II Liceum Ogólnokształcące im. Mikołaja Kopernika w Lesznie
z Oddziałami Dwujęzycznymi i Międzynarodowymi
ul. Prusa 33, 64-100 Leszno

„Jeśli nie potrafisz wytłumaczyć czegoś w prosty sposób, to znaczy, że tak naprawdę tego nie rozumiesz”

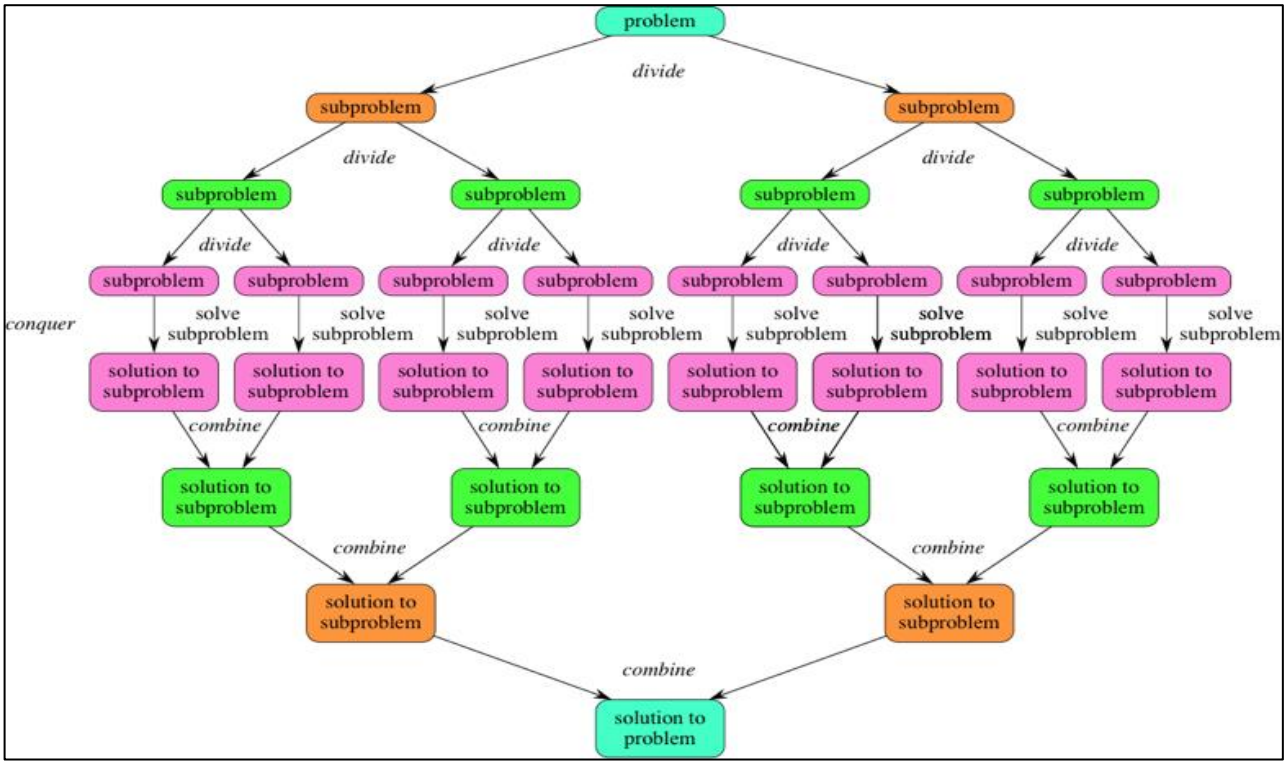
Praca przygotowana przez
Klaudię Błażyczek i Patrycję Klein
pod kierunkiem
Dominika Siecińskiego

Leszno, 10.10.2018

„Jeśli nie potrafisz wytłumaczyć czegoś w prosty sposób, to znaczy, że tak naprawdę tego nie rozumiesz”

Pewien znany na całym świecie fizyk powiedział kiedyś - *Jeśli nie potrafisz wytłumaczyć czegoś w prosty sposób, to znaczy, że tak naprawdę tego nie rozumiesz.* – autorem tego cytatu jest oczywiście Albert Einstein. Według niego, prawdziwe zrozumienie jakiegoś zagadnienia występuje wtedy, gdy człowiek jest w stanie wytłumaczyć dany problem w prosty i przejrzysty sposób. Twierdzenie to wydaje się być uzasadnione, jednak czy jest tak naprawdę? Sprawdzimy to na podstawie algorytmu jednoczesnego szukania minimum i maksimum w jakimś zbiorze. Zostanie on również zaprezentowany na lekcji informatyki w szkole.

