OSTRAVSKÁ UNIVERZITA
PEDAGOGICKÁ FAKULTA
KATEDRA INFORMAČNÍCH A KOMUNIKAČNÍCH TECHNOLOGIÍ

Algoritmizace v prostředí ZŠ

Bakalářská práce

Autor práce: Vojtěch Prokop
Vedoucí práce: Mgr. Tatiana Havlásková, Ph.D.

2022

UNIVERSITY OF OSTRAVA
FACULTY OF EDUCATION
DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGIES

Algorithmization in elementary school

Bachelor thesis

Author: Vojtěch Prokop
Supervisor: Mgr. Tatiana Havlásková, Ph.D.

2022

OSTRAVSKÁ UNIVERZITA

Pedagogická fakulta

Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Vojtěch PROKOP**
Osobní číslo:	**L19331**
Studijní program:	**B7507 Specializace v pedagogice**
Studijní obor:	**Informační a komunikační technologie ve vzdělávání**
Téma práce:	**Algoritmizace v prostředí ZŠ**
Zadávající katedra:	**Katedra informačních a komunikačních technologií**

Zásady pro vypracování

Cílem práce je popsat různé možnosti rozvoje algoritmizace u žáků 2. stupně ZŠ, zaměřit se jak na digitální, tak i na nedigitální aktivity.

Harmonogram bakalářské práce:
1) Konzultace s vedoucím práce (zadaní tématu, stanovení etap práce): listopad 2020.
2) Studium doporučených materiálů: leden – březen 2021
3) Návrh a tvorba aktivit: duben – srpen 2021
4) Kompletace aktivit: září – říjen 2021
5) Finální zpracovaní práce: listopad 2021 – únor 2022
6) Odevzdání: duben 2022

Rozsah pracovní zprávy:
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

Horník, T., Musílek, M., Milková, E. (2019). Didaktika programování. Hradec Králové.
Jakeš, T., Baťko, J., Simbartl, P. (2020). Robotika s LEGO Mindstorms pro 2. stupeň ZŠ. Západočeská univerzita v Plzni.
Musílek, M. (2012). Dětské programovací jazyky. Hradec Králové.
Sweigart, A. (2015). Invent Your Own Computer Games with Python 3rd Edition.

Vedoucí bakalářské práce: **Mgr. Tatiana Havlásková, Ph.D.**
Katedra informačních a komunikačních technologií

Datum zadání bakalářské práce: 28. února 2021
Termín odevzdání bakalářské práce: 1. dubna 2022

Mgr. Tatiana Havlásková, Ph.D.
vedoucí bakalářské práce

doc. Ing. Kateřina Kostolányová, Ph.D.
vedoucí katedry

ABSTRAKT

Bakalářská práce se zabývá rozvojem algoritmizace v prostředí druhého stupně základních škol. Teoretická část je zaměřena na uvedení do problematiky algoritmizace a její terminologie. Dále pak představuje přehled vybraných pomůcek určených k jejímu rozvoji. Praktická část se soustředí na návrh aktivit vedoucích k rozvoji algoritmizace a algoritmického myšlení, a proto byly vytvořeny, jak digitální, tak nedigitální aktivity. Digitální aktivity jsou založeny na blokovém programování vývojové desky Arduino v online prostředí s možností simulace. Nedigitální aktivity se zaměřují na tvorbu algoritmu pomocí papírových kartiček, které představují základní algoritmické konstrukce.

Klíčová slova:

Algoritmické myšlení, algoritmizace, algoritmus, Arduino, blokové programování, digitální aktivity, nedigitální aktivity

ABSTRACT

This Bachelor thesis deals with algorithmic development at the second stage of primary schools. The theoretical part focuses on introducing algorithmization and its specific terminology as well as it presents selected tools used for its development. The aim of the practical part is to design individual activities leading to algorithmic development and algorithmic way of thinking. Therefore, digital as well as non-digital activities were created for this purpose. Digital activities are based on block programming of development board Arduino in online environment also with the possibility of simulation. Non-digital activities focus on creating an algorithm using paper cards which present basic algorithmic constructions.

Keywords:

Algorithmic thinking, algorithmization, algorithm, Arduino, block programming, digital activities, non-digital activities

ČESTNÉ PROHLÁŠENÍ

Já, níže podepsaný student, tímto čestně prohlašuji, že text mnou odevzdané závěrečné práce v písemné podobě je totožný s textem závěrečné práce vloženým v databázi DIPL2. Prohlašuji, že předložená práce je mým původním autorským dílem, které jsem vypracoval samostatně. Veškerou literaturu a další zdroje, z nichž jsem při zpracování čerpal, v práci řádně cituji a jsou uvedeny v seznamu použité literatury.

Ostrava dne 18.3.2022

……
podpis studenta

PODĚKOVÁNÍ

Rád bych poděkoval své vedoucí bakalářské práce Mgr. Tatianě Havláskové, Ph.D. za cenné rady, odborné vedení a vstřícnost při konzultacích.

OBSAH

ÚVOD

Používání moderních technologií včetně výpočetní techniky není v současnosti výsada malé skupiny odborníků, nýbrž široké veřejnosti. S rozvojem moderních technologií přichází také potřeba rozvoje digitálních kompetencí všech jedinců. Součástí těchto kompetencí je informatické a algoritmické myšlení. Rozvoj takového myšlení nemá význam pouze v oblasti informačních a komunikačních technologií, ale také v rovině běžného života. Jedná se o přístup k řešení jakéhokoliv problému v obecné rovině. Cílem této práce je popsat různé možnosti rozvoje algoritmizace u žáků druhého stupně základních škol v České republice a navrhnout konkrétní aktivity, které k rozvoji pomohou. Teoretická část je zaměřena na uvedení do problematiky algoritmizace a jsou zde nastíněny konkrétní možnosti jejího rozvoje. Praktická část se zaměřuje na návrh aktivit pro žáky druhého stupně ZŠ. Přestože je s rozvojem algoritmizace spojována především výpočetní technika, práce obsahuje jak digitální, tak nedigitální aktivity.

1 ZÁKLADNÍ POJMY

1.1 Algoritmus

Algoritmy zasahují do života pravděpodobně většiny lidí, aniž by si to plně uvědomovali. Díky algoritmům lze vykonávat dnes již běžné činnosti, například využívat ke své práci počítač, chytrý telefon nebo také zapnout pračku či jiný domácí spotřebič. Algoritmy jsou prakticky všude kolem nás. Kamerové systémy, řízené algoritmy hlídají naši bezpečnost, jiné řídí dopravu. Tento výčet je jen malým zlomkem toho, co algoritmy představují pro moderní společnost. Otázkou je, co to vlastně algoritmy jsou.

Algoritmus je definován jako postup k řešení daného problému (Šarmanová, 2014, s. 5). Avšak ne každý postup k řešení může být nazýván algoritmem. Aby se tak nazývat dal, musí splňovat určitá kritéria. Podle Šarmanové (2014, s. 15) se jedná o tyto:

- Rezultativnost – Vždy vede ke správnému výsledku
- Hromadnost – Algoritmus neřeší pouze jeden problém, nýbrž je řešením pro celou skupinu problémů stejného charakteru
- Determinovanost – Neboli jednoznačnost, je vlastnost, která říká, že každý krok navazuje na další. To znamená, že nemůže nastat během řešení problému situace, která nemá jasný další krok. Ve chvíli, kdy nastane nejednoznačná situace, která vede k více možnostem, musí být jasně definováno, jak dále postupovat.
- Opakovatelnost – Tato vlastnost říká, že pokud zadáme pokaždé stejné vstupní údaje, musí být pokaždé totožný výstup neboli výsledek.
- Konečnost – Znamená, že výsledek je poskytnut v rozumně dlouhé době.

Většina autorů definuje pojem algoritmus podobným způsobem. Zde jsou některé z dalších definic:

- „algoritmus je podrobná posloupnost jednotlivých instrukcí vedoucí k vyřešení úkolu. V informatice programátoři pomocí algoritmu říkají počítači, jak daný úkol provést.“ (TYNKER, 2019)
- „Jednoduše lze říci, že algoritmus je postup s přesně popsanými kroky, který vede k danému cíli, tedy řešení úlohy. Formálně lze algoritmus definovat např. následovně: Jednoznačně stanovená posloupnost operací, které řeší daný problém“ (Blahuta, 2017, s. 10)

- Podle Kostolányové (2002, s. 9) je algoritmus předpis, který přesně definuje postup při řešení problému, má konečný počet kroků a vede k požadovanému výsledku.

1.2 Algoritmizace

Algoritmizace se zabývá tvorbou algoritmů. Je to tedy metodický přístup, jak postupovat ke tvorbě programu. Důležité je zmínit, že algoritmizace je v čase neměnná. Přestože se provedení programu mění, zastaralé programovací jazyky jsou nahrazeny novými, tak postup je stále stejný (Dohnal, 2009, s. 10).

