

Mikrobitowcy

Autorzy: Dawid Osuchowski, Filip Klębczyk

Lekcja 8: Micro:bit - listy

W ramach zajęć uczniowie zapoznają się z kontenerowym typem danych - listą. Zgłębią zasady poprawnego odwoływania się do elementów listy poprzez indeksowanie. Zdobytą wiedzę wykorzystają do stworzenia programu - przewijanej listy obrazków sterowanej przyciskami.

Cele lekcji:

Uczeń powinien:

- Znać pojęcia: lista, indeks, inkrementacja, dekrementacja
- Wiedzieć, jakie zastosowanie w języku Python ma lista
- Umieć definiować listę i odwoływać się do jej elementów
- Znać zasady poprawnego indeksowania, określania długości listy i wiedzieć, jaki indeks mają element pierwszy i ostatni,
- Potrafić stworzyć zmienną przechowującą wartości liczbowe.

Materiały pomocnicze:

- <http://microbit.org/>
- <https://bbcmicrobitmicropython.readthedocs.io/>
- <https://github.com/MicrobitPolska/FunWithMicrobit>

Pojęcia kluczowe:

→ lista → indeks → inkrementacja → dekrementacja

Czas realizacji: 45 min.

Metody pracy:

- wykład,
- dyskusja,
- ćwiczenia praktyczne przy komputerze,
- szukanie rozwiązań na postawiony problem
- prezentowanie efektów pracy.

Treści programowe:

Podstawa programowa kształcenia ogólnego dla szkół podstawowych – II etap edukacyjny – klasy VII-VIII, informatyka:

I. Rozumienie, analizowanie i rozwiązywanie problemów. Uczeń:

- 1) formułuje problem w postaci specyfikacji (czyli opisuje dane i wyniki) i wyróżnia kroki w algorytmicznym rozwiązywaniu problemów. Stosuje różne sposoby przedstawiania algorytmów, w tym w języku naturalnym, w postaci schematów blokowych, listy kroków;

3) przedstawia sposoby reprezentowania w komputerze wartości logicznych, liczb naturalnych (system binarny), znaków (kody ASCII) i tekstów;

4) rozwija znajomość algorytmów i wykonuje eksperymenty z algorytmami, korzystając z pomocy dydaktycznych lub dostępnego oprogramowania do demonstracji działania algorytmów;

5) prezentuje przykłady zastosowań informatyki w innych dziedzinach, w zakresie pojęć, obiektów oraz algorytmów.

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

1) projektuje, tworzy i testuje programy w procesie rozwiązywania problemów. W programach stosuje: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje oraz zmienne i tablice;

2) projektuje, tworzy i testuje oprogramowanie sterujące robotem lub innym obiektem na ekranie lub w rzeczywistości;

4) zapisuje efekty swojej pracy w różnych formatach i przygotowuje wydruki;

III. Posługiwanie się komputerem, urządzeniami cyfrowymi i sieciami komputerowymi. Uczeń:

2) rozwija umiejętności korzystania z różnych urządzeń do tworzenia elektronicznych wersji tekstów, obrazów, dźwięków, filmów i animacji;

3) poprawnie posługuje się terminologią związaną z informatyką i technologią.

IV. Rozwijanie kompetencji społecznych. Uczeń:

1) bierze udział w różnych formach współpracy, jak: programowanie w parach lub w zespole, realizacja projektów, uczestnictwo w zorganizowanej grupie uczących się, projektuje, tworzy i prezentuje efekty wspólnej pracy;

Wprowadzenie w tematykę i integracja grupy (10 min.)

Ucniowie przypominają sobie, co robiliśmy na ostatniej lekcji i co z niej pamiętają.

Następnie nauczyciel przedstawia koncepcję listy w programowaniu na przykładzie np. listy zakupów.

```
shopping_list = ["Milk", "Bread", "Cheese"]
```

Prosimy uczniów, aby wymyślili własną listę określonych elementów. Pomysł można przedstawić na forum klasy.

Część zasadnicza (30 min.)

Stworzymy listę obrazków przewijaną za pomocą przycisków. Pierwszym krokiem będzie więc zdefiniowanie listy z obrazkami, na przykład takiej, jak poniżej:

```
image_list = [Image.HAPPY, Image.SAD, Image.ANGRY, Image.CONFUSED]
```

W języku Python do definiowania list używamy nawiasów kwadratowych [], a poszczególne jej elementy oddzielamy znakiem przecinka.

Do poszczególnych elementów listy, możemy odwoływać się za pomocą indeksu (liczby). Należy zwrócić uwagę na to, że elementy listy (tak jak w wielu innych językach programowania) liczymy od 0.