Algorytm szuka jednocześnie liczby najmniejszej i największej ze zbioru przynajmniej dwuelementowego, wykorzystując do tego celu metodę dziel i zwyciężaj, która znacznie usprawnia szybkość działania. Zasada funkcjonowania jest następująca: za każdym razem bierzemy dwie kolejne liczby i porównujemy je ze sobą, następnie mniejsza jest przydzielana do zbioru teoretycznych liczb minimalnych, a większa z pary, do zbioru teoretycznych liczb maksymalnych. Dzięki temu zabiegowi mamy teraz dwa podzbiory. Czyli główny problem znalezienia dwóch liczb jednocześnie został rozdzielony na dwa pomniejsze problemy, by następnie ich wyniki połączyć w jedną całość, czyli wynik końcowy. To właśnie jest metoda dziel i zwyciężaj, której schemat, dla uproszczenia, można by pokazać w następujący sposób.



Rysunek nr 1
Przedstawia schemat metody dziel i zwyciężaj

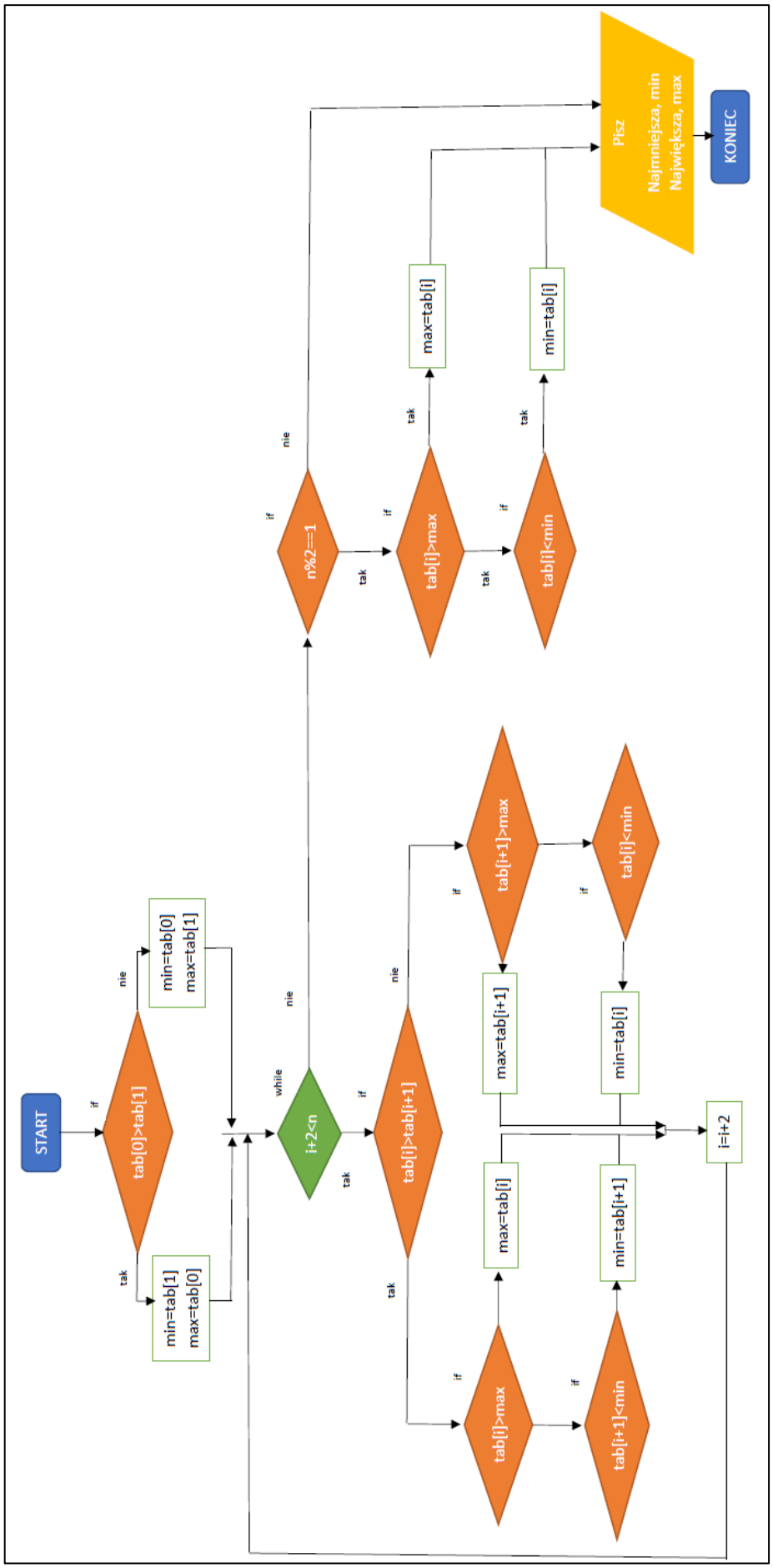
W stworzonych w ten sposób dwóch podzbiorach wystarczy poszukać liczby minimalnej i maksymalnej w sposób liniowy, by złożyć dwa wyniki w jeden, końcowy. Stworzony przez nas algorytm działa jednak w trochę inny sposób. Zamiast tworzyć nowe dwa zbiory, liczby szukane są od razu podczas wczytywania danych. Poniżej przedstawiamy pseudokod programu, który ułatwi zrozumienie jego działania.