Pokud má být úloha řešena pomocí algoritmu, je nutné aby byla specifikována. To znamená, že musí být k dispozici vstupní údaje. Tedy to, co má být řešeno a také je potřeba vědět, jaké mají být výstupní údaje – co má být výstup řešení. Je potřeba znát vstupní údaje a produktem řešení jsou výstupní údaje (Kostolányová, 2002, s. 9). Ve chvíli, kdy je zapotřebí omezit vstupní údaje definují se vstupní podmínky. Znamená to, že vstupní údaje nemohou být jakékoliv hodnoty. Stejně tak je nezbytno definovat výstupní podmínky, tedy požadavky na výsledek (Šarmanová, 2014, s. 17). Jinými slovy: "Vstupní a výstupní podmínka charakterizuje úlohu tj. specifikuje to, co má být řešeno. Nejedná se o konkrétní úlohu, ale o celou třídu úloh, jejichž vstupní údaje splňují vstupní podmínku a výstupní údaje výstupní podmínku" (Kostolányová, 2002, s. 10).

Postup pro vytvoření algoritmu podle Šarmanové (2014, s. 18):

- Formulace problému
- Analýza úlohy – V této fázi je zjištěno, zda je úloha řešitelná, jestli jsou vstupní údaje dostatečné, počet řešení a další specifika
- Algoritmizace úlohy - Výsledkem je podrobný algoritmus, podle kterého se sestaví počítačový program
- Ověření – V tomto posledním kroku dochází k ověření, zda jsou výsledky správné

Příklad: algoritmus zašroubování šroubku:

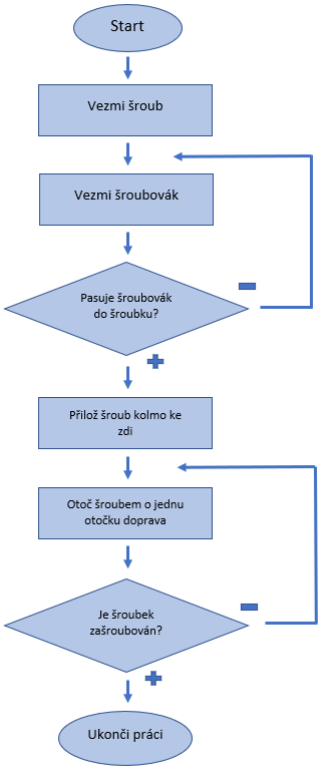
- Formulace problému
    - Zašroubovat šroubek do zdi
- Analýza úlohy
    - Vstupní proměnná – sada šroubováků, šroubek, zeď
    - Výstupní proměnná – šroubek zašroubován do zdi
- Algoritmizace úlohy a ověření výsledku

1. Vezmi šroubek
2. Vezmi šroubovák
3. Pasuje šroubovák do šroubku?
    - ANO – Pokračuj bodem 4
    - NE – Vrať se na bod 2
4. Přilož šroubek kolmo ke zdi
5. Zatlač špičku šroubováku na hlavičku šroubku a otoč šroubovákem ve směru hodinových ručiček o jednu otočku
6. Je šroubek zašroubován?
    - ANO – Pokračuj bodem 7
    - NE – Vrať se k bodu 5
7. Ukonči práci a polož šroubovák

Na tomto jednoduchém příkladu lze demonstrovat vlastnosti algoritmu. Vede vždy algoritmus ke správnému výsledku? Ano, jelikož šroubek je vždy ve výsledku zašroubován. Je hromadný? Šroubování lze aplikovat na různé druhy šroubku – algoritmus mění šroubovák tak dlouho, dokud nepasuje do hlavičky šroubku. Algoritmus je determinovaný – jednotlivé kroky na sebe navazují a vždy je jasný další krok. Výsledek skončí v rozumné době a při stejném vstupu dostaneme vždy stejný výstup. Výše uvedený příklad splňuje všechny základní podmínky k tomu, aby tento postup mohl být označen za algoritmus.
V podstatě existují jen čtyři možnosti, jak lze algoritmus vyjádřit.

1. Slovně – Postup řešení je vyjádřen ve srozumitelném jazyce (Blahuta, 2017, s. 14). Slovní popis se však stane nepřehledný, pokud bude takto popsán složitější algoritmus (Štefan, 2006, s. 6). Tento typ můžeme vidět v předcházejícím příkladu (viz algoritmus zašroubování šroubku).
2. Graficky – Jedná se o zápis pomocí grafických symbolů. Nejčastěji je pro vyjádření algoritmu použit například vývojový diagram a strukturogram (Štefan, 2006, s. 6). Ukázka vývojového diagramu se nachází na obr. 1.
3. Matematicky – Algoritmus je zapsán rovnicí nebo jiným matematickým způsobem (Blahuta, 2017, s. 14).

4. Pomocí programu – jednotlivé kroky algoritmu jsou napsány jako instrukce pro procesor. V tomto případě se již může hovořit o programování (Blahuta, 2017, s. 14). Ukázka jednoduchého programu, který je napsán pomocí programovacího jazyka Python, se nachází na obr. 2.



Obrázek 1: algoritmus

```python
print("kalkulacka\n")

cislo1=int(input("zadejte prvni cislo: "))
cislo2=int(input("zadejte prvni cislo: "))

scitani=cislo1+cislo2
odcitani=cislo1-cislo2
nasobeni=cislo1*cislo2
deleni=cislo1/cislo2

print("\nStiskni 1 pro soucet")
print("Stiskni 2 pro rozdil")
print("Stiskni 3 pro nasobek")
print("Stiskni 4 pro podil\n")

vyber=int(input("Vyberte volbu: "))

if vyber == 1:
    print("\nSoucet cisel je: ", scitani)
elif vyber == 2:
    print("\nRozdil cisel je: ", odcitani)
elif vyber == 3:
    print("\nNasobek cisel je: ", nasobeni)
elif vyber == 4:
    print("\nPodil cisel je: ", deleni)
else:
    print("\nNeplatna volba")

input("\nStisknete libovolnou klavesu pro konec")
```

Obrázek 2: program v jazyce Python

1.3 Základní algoritmické konstrukce

Základní konstrukce algoritmů lze rozdělit jako:

1. Sekvence (posloupnost)
2. Rozhodování (větvení)
3. Opakování (cykly)

Sekvence (posloupnost)

Aby mohl být sestaven algoritmus, který bude komunikovat prostřednictvím algoritmického jazyka, je zapotřebí určit příkazy. Příkazy jakožto elementární součásti algoritmu se budou vykonávat v pořadí tak, jak jsou zapsány. Tomuto se říká sekvence

(Skalka, 2007, s. 18). Sekvence neboli posloupnost je „řada za sebou jdoucích kroků, jejichž pořadí je předem pevně dáno. Posloupnost má svůj začátek a konec. Žádný krok nemůže být vynechán“ (Štefan, 2006, s. 12).

Rozhodování (větvení)

Možnost volby při rozhodování v průběhu algoritmu závisí na podmínce. V závislosti na splnění dané podmínky se algoritmus rozdělí a pokračuje v dané větvi (Skalka, 2007, s. 22). Větvení se skládá ze tří částí:

- První část je otázka, na kterou existuje buďto kladná nebo záporná odpověď (Pravdivá nebo nepravdivá).
- Druhá část se provede, pokud je odpověď na otázku kladná.
- Třetí část se naopak provede ve chvíli, kdy je odpověď na otázku záporná.

(Štefan, 2006, s. 15 - 18)

Otázka, tedy první část větvení je povinná. Druhá a třetí část povinná není. Pokud však větvení postrádá dvě poslední části, ztrácí smysl. Větvení se rozděluje na

- Úplné

  Úplné větvení znamená, že další kroky jsou vytvořeny v případě kladné i záporné odpovědi.
- Neúplné

  Neúplné větvení znamená, že chybí krok pro kladnou nebo zápornou odpověď.
- Vnořené

  U vnořeného větvení je dalším krokem pro kladnou nebo zápornou větev (nebo pro obě) další větvení.

(Štefan, 2006, s. 15 - 18)

Opakování (cykly)

Pokud se v algoritmu opakuje nějaká činnost nebo činnosti, využije se opakování neboli cyklus. Cyklus se skládá ze dvou částí. Z těla cyklu a podmínky cyklu.

Tělo cyklu je právě ta činnost, která se opakuje. Podmínka cyklu říká, jak dlouho se bude cyklus opakovat. Podle vztahu mezi podmínkou a tělem cyklu dělíme na

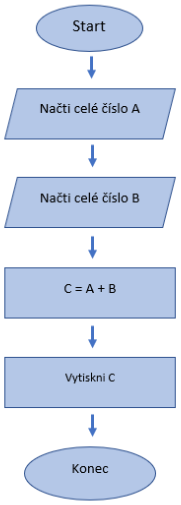
- cyklus se známým počtem opakování
- cyklus s podmínkou na začátku

- cyklus s podmínkou na konci.

