błędu, możemy wybrać opcję `Step Over`, która znajduje się w karcie `Debug` w dolnej części programu. Ewentualnie możemy nacisnąć klawisz `F8`.



Ikona "Step Over" w środowisku PyCharm.

Łatwo zauważyć, że pomimo wielokrotnego wyboru opcji `Step Over`, nie zmienia się stan zmiennej `i`. Jeżeli zmienna się nie zmienia, to program nie wyjdzie z pętli, ponieważ tylko zmienna `i` jest sprawdzana w warunku pętli. Może nam się nasunąć prosty wniosek, tj. programista zapomniał o zwiększaniu wartości zmiennej `i` wewnątrz pętli. Poprawny kod wygląda następująco:

```
dane = [1, 5, 49, 212, 735, 231, 532, 103, 147]

i = 0
k = 0

while i < 9:
    if dane[i] % 10 == 2 and dane[i] % 7 == 0:
        k = k + 1
    i = i + 1

print(k)
```

Słownik

breakpoint

punkt wstrzymania lub pułapka – miejsce docelowego przerwania działania programu w celu sprawdzenia jego stanu w danym momencie; program uruchomiony pod kontrolą debuggera przerwie działanie w danym miejscu

debugger

służy do szybkiego znajdowania błędów w kodzie programu i ich eliminowania

IDE

(ang. *Integrated Development Environment* - zintegrowane środowisko programistyczne) zbiór programów, służących do tworzenia, modyfikowania oraz utrzymywania oprogramowania

Infografika

Polecenie 1

Zapoznaj się z infografiką. Jakie znasz metody radzenia sobie z błędami w kodzie?

Polecenie 2

Poszukaj w dostępnych źródłach informacji na temat metody gumowej kaczuszki. Jak oceniasz jej przydatność?

Dokumentacja

Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Polecenie 3

Opracuj notatkę podsumowującą najważniejsze informacje przedstawione w infografice.

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Wskaż wszystkie poprawne odpowiedzi. Zaznacz, które z zadań mogą zostać wykonane przez debugger.

- [ ] optymalizacja czasowa kodu
- [ ] zmienienie kodu w celu naprawy popełnionych przez programistę błędów
- [ ] przegląd zmiennych w danym czasie
- [ ] wychwycenie momentu wstrzymania oraz podanie stanu programu w tymże momencie

Ćwiczenie 2

Uzupełnij zapytanie skierowane do wyszukiwarki Google tak, aby wyniki pochodziły tylko ze strony stackoverflow.com.

c++ segmentation fault [ ] stackoverflow.com

Ćwiczenie 3

Wskaż, ile razy droższe, zgodnie z zasadą 1-10-100, jest naprawienie błędu w oprogramowaniu dostarczonym do klienta od naprawienia tego samego błędu na etapie tworzenia oprogramowania. Zaznacz wszystkie poprawne odpowiedzi.

- [ ] 10
- [ ] 100
- [ ] 1
- [ ] 0

Ćwiczenie 4

Wskaż, jakim błędem może zakończyć się uruchomienie nieskończonej rekurencji.

- ◯ błąd wywołania funkcji (*function call error*)
- ◯ błąd ładowania strony (*page fault error*)
- ◯ brak poprawnej odpowiedzi
- ◯ przepełnienie stosu (*stack overflow*)

Ćwiczenie 5

Wskaż, które z narzędzi może pomóc znaleźć błędy semantyczne lub logiczne programu.

- ◯ dokumentacja zapisana na dysku
- ◯ debugger
- ◯ forum stackoverflow.com
- ◯ wyszukiwarka internetowa

Ćwiczenie 6

Przeanalizuj kod i wskaż błąd.

```cpp
#include <iostream>
#include <cmath>

using namespace std;

int main() {
    int k = 13;
    int i = 2;

    for (int i = 2; i <= sqrt(k); ++i) {
        if (k % i != 0) {
            i = k;
            break;
        }
    }

    if (i == k) {
        cout << k << " jest liczba pierwsza" << endl;
    }
    else {
        cout << k << " nie jest liczba pierwsza" << endl;
    }
}
```

- ◯ niewłaściwa inkrementacja zmiennej `i`
- ◯ przesłanienie nazwy – ponowna deklaracja zmiennej `i`
- ◯ niewłaściwe sprawdzenie podzielności zmiennej `k` przez zmienną `i`
- ◯ odwrotnie skonstruowany warunek instrukcji warunkowej odpowiedzialnej za wypisanie wyniku