„Jeśli nie potrafisz wytłumaczyć czegoś w prosty sposób, to znaczy, że tak naprawdę tego nie rozumiesz”

```
begin

 for a=0 to n-1 step 2 do
   x := RandInt(0, 10000)
   tablica[a] := x
 endfor

 if tablica[0] > tablica[1] then
   MIN = tablica[1]
   MAX = tablica[0]
 else
   MIN = tablica[0]
   MAX = tablica[1]

 while i+2 < n
   if tablica[i] > tablica[i+1] then
     if tablica[i] > MAX then
       MAX = tablica[i]
     if tablica[i+1] < MIN then
       MIN = tablica[i+1]
   else
     if tablica[i+1] > MAX then
       MAX = tablica[i+1]
     if tablica[i] < MIN then
       MIN = tablica[i]
   i = i+2
 endwhile

 if n % mod == 1
   if tablica[i] > MAX
     MAX = tablica[i]
   if tablica[i] < MIN
     MIN = tablica[i]

  print line "Najmniejsza: " and MIN
  print line "Najwieksza: " and MAX

end
```

Rysunek nr 2
Przedstawia pseudokod programu

Program jest też szybszy niż zwykłe wyszukiwanie liniowe. Przykładowo wyszukanie liczby minimalnej i maksymalnej ze zbioru składającego się ze 100 elementów w przypadku drugiego sposobu wyszukiwania, będzie złożone z 198 powtórzeń pętli if. 99 dla minimum i 99 dla maksimum. Naszym sposobem, w przypadku zbioru tej samej wielkości, tych powtórzeń będzie 148. Potwierdzenie tych danych można znaleźć na stronie z algorytmami maturalnymi, gdzie wzór na ilość powtórzeń w wyszukiwaniu liniowym prezentuje się tak: $\underline{2(n-1)}^{(1)\cdot}$, gdzie n to liczba elementów w zbiorze. Liczbę wykonanych powtórzeń z prezentowanego przez nas programu można wyliczyć ze wzoru: $\underline{(n/2) + 2(n/2-1)}^{(1)\cdot}$, czyli: $\underline{(3n-4)/2}^{(1)\cdot}$. Wzór ten wymaga wyjaśnienia. Najpierw dzielimy n, czyli 100 na pół ,czyli dwa podzbiory (metoda dziel i zwyciężaj), następnie dodajemy do tego podwojoną ilość powtórzeń zrobionych podczas szukania minimum oraz maksimum w utworzonych podzbiorach. Warto wspomnieć, że wzór na liczbę wykonanych powtórzeń w przypadku zbioru o nieparzystej liczbie elementów prezentuje się trochę inaczej: $\underline{(n-1)/2 + 2((n-1)/2-1) + 2}^{(1)\cdot}$. Po uproszczeniu: $\underline{(3n-3)/2}^{(1)\cdot}$. Różnice wynikają z tego, że ostatnia liczba w zbiorze jest porównywana do bieżącej liczby minimalnej i maksymalnej.



Rysunek nr 3

Przedstawia schemat blokowy programu

„Jeśli nie potrafisz wytłumaczyć czegoś w prosty sposób, to znaczy, że tak naprawdę tego nie rozumiesz”

Wyżej przedstawiony schemat, w sposób obrazowy pokazuje w jaki sposób działa program, dzięki niemu można o wiele łatwiej wyobrazić sobie jego działanie oraz sprawdzić ilość użytych pętli if. Rysunek nie przedstawia jednak tworzenia tablic oraz użytych zmiennych.