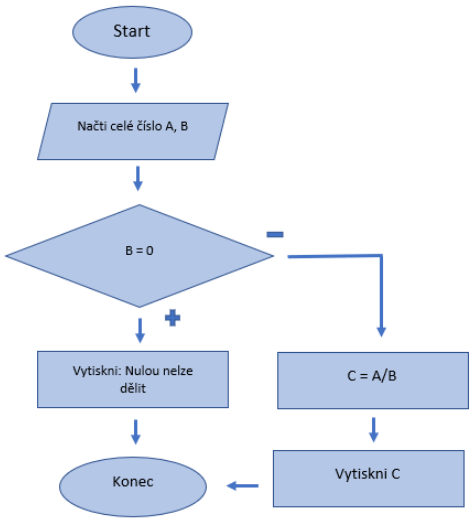
(Skalka, 2007, s. 31)

Cyklus se známým počtem opakování se používá tehdy, pokud daný počet opakování nezávisí na činnosti v těle cyklu. Cyklus s podmínkou na začátku funguje tak, že dojde nejdříve k vyhodnocení podmínky a pokud je výsledek podmínky pravdivý, provede se tělo cyklu. Po vykonání těla se automaticky provede návrat k podmínce cyklu, která se opět vyhodnotí. Cyklus se v tomto případě nemusí provést ani jednou, a to v případě kdy dojde k vyhodnocení podmínky jako nepravdivé hned na začátku. Naproti tomu u cyklu s podmínkou na konci se nejdříve vykoná tělo cyklu, tedy cyklus se vždy vykoná alespoň jedenkrát, a až pak se provede vyhodnocení podmínky. Pokud je výsledek podmínky nepravdivý, pokračuje znovu na začátek těla cyklu a ten se znovu vykoná. Poté se podmínka znovu vyhodnotí. Cyklus skončí ve chvíli, kdy je výsledek podmínky pravdivý (Štefan, 2006, s. 25).

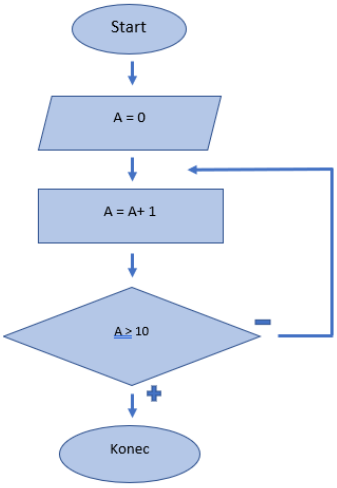
Ukázka vývojových diagramů základních algoritmických konstrukcí:



Obrázek 4: sekvence



Obrázek 5: větvení



Obrázek 3: cyklus

1.4 Programování

Pokud je zapotřebí komunikovat se zařízením, které je schopno vykonávat algoritmy, přichází nutnost nějakého jazyka, který umožní s takovým zařízením komunikovat. Aby člověk dokázal se zařízením srozumitelně komunikovat a psát algoritmy, byli vytvořeny

umělé, tzv. programovací jazyky. Jazyk, kterému rozumí počítač, se nazývá strojový kód. Pro člověka je tento kód příliš obtížný, a proto byli vytvořeny právě jazyky programovací. Také byli vytvořeny překladače, které převádějí programovací jazyky do strojového kódu. Zápis algoritmů do programovacího jazyka se nazývá programování. Výsledný produkt programování se nazývá program. Je to tedy algoritmus zapsaný v programovacím jazyce, který je následně do strojového kódu přeložen. Z předchozího textu vyplývá rozdíl mezi algoritmizací a programování. Algoritmizace je tedy pouze mezistupeň mezi formulací problému a jeho vyřešením v počítači. Algoritmizace tak představuje promyšlení algoritmu, který je následně naprogramován pomocí programovacího jazyka a v posledním kroku se program ladí a testuje (Skalka, 2007, s. 40 - 41).

Podle Pšenčíkové (2009) je programování definováno jako přepis algoritmu, který řeší určitý problém, do programovacího jazyka.

„Každý algoritmus je pouze teoretickým řešením postupu. Druhou částí je praktické programování. Programovat, tedy „napsat“ algoritmus v programovacím jazyce je praktickým výstupem úlohy k programování.“ (Blahuta, 2017, s. 13)

1.5 Informatické a algoritmické myšlení

1.5.1 Informatické myšlení

Webové stránky www.umimeprogramovat.cz hovoří o informatickém myšlení (v anglickém jazyce se tento termín nazývá computational thinking) jako o relativně novém pojmu, který se používá od roku 2006 ve smyslu dnešních definic. V České republice se tento termín používá ještě kratší dobu. Definice a základní principy jsou popsány téměř shodně jako na webu www.imysleni.cz, kde je informatické myšlení definováno jako způsob uvažování, zaměřující se na popis a analýzu problému a zabývá se hledáním efektivního řešení. Díky sady nástrojů a postupů se dají například systematicky vyhodnocovat různá řešení a nejvhodnější z nich aplikovat na konkrétní problém, který lze rozdělit na dílčí části, jenž se dají snáze řešit. Je možno plánovat a řídit činnosti, vytvářet postupy, které vedou úspěšně k cíli, i když je používá někdo jiný. Je zde taky zahrnuta práce s daty a jazyky, pomocí kterých se lze domluvit s počítači. K rozvoji informatického myšlení dochází pozvolna. Nejprve se uplatňují dané postupy na jednoduchých úlohách a posléze je lze aplikovat na

složitější. Čím více je řešený problém komplexnější, tím je efektivita takových nástrojů a postupů větší.

1.5.2 Algoritmické myšlení

Podle webu www.umimeprogramovat.cz je „algoritmické myšlení součástí obecnějšího informatického myšlení, které se zaměřuje na navrhování algoritmů... Algoritmické myšlení se typicky využívá při programování, tj. při zápisu algoritmů, které provádí počítač. Má však svoje využití i v běžném životě: typickými příklady jednoduchých algoritmů jsou recepty na vaření nebo instrukce k sledování cesty z jednoho místa na druhé.“

Pojem algoritmické myšlení je velmi často označován, jako jedna z nejdůležitějších kompetencí, kterou lze nabýt vzděláním v informatice. Jedná se o souhrn schopností, které zahrnují porozumění algoritmům a jeho konstrukcím. Například schopnost analyzovat daný problém a přesně jej specifikovat. Schopnost zvolit správný postup a tvořit algoritmy využívající základních konstrukcí k vyřešení daného problému. Algoritmické myšlení vyžaduje kreativitu ke tvorbě nových algoritmů, které vedou k vyřešení problému (Futschek, 2006. s. 160).

2 ROZVOJ ALGORITMIZACE U ŽÁKŮ ZŠ

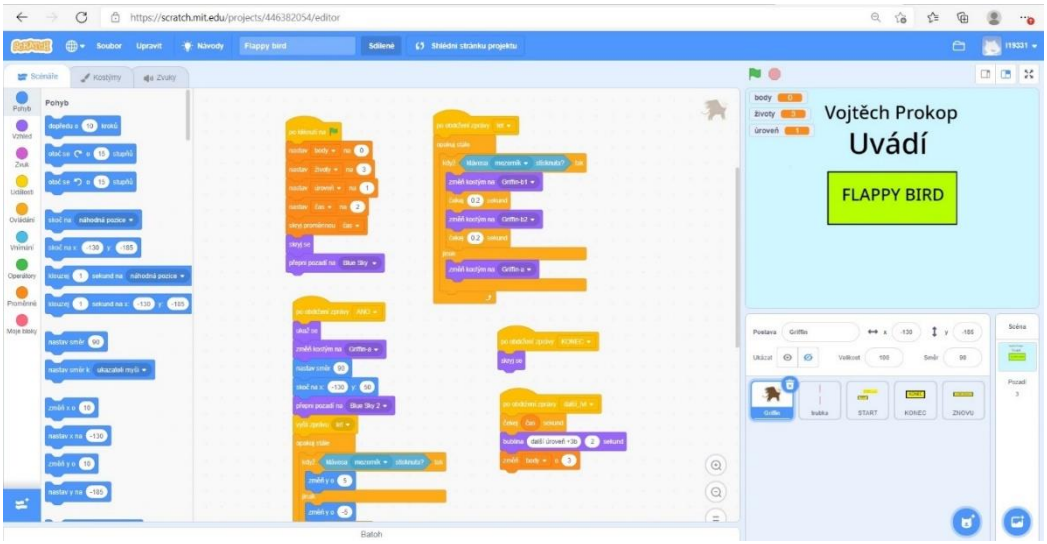
Algoritmické myšlení a algoritmizace se dají rozvíjet mnoha způsoby. Cílem této kapitoly je popsat aktivity, vedoucí k rozvoji algoritmizace u žáků druhého stupně základních škol. Popsány jsou aktivity digitální i nedigitální. Nejedná se o kompletní seznam všech dostupných, ale pouze o výběr některých. Výběr ukazuje velkou rozmanitost dostupných aktivit. K dispozici jsou nejen online nástroje a výukové programovací jazyky, ale také robotické pomůcky, programovatelné výukové desky nebo dokonce aktivity podobné deskovým hrám.

