Mikrobitowcy

Autorzy: Dawid Osuchowski, Filip Kłębczyk

Lekcja 7:

Micro:bit - instrukcja warunkowa, pętla zdarzeń i przyciski

W trakcie zajęć wrócimy do tematu instrukcji warunkowej, która pozwoli rozgałęziać sterowanie w naszych programach. Przyjrzymy się również przyciskom dostępnym na płytce Micro:bit i nauczymy się obsługiwać w kodzie zdarzenia z nimi związane.

Cele lekcji:
Uczeń powinien:
- Znać pojęcia: instrukcja warunkowa, przyciski, komentarze w kodzie, wartości logiczne True i False
- Wiedzieć, jak obsłużyć w programie zdarzenia związane z programowalnymi przyciskami na Micro:bicie
- Rozumieć, jak działa instrukcja warunkowa i jakie słowa kluczowe się składają na różne jej warianty
- Znać przeznaczenie komentarzy i wiedzieć, jak je stosować

Materiały pomocnicze:
- **http://microbit.org/**
- **https://bbcmicrobitmicropython.readthedocs.io/**
- **https://github.com/MicrobitPolska/FunWithMicrobit**

Pojęcia kluczowe:
→ instrukcja warunkowa → przycisk → pętla zdarzeń → komentarz → wartość logiczna → funkcja

Czas realizacji: 45 min.

Metody pracy:
- wykład,
- dyskusja,
- ćwiczenia praktyczne przy komputerze,
- 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;
5) wyszukuje w sieci informacje potrzebne do realizacji wykonywanego zadania, stosując złożone postaci zapytań i korzysta z zaawansowanych możliwości wyszukiwarek.
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;

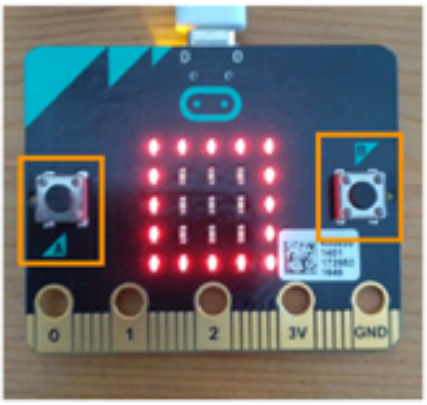
Wprowadzenie w tematykę i integracja grupy (10 min.)

Nauczyciel pyta uczniów, czy spotkali się z pojęciem instrukcji warunkowej w programowaniu? Jeśli nauczyciel zrealizował z uczniami lekcje 1-4, przypomina zakres lekcji dotyczącej instrukcji warunowej.

Jeżeli uczniowie nie potrafią wytłumaczyć działania instrukcji warunkowej lub mają z tym problemy, nauczyciel wyjaśnia pojęcie oraz stara się je zobrazować na łatwo przystępnych przykładach (np. zwrotnica kolejowa decydująca o tym, którym torem pojedzie pociąg).

Następnie nauczyciel pyta, jakie urządzenia w domu zawierają mechaniczne przyciski (np. klawiatura, pralka, konsole, pilot do telewizora, kontroler do gier) w przeciwieństwie do interfejsów dotykowych (np. w martfonach) i przeprowadza dyskusję na temat zalet jednych i drugich.

W dalszej kolejności pokazuje, gdzie znajdują się programowalne przyciski A i B na Micro:bicie (w przeciwieństwie do nieprogramowalnych jak przycisk Reset). Potem przedstawia koncepcję pętli zdarzeń i tego, jak ją wykorzystać do obsługi naciśniętych przycisków urządzenia.



Część zasadnicza (30 min.)

Napiszemy program, który po naciśnięciu przycisków będzie zmieniać obrazki na wyświetlaczu. Nauczyciel prosi uczniów, aby zastanowili się, jakie obrazki chcą wyświetlać.

Dla niezdecydowanych osób, możemy zaproponować obrazki ze zwierzętami (np. **Image.RABBIT**, **Image.COW**, **Image.DUCK** itp.).