W algorytmie przyjmujemy, że tablica (zbiór) składa się z minimum dwóch elementów, a ich wartości są losowane przez komputer, co pokażemy na przykładach gotowych, działających programów w czterech językach programowania. W kodzie pojawia się także zmienna „i”, która jest numerem liczby w tablicy. Zwiększającej się ona o dwa w każdym kolejnym powtórzeniu pętli while, dopóki „i” nie jest większa od ilości liczb w tablicy n-elementowej. Następnie w przypadku tablicy o nieparzystej liczbie elementów, sprawdza ostatnią z nich i podaje wynik końcowy, jeżeli tablica składa się z parzystej ilości składników, od razu podaje efekt końcowy swojej pracy.

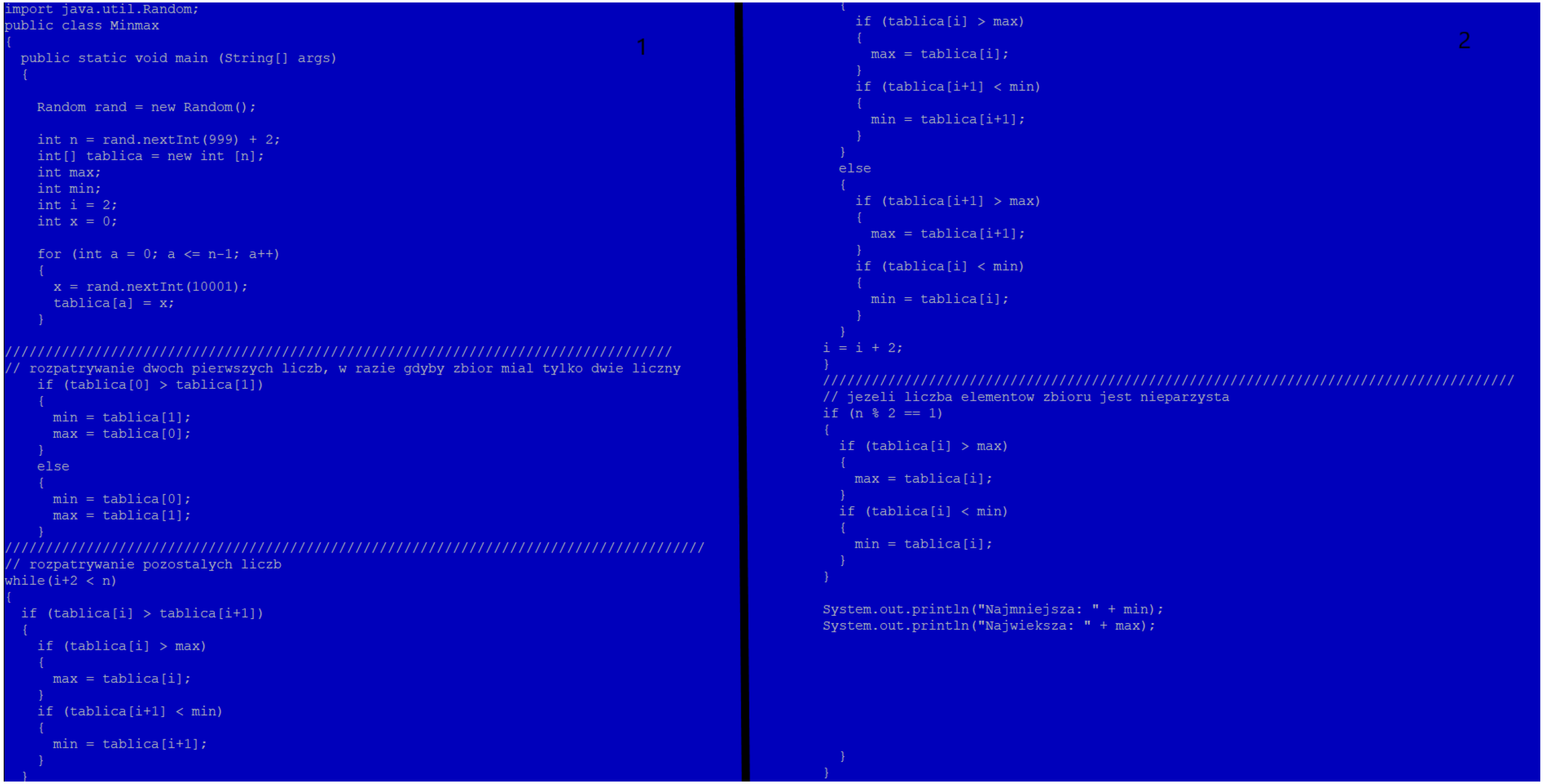
Poniżej przedstawimy także kody programów, które napisane zostały w czterech popularnych językach programowania, a mianowicie: *swift, java, python, C++*.

```
import random

n = random.randint(2, 1000)
tablica = []
i = 2
x = 0

for a in range(n):
  x = random.randint(0, 10000)
  tablica.append(x)

if tablica[0] > tablica[1]:
  min = tablica[1]
  max = tablica[0]
else:
  min = tablica[0]
  max = tablica[1]

while i+2 < n:
 if tablica[i] > tablica[i+1]:
     if tablica[i] > max:
         max = tablica[i]
     elif tablica[i+1] < min:
         min = tablica[i+1]
 else:
     if tablica[i+1] > max:
         max = tablica[i+1]
     elif tablica[i]< min:
         min = tablica[i]
 i = i+2

if n % 2 == 1:
    if tablica[i] > max:
        max = tablica[i]
    elif tablica[i] < min:
        min = tablica[i]

print("Najmniejsza:", min)
print("Najwieksza:", max)
```