2.1 Programovací jazyk Scratch

Jedná se o dětský, blokově orientovaný programovací jazyk, který rozvíjí nejen u dětí algoritmické myšlení a algoritmizaci. Programovací jazyk je dostupný ze stránek www.scratch.mit.edu.

Pomocí jazyka Scratch je možno rozpohybovat postavičku, naprogramovat malování či tvořit interaktivní hry a další zajímavé projekty. Také díky tvorbě her je tento jazyk velmi populární pro žáky na základní škole. Není však podmínkou, že jej musí používat pouze děti. I pro dospělé může sloužit jako dobrá pomůcka pro pochopení základních algoritmických konstrukcí a porozumění algoritmům. Vytvořené projekty se navíc dají sdílet s ostatními uživateli, což poskytuje jak dobrou zpětnou vazbu pro tvůrce projektu, tak inspiraci pro ostatní. Do sdíleného projektu se dá nejen nahlížet, ale je zde také možnost celý projekt zkopírovat a libovolně upravovat.

Nespornou výhodou programovacího jazyka Scratch je jeho dostupnost. Je zcela zdarma. Prostředí je přístupné prostřednictvím webového prohlížeče a není tak potřeba nic instalovat. Pro některé uživatele může být nevýhodou nutnost připojení k internetu. Grafické a intuitivní prostředí je však pro začátečníky a žáky základních škol přívětivým vstupem do světa algoritmizace a programování. Ukázka prostředí jazyka Scratch se nachází na obr. 6.

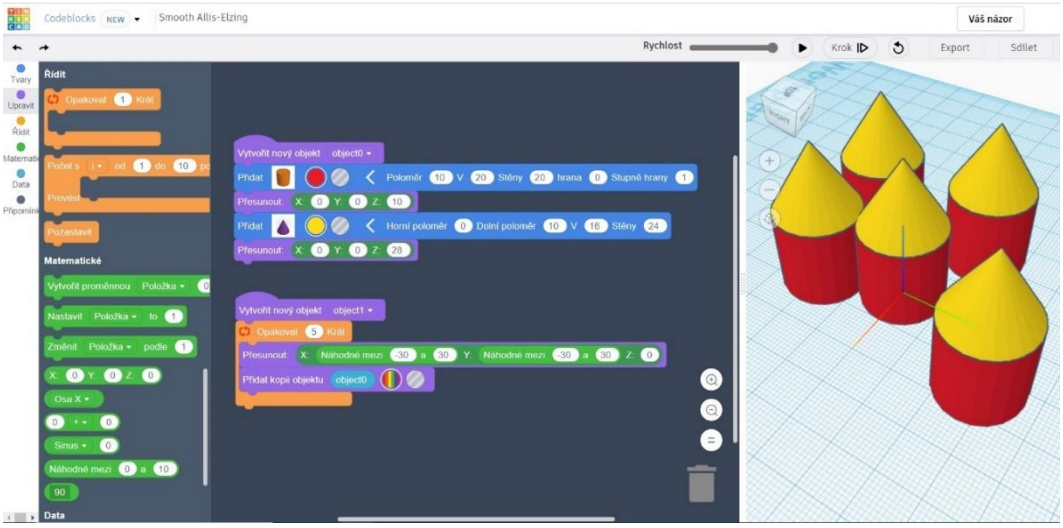


Obrázek 6: Scratch

2.2 www.tinkercad.com

Dalším online nástrojem, dostupným z webu www.tinkercad.com je všestranný produkt od firmy Autodesk. Není výhradně určen k výuce algoritmizace či programování. Zejména je známý jako graficky přívětivý prostředek pro tvorbu 3D modelů, vhodný pro začátečníky. Mezi jeho další funkce patří simulátor elektronických obvodů a nově také funkce Codeblocks, která slouží k vytváření 3D modelů za pomocí blokového programování.

Výsledný model, vytvořený za pomocí bloků se dá exportovat v několika formátech pro další práci. Vytvořené projekty lze také zaznamenat jako snímek obrazovky nebo animaci. Tato možnost je integrována přímo do aplikace Tinkercad. Vše je zcela zdarma, a kromě již zmíněných nabízí funkce, díky kterým lze software efektivně zařadit do výuky. Jedná se například o možnost vytvoření virtuální třídy, do které lze pozvat studenty. Učitel vidí pracovní postup svých studentů a má možnost dané práce opravovat. Pro samouky jsou zde lekce, díky kterým se krok po kroku mohou seznámit s prostředím a jeho funkcemi. Výhodou je, že vytvořené projekty lze sdílet s ostatními. Výše popsané prostředí je na obr. 7.

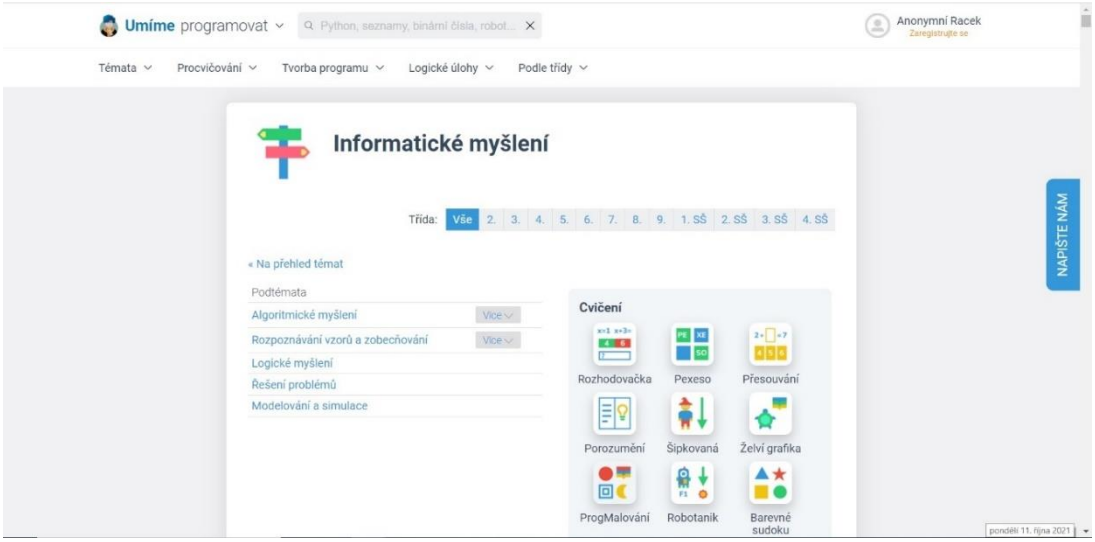


Obrázek 7: www.tinkercad.com

2.3 www.umimeprogramovat.cz

Webová stránka www.umimeprogramovat.cz spadá pod projekt Umíme, který byl původně vytvořen za účelem procvičování logických úloh. Za projektem stojí řada autorů, vývojářů a pedagogů. Ve své současné podobě se projekt zaměřuje na procvičování učiva z různých oborů, jako například matematika, čeština, jazyky, programování a další. Obsah je zaměřen na širokou skupinu studentů od 1. třídy až po maturitní ročníky na střední škole. Procvičování je koncipováno pestrou a hravou formou. Pokud student udělá během procvičování chybu, tak i s tím se počítá a všechny chyby lze napravit, jelikož chyby jsou neoddělitelnou součástí procesu učení. Procvičování je zdarma, ale je omezeno na určitý počet odpovědí za den. Pokud chce student neomezeně procvičovat, lze tak učinit po zaplacení poplatku. Kromě placeného účtu pro jednotlivce existuje také zvláštní ceník pro školy. Výhodou je nižší cena v přepočtu na žáka a možnost využívat učitelského módu, dělit žáky do virtuálních tříd a sledovat jejich postup v procvičování (www.umimeto.org).

Webová stránka slouží nejen k procvičování programování. Lze tady najít také témata určena pro rozvoj informatického myšlení, práce s daty či psaní všemi deseti. Jako podmnožinu informatického myšlení je možnost také rozvíjet za pomocí různých aktivit algoritmické myšlení. Celý web je přehledně členěn na jednotlivá témata a podtémata. V každém podtématu je soubor úloh, sloužící k procvičení konkrétních znalostí. Úlohy jsou řazeny od nejjednodušších po nejsložitější (www.umimeprogramovat.cz). Ukázka webu na obr. 8.