Przykładowo, aby dostać się do pierwszego elementu listy **image_list**, możemy użyć kawałka kodu podanego poniżej:

```
image_list[0]
```

Możemy wyświetlić ten obrazek na naszym Micro:bitcie:

```
display.show(image_list[0])
```

Zachęcamy uczniów, aby zmieniali wyświetlany element, modyfikując indeks. Docelowo chcemy jednak, by indeks mógł zmieniać się w czasie wykonywania programu. W tym celu zdefiniujemy zmienną, która będzie przechowywała indeks obrazka z listy.

```
index = 0
```

Teraz jesteśmy w stanie zmodyfikować nasz indeks w sposób dynamiczny z poziomu programu. W kodzie wykorzystamy wyrażenie **index += 1**, które jest skrótową formą zapisu **index = index + 1**. Operację zwiększenia wartości zmiennej o jeden nazywamy inkrementacją. Poniższy kod uczniowie uzupełniają zgodnie ze wskazówkami w komentarzach.

```
1 from microbit import *
2
3 image_list = [] # zdefiniuj własną listę obrazków
4
5 index = 0
6
7 while True:
8     # napisz tutaj kod, który po wciśnięciu przycisku B będzie przewijał listę
   w prawo
9     display.show(image_list[index])
```

Przykładowe rozwiązanie:

```
1 from microbit import *
2
3 image_list = [Image.HAPPY, Image.SAD, Image.ANGRY, Image.CONFUSED]
4
5 index = 0
6
7 while True:
8     if button_b.was_pressed():
9         index += 1 # zwiększenie indeksu o 1
10        display.show(image_list[index])
```

Uczniowie testują program i muszą odpowiedzieć na pytanie, co dzieje się w momencie naciśnięcia przycisku na ostatnim elemencie listy. Po tej części nauczyciel komentuje przyczynę występującego błędu i proponuje rozwiązanie poprzez zerowanie indeksu w momencie, gdy indeks przekroczy liczbę elementów w liście. Zerowanie to powrót do pierwszego elementu listy, co umożliwi nam jej przewijanie cykliczne w nieskończoność. Ponadto wykorzystamy funkcję `len()`, która umożliwi sprawdzenie długości listy. Zwróćmy uczniom uwagę, że indeks ostatniego elementu listy to jej długość minus jeden.

```
1 from microbit import *
2
3 image_list = [Image.HAPPY, Image.SAD, Image.ANGRY, Image.CONFUSED]
4
5 index = 0
6
7 while True:
8     if button_b.was_pressed():
9         index += 1 # zwiększenie indeksu o 1
10        if index == len(image_list):
11            index = 0;
12        display.show(image_list[index])
```

Teraz program nie zgłasza żadnych błędów, więc można pokusić się o dodanie przewijania listy obrazków w drugą stronę z wykorzystaniem przycisku A. W kolejnej wersji programu zastosujemy analogiczny do inkrementacji zapis `index -= 1`, który jest skrótową formą zapisu `index = index - 1`. Operację zmniejszenia wartości zmiennej o jeden nazywamy dekrementacją. Poniższy kod uczniowie uzupełniają zgodnie ze wskazówkami w komentarzach.

```
1 from microbit import *
2
3 image_list = [Image.HAPPY, Image.SAD, Image.ANGRY, Image.CONFUSED]
4
5 index = 0
6
7 while True:
8     if button_b.was_pressed():
9         index += 1 # zwiększenie indeksu o 1
10        if index == len(image_list):
11            index = 0;
12        # napisz tutaj kod, analogiczny do powyższego
13        # który po wciśnięciu przycisku A będzie przewijał listę w lewo
14        display.show(image_list[index])
```

Przykładowe rozwiązanie:

```
1 from microbit import *
2
3 image_list = [Image.HAPPY, Image.SAD, Image.ANGRY, Image.CONFUSED]
4
5 index = 0
6
7 while True:
8     if button_b.was_pressed():
9         index += 1 # zwiększenie indeksu o 1
10        if index == len(image_list):
11            index = 0;
12        elif button_a.was_pressed():
13            index -= 1 # zmniejszenie indeksu o 1
14            if index == -1:
15                index = len(image_list) - 1;
16        display.show(image_list[index])
```

W powyższym kodzie, analogicznie jak w przypadku przewijania w prawo, musimy uważać na indeks, by nie zmniejszyć go zbyt bardzo. Dlatego, gdy tylko osiąga wartość -1, zmieniamy go na indeks ostatniego elementu listy. Daje nam to w rezultacie możliwość cyklicznego przewijania listy w lewo w nieskończoność.

Ponieważ możemy przewijać w obie strony, zbędne jest modyfikowanie indeksu przy przekroczeniu jego krańcowych wartości (-1, len()). Prosimy więc uczniów o zmodyfikowanie programu tak, by w momencie dojścia do krańcowego elementu indeks nie zmieniał się przy próbie przekroczenia zakresu. Poniżej przykładowe rozwiązanie tego zadania:

```
1 from microbit import *
2
3 image_list = [Image.HAPPY, Image.SAD, Image.ANGRY, Image.CONFUSED]
4
5 index = 0
6
7 while True:
8     if button_b.was_pressed():
9         index += 1 # zwiększenie indeksu o 1
10        if index == len(image_list):
11            index -= 1;
12        elif button_a.was_pressed():
13            index -= 1 # zmniejszenie indeksu o 1
14            if index == -1:
15                index += 1;
16        display.show(image_list[index])
```

Podsumowanie (5 min.)

Na końcu lekcji z uczniami krótko omawiamy poznane zagadnienia i zastanawiamy się, w jakich jeszcze innych sytuacjach może znaleźć zastosowanie lista.