Zdjęcie nr 4

Przedstawia kod w pythonie

„Jeśli nie potrafisz wytłumaczyć czegoś w prosty sposób, to znaczy, że tak naprawdę tego nie rozumiesz”



```java
import java.util.Random;
public class Minmax
{
  public static void main (String[] args)
  {

    Random rand = new Random();

    int n = rand.nextInt(999) + 2;
    int[] tablica = new int [n];
    int max;
    int min;
    int i = 2;
    int x = 0;

    for (int a = 0; a <= n-1; a++)
    {
      x = rand.nextInt(10001);
      tablica[a] = x;
    }

/////////////////////////////////////////////////////////////////////////////////
// rozpatrywanie dwoch pierwszych liczb, w razie gdyby zbior mial tylko dwie liczny
    if (tablica[0] > tablica[1])
    {
      min = tablica[1];
      max = tablica[0];
    }
    else
    {
      min = tablica[0];
      max = tablica[1];
    }
/////////////////////////////////////////////////////////////////////////////////////
// rozpatrywanie pozostalych liczb
while(i+2 < n)
{
  if (tablica[i] > tablica[i+1])
  {
    if (tablica[i] > max)
    {
      max = tablica[i];
    }
    if (tablica[i+1] < min)
    {
      min = tablica[i+1];
    }
  }
  else
  {
    if (tablica[i+1] > max)
    {
      max = tablica[i+1];
    }
    if (tablica[i] < min)
    {
      min = tablica[i];
    }
  }
i = i + 2;
}
/////////////////////////////////////////////////////////////////////////////////////
// jezeli liczba elementow zbioru jest nieparzysta
if (n % 2 == 1)
{
  if (tablica[i] > max)
  {
    max = tablica[i];
  }
  if (tablica[i] < min)
  {
    min = tablica[i];
  }
}

System.out.println("Najmniejsza: " + min);
System.out.println("Najwieksza: " + max);

  }
}
```

Zdjęcie nr 5

Przedstawia kod w javie

„Jeśli nie potrafisz wytłumaczyć czegoś w prosty sposób, to znaczy, że tak naprawdę tego nie rozumiesz”

```
import Foundation

#if os(Linux)
    srandom(UInt32(time(nil)))
#endif

func losowa(_ min: Int, _ max: Int) -> Int
{
    #if os(Linux)
        return Int(random() % max) + min
    #else
        return Int(arc4random_uniform(UInt32(max)) + UInt32(min))
    #endif
}


let n = losowa(2,1000)              // wielkosc tablicy
var tablica = [Int]()
var max = Int()
var min = Int()
var i = 2

for _ in 0...n - 1
{
  tablica.append(losowa(0,10000))
}

////////////////////////////////////////////////////////////////////////////////
//rozpatrzenie dwoch pierwszych liczb, w razie gdyby zbior mial tylko dwie liczby

if (tablica[0] > tablica[1])
{
  min = tablica[1]
  max = tablica[0]
}
else
{
  min = tablica[0]
  max = tablica[1]
}
////////////////////////////////////////////////////////////////////////////////
//rozpatrywanie pozostalych liczb
while(i+2 < n)
{
  if (tablica[i] > tablica[i+1])
  {
    if (tablica[i] > max)
    {
      max = tablica[i]
    }
    if (tablica[i+1] < min)
    {
      min = tablica[i+1]
    }
  }
  else
  {
    if (tablica[i+1] > max)
    {
      max = tablica[i+1]
    }
    if (tablica[i] < min)
    {
      min = tablica[i]
    }
  }
  i = i + 2
}
////////////////////////////////////////////////////////////////////////////////
//jezeli liczba elementow zbioru jest nieparzysta
if(n % 2 == 1)
{
  if (tablica[i] > max)
  {
    max = tablica[i]
  }
  if (tablica[i] < min)
  {
    min = tablica[i]
  }
}
////////////////////////////////////////////////////////////////////////////////
print("liczba minimalna: \(min) ")
print("liczba maksymalna: \(max) ")
```