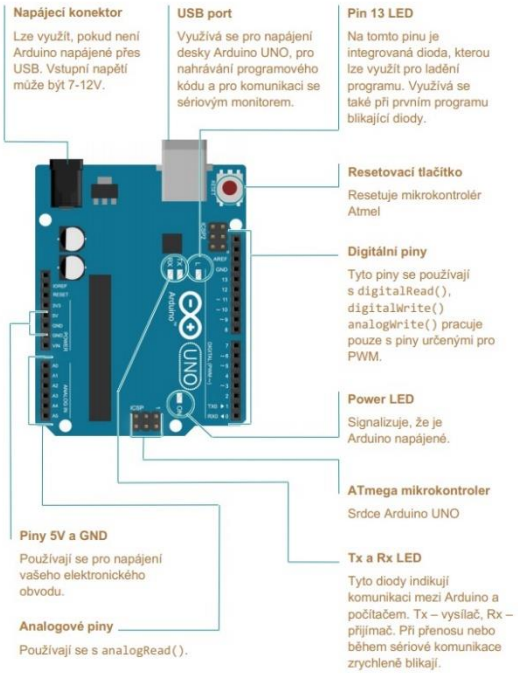
Obrázek 8: www.umimeprogramovat.cz

2.4 Arduino

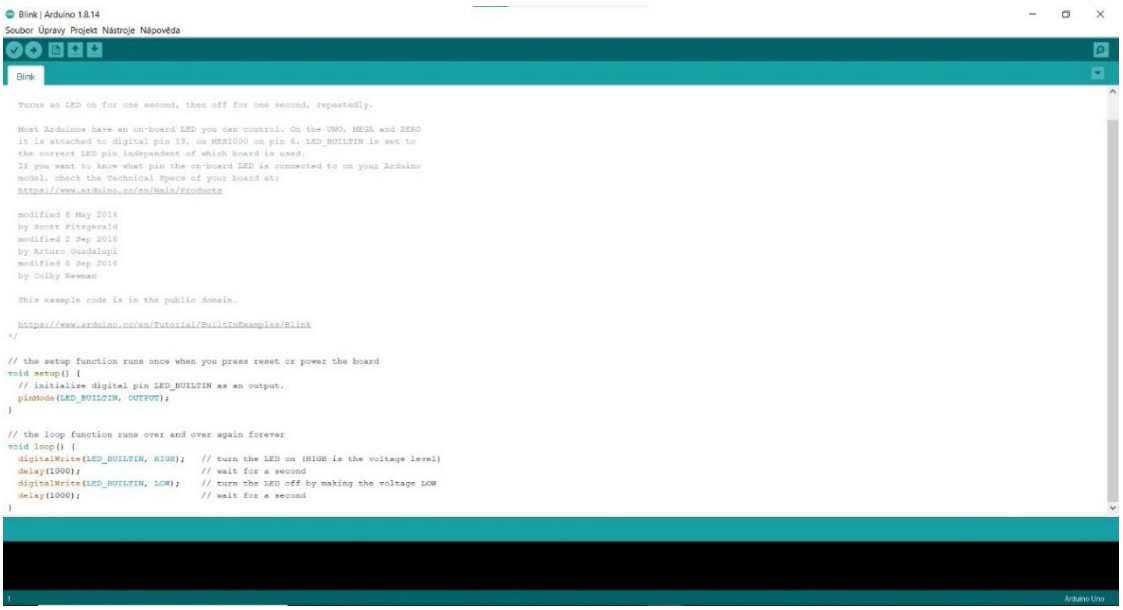
Arduino je programovatelná prototypovací deska. Jedná se o hardware, který lze pomocí příslušného software programovat. K desce je možno připojit další hardwarové komponenty a tvořit tak nejen jednoduché, ale i velmi složité projekty. Navíc existuje velké množství tzv. knihoven, které se používají pro rozšíření programu. Knihovny jsou open source. Platforma Arduino je určena pro výuku programování. Pomocí Arduino desky lze vytvářet roboty, dálkově řízená vozidla nebo jej například využít pro domácí automatizaci. Popis desky se nachází na obr. 10 (www.imysleni.cz). Jelikož je Arduino otevřená platforma, vznikají tak jeho klony od dalších výrobců. Hlavní částí většiny desek je procesor Atmel. Deska obsahuje i další komponenty, které jsou důležité pro jeho funkci. Většina desek komunikuje s počítačem za pomocí USB (Voda, 2018, s. 12).

Aby mohla být deska programována, je zapotřebí nainstalovat do počítače či chytrého zařízení potřebný software, tzv. Arduino IDE. Program je dostupný pro windows, linux i MAC OS. Arduino lze programovat pomocí jazyka C nebo C++. Pro snadnější použití byla vytvořena knihova jazyka C++ zvaná Wiring. Pomocí ní je Arduino možno naprogramovat. Někdy bývá knihovna Wiring označována jako samostatný programovací jazyk (Voda, 2018, s. 23 - 24). Ukázka prostředí s kódem v jazyce Wiring se nachází na obr. 9.

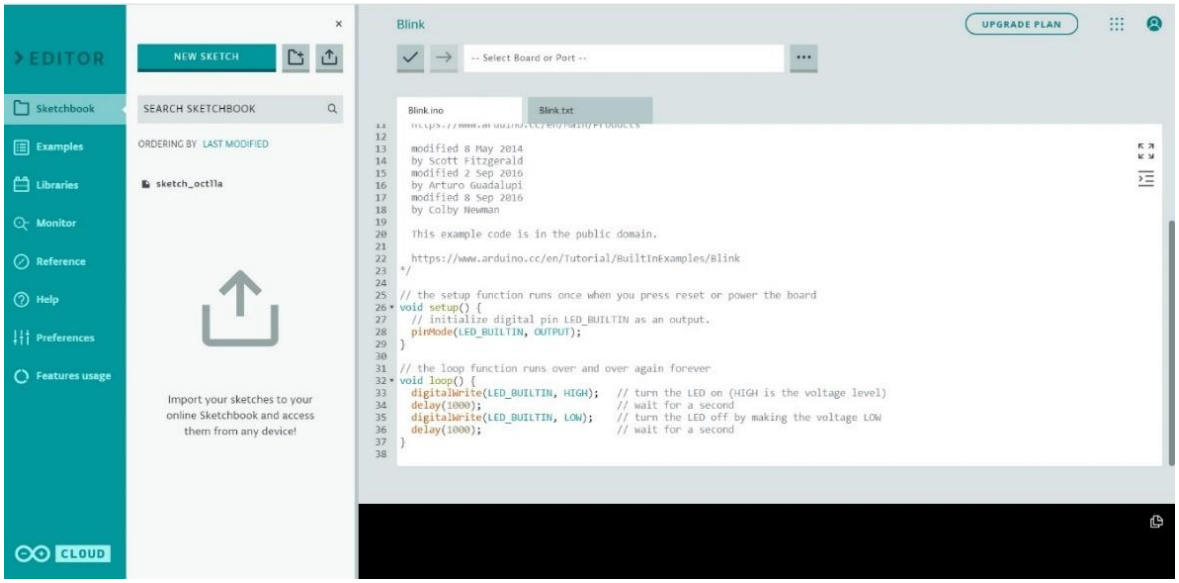
Alternativou k Arduino IDE je webová aplikace Web Editor, kterou není nutno instalovat. Zapotřebí je pouze registrace na oficiálním webu (www.arduino.cc). Prostředí webového editoru je velmi podobné prostředí Arduino IDE. Uživatel se tak téměř nemusí seznamovat s novým prostředím. Ukázka webového editoru se nachází na obr. 11.