Ćwiczenie 7

Przeanalizuj kod i wskaż błąd.

```
1  public class Project {
2      public static void main(String[] args) {
3          int k = 15;
4          int i = 2;
5
6          for (; i <= Math.sqrt(k); ++i) {
7              if (k / i != 0) {
8                  i = k;
9                  break;
10             }
11         }
12
13         if (i == k) {
14             System.out.println(k + " jest liczba pierwsza");
15         }
16         else {
17             System.out.println(k + " nie jest liczba pierwsza")
18         }
19     }
20 }
```

- ◯ błędny warunek kończący pętlę `for`
- ◯ niepoprawne wykorzystanie instrukcji `break`
- ◯ brak instrukcji początkowej pętli `for`
- ◯ niepoprawne sprawdzanie podzielności zmiennej `k` przez zmienną `i`

Ćwiczenie 8

Przeanalizuj kod i wskaż błąd.

```
import math

k = 13
i = 2

while i <= math.sqrt(k):
    if k % i != 0:
        i = k
        break
    i += 1

if i != k:
    print(str(k) + " jest liczba pierwsza")
else:
    print(str(k) + " nie jest liczba pierwsza")
```

- ◯ niepoprawne wcięcie kodu
- ◯ odwrotnie sformułowany warunek wypisujący wynik
- ◯ błędnie sformułowany warunek podzielności
- ◯ niewłaściwa inkrementacja zmiennej `i`

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Jak rozwiązywać problemy programistyczne?

Grupa docelowa:

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

Podstawa programowa:

Cele kształcenia – wymagania ogólne

> 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

> II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.
>
> Zakres podstawowy. Uczeń:
>
> 4) wyszukuje w sieci potrzebne informacje i zasoby, ocenia ich przydatność oraz wykorzystuje w rozwiązywanych problemach.

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):

- Przedstawisz sposób przeszukiwania dowolnej strony internetowej za pomocą wybranej wyszukiwarki.
- Zapoznasz się z przykładowymi narzędziami, które wspierają pracę programisty.
- Rozwiążesz problemy informatyczne w kodzie za pomocą debuggera.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- metody aktywizujące.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

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;
- oprogramowanie dla języka Java SE 8 (lub nowszej wersji), w tym Eclipse 4.4 (lub nowszej wersji);
- oprogramowanie dla języka Python 3 (lub nowszej wersji), w tym PyCharm lub IDLE.

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Jak rozwiązywać problemy programistyczne?". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Wyświetlenie przez nauczyciela tematu i celów zajęć, przejście do wspólnego ustalenia kryteriów 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 „Infografika". Uczniowie wspólnie zapoznają się z jej treścią. Zapisują ewentualne pytania. Po czym następuje dyskusja o zasadzie 1-10-100 oraz o tym jak unikać błędów w kodzie i jak naprawiać wykryte usterki.
2. **Ćwiczenie umiejętności.** Nauczyciel przechodzi do sekcji „Sprawdź się". Zapowiada uczniom, że w kolejnym kroku będą rozwiązywać ćwiczenia nr 1-8 – od najprostszych do najtrudniejszych – i będą to robić wspólnie. Wybrany uczestnik zajęć czyta po kolei polecenia. Po każdym przeczytanym poleceniu ochotnik udziela odpowiedzi. Reszta uczniów ustosunkowuje się do niej, proponując swoje pomysły. Nauczyciel w razie potrzeby koryguje odpowiedzi, dopowiada istotne informacje, udziela uczniom informacji zwrotnej.

Faza podsumowująca:

1. Nauczyciel ponownie wyświetla na tablicy temat i cele lekcji zawarte w sekcji „Wprowadzenie". W kontekście ich realizacji następuje omówienie ewentualnych problemów z rozwiązaniem ćwiczeń z sekcji „Sprawdź się".

Praca domowa:

1. Uczniowie wykonują ćwiczenia 9-11 z sekcji „Sprawdź się".
2. Uczniowie wykonują polecenie 2 z sekcji „Infografika".

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla języka C++.
- Oficjalna dokumentacja techniczna dla kompilatora GCC/G++ 4.5 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio.
- Oficjalna dokumentacja techniczna dla języka Java SE 8 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania Eclipse 4.4 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla języka Python 3 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania PyCharm lub IDLE.

Wskazówki metodyczne:

- Treści w sekcji „Infografika" można wykorzystać na lekcji jako podsumowanie i utrwalenie wiedzy uczniów.