Zdjęcie nr 6

Przedstawia kod w swifcie

„Jeśli nie potrafisz wytłumaczyć czegoś w prosty sposób, to znaczy, że tak naprawdę tego nie rozumiesz”

Zdjęcie nr 7

Przedstawia kod w C++

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

int main()
{
   srand( time(NULL) );
   int n = (rand() % 999) + 2;
   int tablica[n];  MAX;
   int MIN;
   int i = 2;
   int x = 0;

   for(int a = 0; a <= n-1; a++)
   {
      x = rand() % 10001;
      tablica[a] = x;
   }

   /* Rozpatrywanie dwoch pierwszych liczb
      w razie gdyby zbior mial tylko dwie liczby */
   if (tablica[0] > tablica[1])
   {
      MIN = tablica[1];
      MAX = tablica[0];
   }
   else
   {
      MIN = tablica[0];
      MAX = tablica[1];
   }

   // Rozpatrywanie pozostalych liczb
   while(i+2 < n)
   {
      if(tablica[i] > tablica[i+1])
      {
         if (tablica[i] > MAX)
         {
            MAX = tablica[i];
         }
         if (tablica[i+1] < MIN)
         {
            MIN = tablica[i+1];
```

1

```
      }
      else
      {
         if (tablica[i+1] > MAX)
         {
            MAX = tablica[i+1];
         }
         if (tablica[i] < MIN)
         {
            MIN = tablica[i];
         }
      }
      i = i + 2;
   }

   // Jezeli liczba elementow zbioru jest nieparzysta
   if (n % 2 == 1)
   {
      if (tablica[i] > MAX)
      {
         MAX = tablica[i];
      }
      if (tablica[i] < MIN)
      {
         MIN = tablica[i];
      }
   }