Obrázek 9: Arduino, popis desky

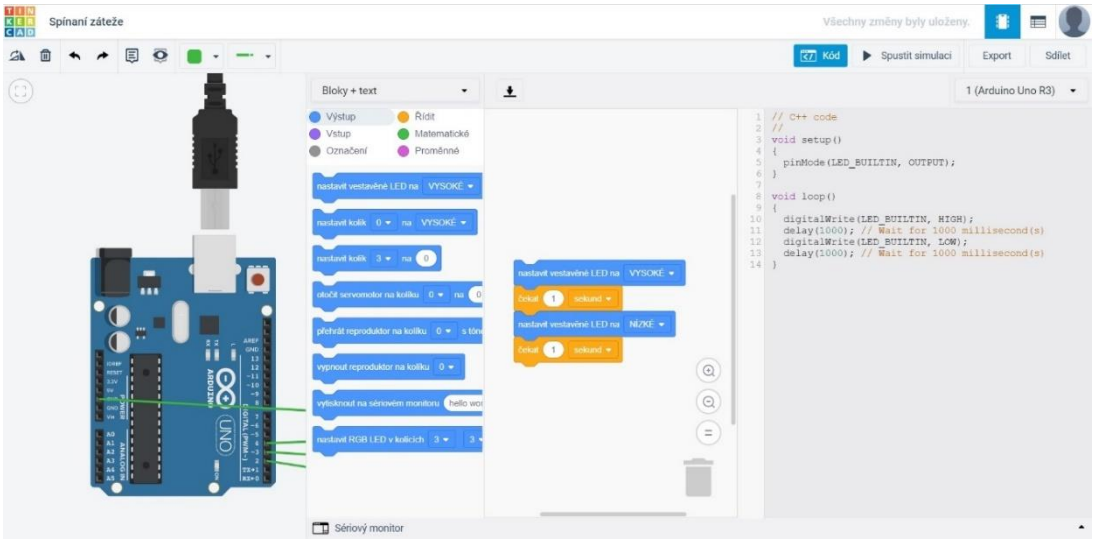


Obrázek 10: Arduino IDE

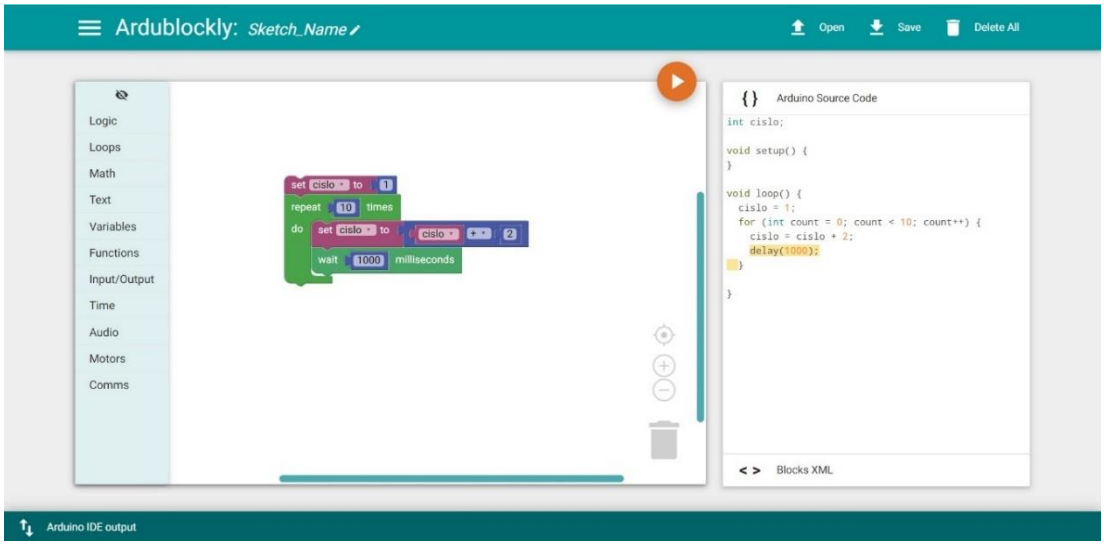


Obrázek 11: Arduino Web Editor

Arduino je určeno především středoškolákům, existuje však alternativa programovacího jazyka Wiring (C++). Arduino lze programovat pomocí blokového programování. Tato varianta je výhodná především pro mladší studenty, tedy také pro žáky druhého stupně ZŠ. Možností, jak Arduino programovat za pomocí bloků, existuje více. Jedna z nich tkví v možnosti programovat Arduino pomocí online nástroje www.tinkercad.com (ukázka na obr. 12). Jak již bylo zmíněno výše v této kapitole, jedna ze součástí aplikace je simulátor obvodů, ve kterém lze virtuálně zapojit elektronický obvod. V nabídce je kromě základních elektronických součástek také deska Arduino, kterou lze programovat, a to buď v programovacím jazyce C++ nebo pomocí bloků, případně lze kombinovat blokové a textové programování. Kombinace obou typů má tu výhodu, že kód, který je tvořen pomocí bloků, se zároveň automaticky generuje v textové podobě, což může usnadnit pozdější přechod do textového programovacího jazyka. V tomto prostředí lze virtuálně zapojený a naprogramovaný obvod také vyzkoušet. Pokud je kód funkční, je možno jej zkopírovat do Arduino IDE a nahrát přímo do fyzické desky Arduino. Jednou z dalších možností je použít webovou aplikaci Ardublockly (viz obr. 13). Lze využít nejen webovou (ardublockly.embeddedlog.com/demo/#), ale také desktopovou aplikaci. Princip programování je téměř totožný, jako ve výše zmiňovaném programu Tinkercad. Na rozdíl od Tinkercadu však nelze virtuálně simulovat funkčnost obvodu. Existuje i řada dalších možností, jak Arduino blokově programovat. Například pomocí programů Node Red, Ardublock, Scratch for Android a dalších aplikací dostupných na internetu.



Obrázek 12: Arduino a www.tinkercad.com



Obrázek 13: Ardublockly

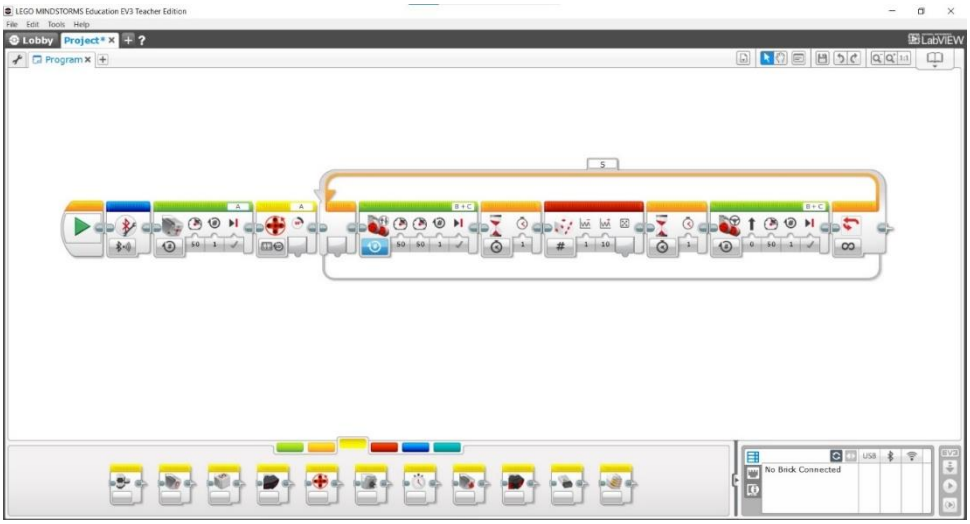
2.5 LEGO Mindstorms

Jedná se o programovatelné roboty z dílny LEGO. První řada robotů byla uvedena již v roce 1998. Pomocí aplikace, která je zdarma ke stažení, je možno přimět robota chodit, mluvit nebo dělat další věci, které si uživatel sám vymyslí. Aplikaci lze stáhnout nejen pro desktopové počítače, ale také pro chytré mobilní zařízení. (www.lego.com)

Online učebnice Robotika s LEGO Mindstorms (Jakeš, Baťko, Simbartl, 2020) říká, že je vhodné pro výuku žáku ve škole používat stavebnici LEGO Mindstorms EV3 Education (značená číslem 45544). Existuje totiž stavebnice ze stejné produktové řady určená pro domácí použití (HOME verze), která je levnější. Cena byla snížena na úkor rozmanitosti

sady. Navíc programovací prostředí v této verzi má úvodní obrazovku, která je plná animací a zvuků a odvádí tak pozornost studentů od práce. Aplikace ve verzi Education má dispozici učitelskou a studentskou verzi. Menší nevýhodou je, že aplikace není v českém jazyce. Jako alternativní software lze použít programovací prostředí Scratch, které má rozšíření pro LEGO Mindstorm nebo prostředí MakeCode, které je velmi podobné prostředí Scratch. Také lze použít prostředí OpenRoberta. Jedná se o variantu prostředí Blockly.

V programovacím prostředí LEGO MINDSTORMS Education EV3 (obr. 14) za pomocí speciálních bloků, představujících například jednotlivé kroky či pozice motorků, příkazy pro práci s Bluetooth nebo základní algoritmické konstrukce jako je cyklus, lze naprogramovat pohyby a chování robota. Bloky se přetáhnou z dolní části obrazovky na plochu nad ní. Výsledkem může být předem nastavený pohyb jako například tanec se zvukovým doprovodem či pohyb po předem vytyčené trase nebo robot sledující čáru a vyhodnocující překážky. Možnosti jsou velké a záleží na fantazii a kreativitě studenta, případně učitele.