Tworzymy pętlę nieskończoną, w której będziemy obsługiwać zdarzenia związane z przyciskami.

```
1  while True:
2      # miejsce na kod związany z obsługą przycisków
```

Przedstawiamy uczniom wbudowane obiekty: **button_a** oraz **button_b**, będące reprezentacją fizycznych przycisków A i B na urządzeniu. Każdy z obiektów posiada metody (funkcje wywoływane na rzecz obiektu):

- **is_pressed()** - zwraca wartość **True**, jeżeli przycisk jest wciśnięty dokładnie w momencie wywołania metody; **False** w przeciwnym wypadku, czyli gdy przycisk dokładnie w tym momencie nie jest wciśnięty. Metoda ta najlepiej sprawdza się w sytuacji, gdy chcemy sprawdzić, czy przycisk jest przytrzymywany czy nie.
- **was_pressed()** - zwraca wartość **True** lub **False**, aby wskazać, czy przycisk był wciśnięty od momentu rozpoczęcia działaniu programu lub ostatniego wywołania tej metody. Różnica w stosunku do **is_pressed()** polega na tym, że **was_pressed()** może zwrócić **True**, gdy w danym momencie nie jest wciśnięty przycisk - wystarczy, że był wciśnięty wcześniej. Metoda ta najlepiej sprawdza się w sytuacji, gdy chcemy sprawdzić, czy przynajmniej jeden raz przycisk został wciśnięty.
- **get_presses()** - zwraca liczbę naciśnięć przycisku do momentu wywołania tej metody; następne wywołanie zwraca liczbę naciśnięć przycisku od momentu jej ostatniego wywołania. Metoda ta najlepiej sprawdza się w przypadku, gdy interesuje nas dokładna liczba jego naciśnięć (na przykład w zależności od liczby naciśnięć uruchamiamy inną opcję).

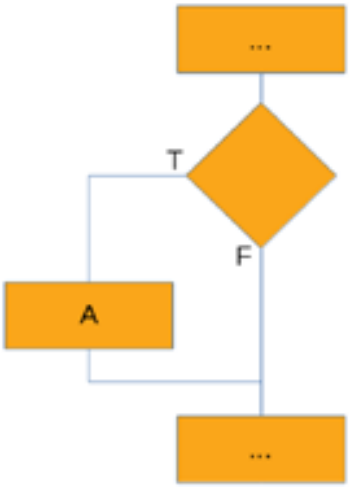
True i **False** w Pythonie to wartości logiczne zwracane najczęściej przez funkcje lub w wyniku porównań i są one używane w pętlach i instrukcjach warunkowych.

Za pomocą instrukcji warunkowej (**if**), sprawdzamy czy przycisk A (**button_a**) został wciśnięty i wyświetlamy (**display.show**) wybrany przez nas obrazek (**Image.RABBIT**):

```
1  from microbit import *
2
3  while True:
4      if button_a.is_pressed():
5          display.show(Image.RABBIT)
```

Uczniowie powinni zfleszować swój kod na urządzenie, następnie poeksperymentować z naciskaniem przycisków oraz obserwować działanie programu.

Działanie instrukcji warunkowej **if** w najprostszej postaci ilustruje poniższy schemat blokowy. W przypadku spełnienia warunku wykonujemy pewien blok operacji (na schemacie A), po czym przechodzimy do wykonania dalszej części programu. Gdy warunek nie jest spełniony, wykonanie programu przechodzi do dalszej części programu.



Odpowiadający kod w języku Python:

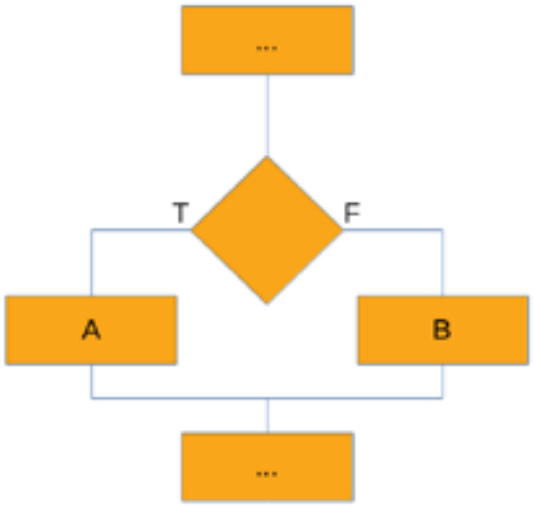
```
...
if warunek:
    Blok instrukcji A
...
```

Wyświetlanie jednego obrazka szybko się nudzi, dlatego lekko zmodyfikujemy nasz program. Teraz jeden obrazek będzie wyświetlać się przy naciśniętym przycisku, a drugi bez wciśniętego przycisku. Do tego celu wykorzystamy słowo kluczowe **else** (oznacza «w przeciwnym wypadku»), rozbudowując naszą instrukcję warunkową o tę możliwość:

```
1  from microbit import *
2
3  while True:
4      if button_a.is_pressed():
5          display.show(Image.RABBIT)
6      else:
7          display.show(Image.DUCK)
```

Jeszcze raz fleszujemy kod na urządzenie. Prosimy uczniów, aby opisali co zmieniło się w działaniu programu.