   cout << "Najmniejsza: " << MIN << endl;
   cout << "Najwieksza: " << MAX << endl;
   return 0;
}
```

2

„Jeśli nie potrafisz wytłumaczyć czegoś w prosty sposób, to znaczy, że tak naprawdę tego nie rozumiesz”

Algorytm został przedstawiony na lekcji informatyki w szkole, a po zakończonym wykładzie, przeprowadzono test sprawdzający stopień zrozumienia tematu przez uczniów. Składał się on kolejno z dwóch pytań otwartych i czterech zamkniętych. Jego wyniki prezentują się następująco.

	P1	P2	P3	P4	P5	P6	Pkt. ogółem	Cel max.	Wynik testu
U1	0	1	0	1	1	1	4	6	67%
U2	0	1	1	1	1	1	5	6	83%
U3	0	1	0	1	1	1	4	6	67%
U4	0	1	1	1	1	1	5	6	83%
U5	1	1	1	1	1	1	6	6	100%
U6	1	0	1	1	1	1	5	6	83%
Suma	2	5	4	6	6	6	29	36	81%
% poprawnych odpowiedzi	33%	83%	67%	100%	100%	100%			

Tabela nr 8

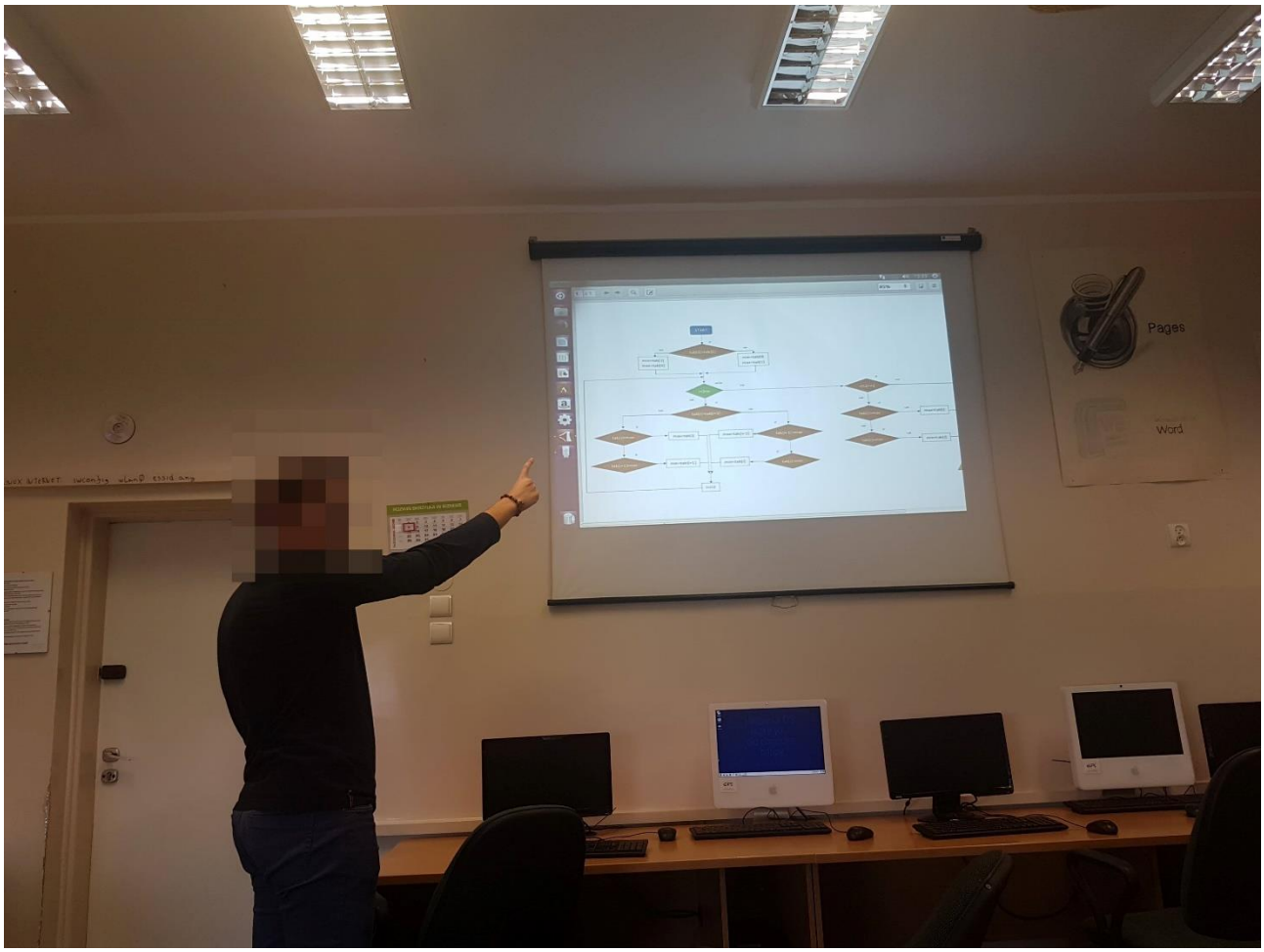
Przedstawia wyniki uczniów z testu

P1-P6 - oznaczają kolejno numery pytań, U1- U6 – uczniów. Każdy z nich mógł zdobyć maksymalnie 6 punktów. Tabela prezentuje ile punktów i procent z egzaminu dostał każdy egzaminowany oraz jaki procent uczniów zdołało dobrze odpowiedzieć na każde z pytań. Podsumowuje także sumę wszystkich punktów całej klasy oraz procent z całego egzaminu wszystkich obecnych.

Zdjęcie nr 9

Przedstawia uczniów na wykładzie

„Jeśli nie potrafisz wytłumaczyć czegoś w prosty sposób, to znaczy, że tak naprawdę tego nie rozumiesz”



Zdjęcie nr 10
Przedstawia ucznia na wykładzie

„Jeśli nie potrafisz wytłumaczyć czegoś w prosty sposób, to znaczy, że tak naprawdę tego nie rozumiesz”

Nasze wnioski po ukończeniu projektu są następujące: Po przejrzystym i prostym przedstawieniu przez nas metody wyszukiwania najmniejszej i największej wartości w zbiorze co najmniej dwuelementowym i przeprowadzeniu lekcji dla klasy trzeciej wykonałyśmy test, po przeanalizowaniu którego, wyniki pokazały nam, że uczestnicy zajęć zrozumieli materiał łącznie w 81%. Po odrzuceniu wyników najlepszego i najgorszego z uczniów, w celu poprawnego statystycznego przeanalizowania danych, okazało się, że egzaminowani zaliczyli sprawdzian na 79%. Według nas jest to wysokoprocentowy wynik, który równa się z dobrym zaznajomieniem się z metodą. Uczniowie prawie bezbłędnie rozwiązali zadanie, które miało sprawdzić znajomość samej logiki działania, czyli na napisaniu na jakie podzbiory algorytm podzieliłby dany zbiór liczb, co ukazało nam, że osoby wiedzą w jaki sposób działa kod. Niestety, pytanie o najmniejszej ilości poprawnych odpowiedzi dotyczyło rozpoznania do jakiej metody programistycznej należy przedstawiony przez nas temat, którą było dziel i zwyciężaj. Mogło to wynikać z nieuwagi uczniów w zapamiętywaniu pozornie małoważnych terminów. Nie przeszkodziło to jednak w poprawnym zrozumieniu działania metody. Natomiast pytania dotyczące uzupełnienia kodu i stwierdzenia czy przedstawiona przez nas metoda jest szybsza od zwyczajnej metody wyszukiwania, odznaczyły się 100% poprawnością odpowiedzi. Oznacza to, że po wspólnych zajęciach uczestnicy wiedzą na czym polega dany sposób wyszukiwania minimalnej i maksymalnej wartości w danym zbiorze w językach: *java, swift, cpp* i *python* oraz rozumieją sposób w jaki działa podany program. Wiedzą również dlaczego przedstawiona przez nas metoda jest szybsza od klasycznego wyszukiwania liniowego oraz w jaki sposób mogą ją użyć, aby mogła zadziałać. Umożliwia im to w przyszłości wykorzystanie tej metody w celu zoptymalizowania swojego programu.