Obrázek 14: LEGO Mindstorms

2.6 Ozobot

Jedna z dalších robotických pomůcek je Ozobot. Tento kulatý robot o velikosti několika centimetrů je určen pro výuku algoritmizace. Dá se programovat s využitím chytrého zařízení nebo pouze za pomoci barevných fixů.

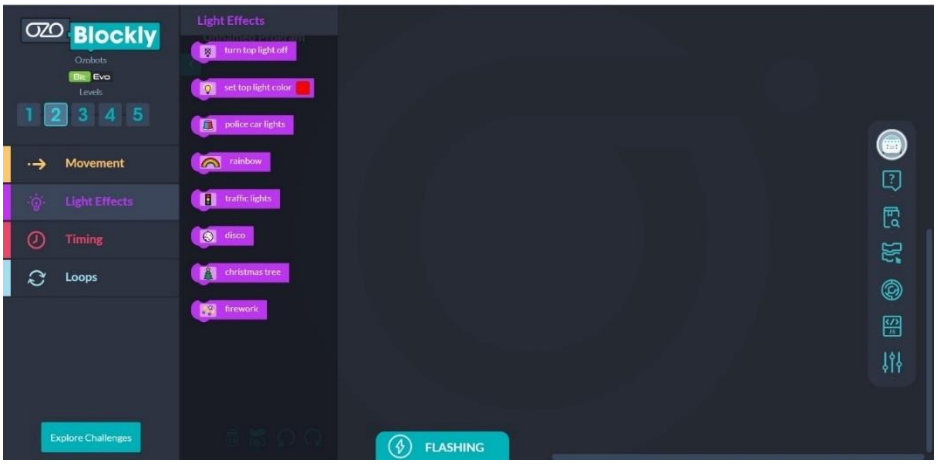
Na webu www.ozobot.com jsou výše zmiňované způsoby popsány takto:

- Barevné kódování
    - Díky senzorům, které snímají barvy, může robot sledovat čáru, po které jezdí.

- Díky kombinaci a střídání barev lze měnit rychlost, směr a další prvky.
- Na webu výrobce jsou k dispozici přesné kombinace barev, které reprezentují jednotlivé příkazy

- Kódování pomocí výpočetní techniky
  - Pomocí počítače či chytrého zařízení lze robota blokově programovat
  - Na desktopovém počítači lze robota programovat díky webové aplikaci OzoBlockly (ukázka na obr. 15). Podobně jako v prostředí Scratch, se bloky, reprezentující příkazy a další algoritmické konstrukce, skládají do sebe. Na výběr jsou bloky z kategorií pohyby, světelné efekty, časování a smyčky.
  - Prostředí je anglicky
  - K programování na mobilním zařízení je určená mobilní aplikace Evo by Ozobot. Programování probíhá podobně jako skrze webovou aplikaci.

Na oficiálních stránkách www.ozobot.com lze nalézt kromě návodů, potřebného software i podporu pro pedagogy (výuková videa, webináře a další podporu). Součástí stránek je také oficiální e-shop.

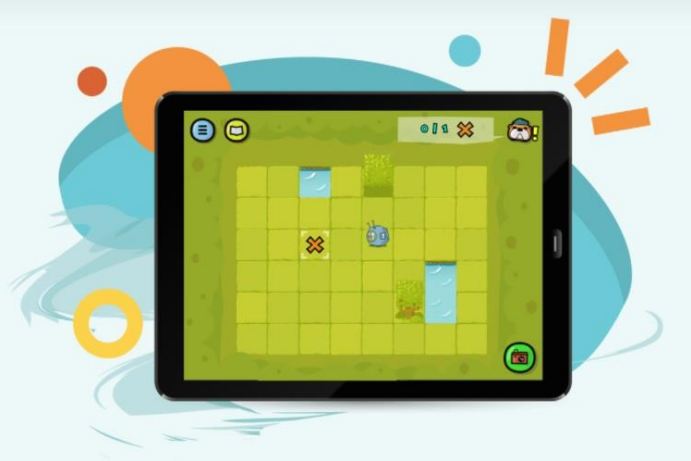


Obrázek 15: OzoBlockly

2.7 Scottie go!

Scottie GO! je vzdělávací hra z dílny polské firmy BeCREO Technologies, určena k výuce programování a rozvoji algoritmického myšlení. Kombinuje nejnovější technologie, jako je například rozšířená realita a osvědčené postupy vzdělávání. Téma celé hry je postaveno na postavě jménem Scottie, který ztroskotal na planetě zemi a žáci se snaží vyřešit úkoly, vedoucí k tomu, aby Scottie opravil svou vesmírnou loď a dostal se na svou domovskou planetu. Hra se vyrábí ve verzích pro vzdělávání a pro domácí použití.

Pro žáky druhého stupně základní školy je nejvhodnější verze Scottie Go! Edu nebo Scottie Go! Dojo pro distanční vzdělávání. Krabicová verze, která je vhodná pro žáky do patnácti let věku, funguje na principu blokového programování. Výsledný algoritmus je složen pomocí kartonových bloků. Algoritmus se stává s každým dalším úkolem obtížnějším. Další součástí hry je aplikace, kterou je možno nainstalovat za pomocí licenčního čísla na tři různá zařízení. Verze Scottie Go! Edu obsahuje 179 programovacích bloků, licenční klíč pro aplikaci, organizér na jednotlivé bloky, základní desku (na které lze bloky programu sestavovat) a manuál. Ukázka aplikace se nachází na obr. 16 a ukázka krabicové verze hry na obr. 17 (www.scottiego.com).



Obrázek 16: Scottie Go! aplikace

Obrázek 17: Scottie Go! krabicová verze

3 NÁVRH AKTIVIT

Cílem této kapitoly je návrh aktivit, rozvíjejících algoritmizaci a algoritmického myšlení u žáků druhého stupně základních škol. První část kapitoly je zaměřena na aktivity digitální, které využívají k dosažení cíle výpočetní techniku a digitální pomůcky. Druhá část se pak zaměřuje na nedigitální aktivity, tedy takové, u kterých není zapotřebí žádná digitální pomůcka ani výpočetní technika.

3.1 Digitální aktivity

Následující lekce jsou zaměřeny na rozvoj algoritmizace za pomocí digitálních pomůcek a výpočetní techniky. Všechny lekce lze realizovat jak na počítači (desktop či laptop), tak na mobilním dotykovém zařízení (ideálně na tabletu). Přesto se jako ideální volba jeví počítač. A to z důvodu větší obrazovky a možnosti práce s myší. Práce tak bude nejen přehlednější, a díky možnosti používat myš také snadnější.

Použitý software, na jehož základě jsou lekce realizovány, byl zvolen online nástroj www.tinkercad.com, který je blíže popsán v předešlé kapitole „Možnosti rozvoje algoritmizace u žáků ZŠ“. Pro začátek je nutné se registrovat. K tomu lze využít možnost „přidej se“, která se nachází v pravém horním rohu obrazovky. V nabídce jsou tři typy účtů, a tedy studentský, lektorský a účet pro osobní potřebu. Pro využití všech výhod ve školním prostředí je vhodné, aby si pedagog vytvořil účet lektora a žák účet studentský. Po jednoduché registraci a přihlášení se načte stránka, zabývající se tvorbou 3D modelů. Tato možnost nebude využita a pro další práci je nutné se v postranním panelu přepnout do sekce obvody, kde lze virtuálně simulovat funkci elektronických obvodů. Před tvorbou vlastních návrhů je dobré, aby se žáci seznámili s prostředím simulátoru. K tomu jsou určeny lekce pro začátečníky, díky kterým si uživatel vyzkouší všechny funkce a naučí se v simulátoru pracovat. Lekce lze nalézt v levém postranním panelu. Před samotným začátkem práce může pedagog vytvořit virtuální třídu, do které žáky připojí. Díky tomu lze sledovat postup všech žáků, hodnotit jejich práce apod. K využití této funkce je nutné přejít v menu na třídy a zvolit možnost vytvořit novou třídu a poté žáky přidat.

Práce v simulátoru je zaměřena na platformu Arduino, která je podrobněji popsána v předešlé kapitole „Rozvoj algoritmizace u žáků ZŠ“. Žáci mohou vytvořit nový obvod

pomocí tlačítka, které se nachází v horní části obrazovky a poté z pravého postranního panelu přidávat součástky včetně Arduina. Arduino je v tomto případě programováno za pomocí bloků. Po kliknutí na tlačítko kód, se otevře editor pro tvorbu programu. Ve vrchní liště lze přepnout mezi blokovým a textovým programováním či možnosti zobrazit kombinaci dvou předchozích. Pedagog se tak musí orientovat nejen v problematice programování, ale také by měl umět pracovat s vývojovou deskou Arduino. Vše probíhá virtuálně a není potřeba desku Arduino nebo jiné komponenty vlastnit. To poskytuje velkou výhodu nejen v podobě zkrácené doby přípravy pro pedagoga či finanční náročnosti, ale také v možnost vést hodinu distančně. Odpadají všechny problémy s instalací potřebného software či problémy s hardwarovými komponenty. Nemůže nastat situace, kdy žák zapojí například napájení desky opačně (prohodí kladný a záporný pól) a dojde tak ke zničení učební pomůcky. Všechny tyto výhody jsou vyváženy tím, že student si desku fyzicky nemůže zapojit a vidět svůj výsledek v reálné (fyzické) podobě.