Działanie instrukcji warunkowej **if** z **else** można przedstawić na poniższym schemacie blokowym. W przypadku spełnienia warunku, wykonujemy pewien blok operacji A, po czym przechodzimy do wykonania dalszej części programu. Gdy warunek nie jest spełniony, wykonujemy pewien blok operacji B, po czym przechodzimy do wykonania dalszej części programu.



Odpowiadający kod w języku Python:

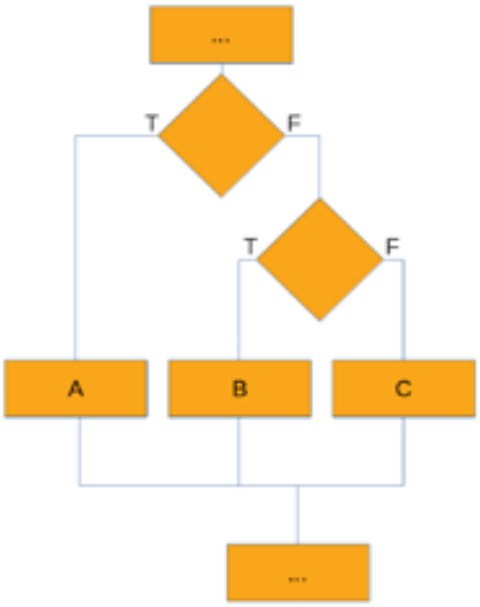
```
...
if warunek:
    Blok instrukcji A
else:
    Blok instrukcji B
...
```

W dotychczasowych programach przycisk B nie został wykorzystany, dlatego dodamy jego obsługę. Wyświetlimy za jego pomocą inny obrazek:

```
1  from microbit import *
2
3  while True:
4      if button_a.is_pressed():
5          # przycisk A jest wciśnięty
6          display.show(Image.RABBIT)
7      elif button_b.is_pressed():
8          # przycisk B jest wciśnięty
9          display.show(Image.COW)
10     else:
11         # żaden z przycisków nie jest wciśnięty
12         display.show(Image.DUCK)
```

Nauczyciel zwraca uczniom uwagę, że linie rozpoczęte znakiem # to komentarze, których nasz program nie wykonuje i służą one do umieszczania informacji pomocnych dla programisty. Inaczej mówiąc jest to taka dokumentacja programu w jego kodzie.

Działanie instrukcji warunkowej **if-elif-else** można przedstawić na poniższym schemacie blokowym. W przypadku spełnienia pierwszego warunku, wykonujemy pewien blok operacji A, po czym przechodzimy do wykonania dalszej części programu (nie sprawdzamy już warunku drugiego). Gdy pierwszy warunek nie jest spełniony, sprawdzamy warunek drugi. Jeśli jest on spełniony, wykonujemy pewien blok operacji B, po czym przechodzimy do wykonania dalszej części programu.



Odpowiadający kod w języku Python:

```
...
if warunek 1:
    Blok instrukcji A
elif warunek 2:
    Blok instrukcji B
else:
    Blok instrukcji C
...
```

Liczbę warunków możemy zwiększać dodając kolejne **elif**.

```
...
if warunek 1:
    Blok instrukcji A
elif warunek 2:
    Blok instrukcji B
elif warunek 3:
    Blok instrukcji C
elif warunek 4:
    Blok instrukcji D
else:
    Blok instrukcji E
...
```

Warto pamiętać, że **else** jest zawsze opcjonalny. To znaczy, że jeśli nie chcemy wykonywać operacji w przypadku gdy żaden z warunków nie jest spełniony, po prostu pomijamy **else**.

```
...
if warunek 1:
    Blok instrukcji A
elif warunek 2:
    Blok instrukcji B
elif warunek 3:
    Blok instrukcji C
elif warunek 4:
    Blok instrukcji D
...
```

Podsumowanie (5 min.)

W ramach podsumowania nauczyciel dyskutuje z uczniami na temat zrealizowanych rozwiązań i wspólnie zastanawiają się, jak można by wykorzystać nabytą w czasie zajęć wiedzę przy realizacji potencjalnych pomysłów własnych.