Podczas prowadzenia zajęć grupa wykazała duże zainteresowanie podanym tematem, co uznaliśmy za jedną z oznak rozumowania materiału. Grupa była dociekliwa, zadawała nam pytania dotyczące działania kodów oraz całości metody. Ich szczególną uwagę przykuł schemat blokowy, dzięki któremu krok po kroku zrozumieli działanie programu. Schemat ukazał im w przejrzysty sposób jak po kolei przebiega wybieranie minimalnej i maksymalnej wartości. Uważamy, że przekazywanie nauki poprzez tą metodę pomaga w prosty sposób przekazać innym swoją myśl, co na pewno pomogło w następnych etapach w wytłumaczeniu poszczególnych wątków naszych kodów.

Z naszej perspektywy możemy dodać, że rzetelne przygotowanie się do wykładu i całego projektu, wymagało dokładnego zagłębienia się w sposób działania programu, by następnie w przejrzysty sposób przedstawić go na zajęciach. Sami uczniowie deklarowali, że rozumieją zasadę działania kodu, co daje kolejny powód, by twierdzić, że przemyślenia Alberta Einsteina są prawdziwe.

„Jeśli nie potrafisz wytłumaczyć czegoś w prosty sposób, to znaczy, że tak naprawdę tego nie rozumiesz”

Uważamy, że po odbytej lekcji i jej analizie, możemy potwierdzić tezę naukowca Alberta Einsteina: "*Jeśli nie potrafisz wytłumaczyć czegoś w prosty sposób, to znaczy, że tak naprawdę tego nie rozumiesz",* ponieważ w prosty i logiczny sposób, przekazałyśmy wspólnie zdobytą przez nas wiedzę pozostałym osobą z naszej grupy informatycznej, co oznacza również pełne zrozumienie algorytmu z naszej strony, zgodnie z założeniem Einsteina. Po odbytych zajęciach nie zostawiliśmy żadnej osoby, która nie potrafiłaby aktualnie równie dobrze jak my przekazać dalej zdobytą wiedzę, którą zdobyła poprzez nasze logiczne i proste wytłumaczenie działania metody. Mimo złożonego algorytmu i skomplikowanej składni programowania, z którą nie każdy może być zaznajomiony i udało się w przejrzysty sposób wytłumaczyć działanie i mechanikę programu. Dzięki temu udało nam się potwierdzić założone na początku postanowienia i zrealizować cel zadania, co oznacza pełne zrozumienie metody dziel i zwyciężaj.

„Jeśli nie potrafisz wytłumaczyć czegoś w prosty sposób, to znaczy, że tak naprawdę tego nie rozumiesz”

Nasza propozycja kolejnego tematu na projekt na stronie Code Kopernik: „ Zamiana liczb z systemu dziesiętnego na binarny”.

Zgadzamy się na udostępnienie mojego projektu na stronie www.code.kopernik-leszno.pl, aby kolejni uczniowie mogli skorzystać z tego materiału przy nauce programowania

„Jeśli nie potrafisz wytłumaczyć czegoś w prosty sposób, to znaczy, że tak naprawdę tego nie rozumiesz”

Źródła

1. Informacja ze strony www.algorytm.edu.pl
   pobrana z dnia 14.10.2018

Spis zdjęć, rysunków, tabel i wykresów