Arduino se obvykle používá v kombinaci s dalšími elektronickými součástkami či moduly, představující vstupní či výstupní periferie. Pod pojmem vstupní periferie si lze představit například různá čidla, senzory, tlačítka, klávesnice apod. Mezi výstupní periferie pak lze zařadit například různé motorky, světelnou signalizaci, displeje a další. Vzhledem k věkové kategorii, pro kterou jsou aktivity určeny, jsou využity pouze jednoduché a základní periferie (součástky). Také není nutné znát žádné schématické značky či disponovat rozsáhlými znalostmi elektrotechniky. Úroveň lekcí je přizpůsobena žákům druhého stupně základních škol, především posledním ročníkům. Všechny potřebné znalosti budou žákům vysvětleny v rámci dané lekce. V následujících lekcích je využita varianta Arduino UNO R3, jedna z nejpoužívanějších variant této desky.

Před každou lekcí je důležité žákům vysvětlit nezbytný teoretický základ pro úspěšné absolvování dané lekce. Je proto důležité, aby pedagog disponoval alespoň základními znalostmi algoritmizace a programování a znalostmi desky Arduino a elektronických součástek.

3.1.1 Lekce 1

Název

Blikání LED

Cíl

Cílem první lekce je seznámit žáky s deskou Arduino UNO R3, možnostmi jejího programování a základními algoritmickými konstrukcemi. Výsledkem bude jednoduchý program, díky kterému žáci rozblikají LED.

Časová náročnost

45 minut

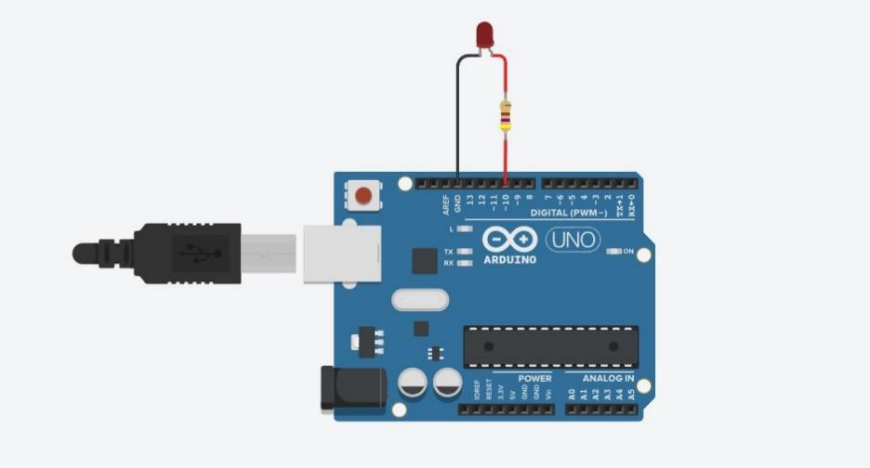
Použité součástky

- Arduino UNO R3
- LED
- Rezistor

Postup

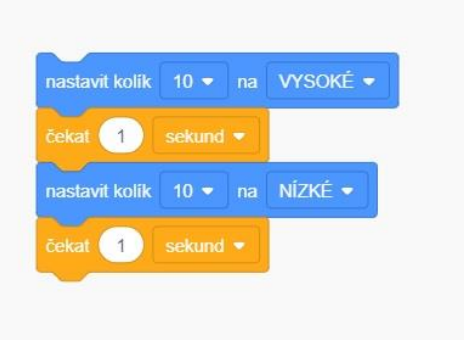
1. Pedagog vysvětlí princip činnosti desky Arduino, popíše její jednotlivé části, popíše jednotlivé vstupy a výstupy (piny).
2. Pedagog vysvětlí význam logické jedničky a nuly.
3. Pedagog vysvětlí princip činnosti LED a rezistoru. Vysvětlí, proč je důležité zapojit rezistor do série s diodou. Zdůrazní, proč je velikost rezistoru rozhodující. Také zdůrazní nutnost dodržení polarity LED.
4. V poslední fázi výkladu pedagog ukáže a na příkladech vysvětlí základní algoritmické konstrukce.
5. Žáci vytvoří v Tinkercadu nový obvod, který zapojí dle přiloženého schématu. V tomto bodě je důležité zdůraznit zvýšenou pozornost při zapojování. Pokud žáci zapojí některou součástku do špatného výstupu desky (pinu) nebo zamění polaritu součástky, tak ani správně naprogramovaný obvod nemusí fungovat korektně.
6. Žáci nastaví odpor rezistoru na 470 Ω
7. Pedagog v tuto chvíli ukáže možnosti programování desky. Vysvětlí, že existuje možnost jak blokového, tak textového programování. Názorně ukáže, jak lze mezi jednotlivými způsoby přepínat.
8. Žáci naprogramují obvod tak, aby LED blikala v pravidelných intervalech 1 sekunda.
9. Pedagog společně se studenty zkontroluje a vyhodnotí správnou funkci programu.

Schéma zapojení



Obrázek 18: Arduino, schéma 1

Zdrojový kód



Obrázek 20: Arduino, bloky 1

```
// C++ code
//
void setup()
{
  pinMode(10, OUTPUT);
}

void loop()
{
  digitalWrite(10, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(10, LOW);
  delay(1000); // Wait for 1000 millisecond(s)
}
```

Obrázek 19: Arduino, program 1

1. Blok způsobí rozsvícení diody.
2. Blok znamená setrvání v tomto stavu po dobu 1 sekundy.
3. Blok způsobí zhasnutí diody.
4. Blok znamená setrvání v tomto stavu po dobu 1 sekundy.

Poznámky

- Žáci mohou přepnout blokově napsaný program do textové podoby a vidět tak svůj výsledek práce v opravdovém programovacím jazyce. Případní zájemci mohou zkoušet textově napsaný program upravovat. Tato možnost se však hodí spíše pro pokročilejší žáky.
- Žáci mají jako nápovědu k dispozici na pracovním listě seznam použitých bloků.
- Žáci mohou použít jiné piny pro připojení LED, avšak tuto skutečnost je nutno zohlednit v programu.

3.1.2 Lekce 2

Název

Spínání LED

Cíl

Cílem lekce je seznámit žáky s funkcí digitálních vstupů a procvičit větvení programu. Výsledkem bude jednoduché zapojení, kde pomocí tlačítka bude přiváděna logická jednička nebo nula na digitální vstup. Díky vytvořenému programu bude zajištěn svit diody po dobu stisku tlačítka.

Časová náročnost

30 minut

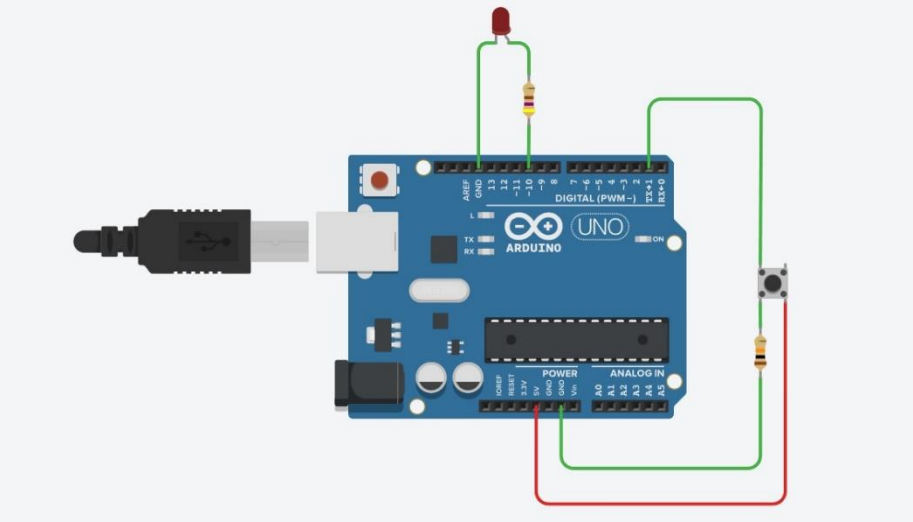
Použité součástky

- Arduino UNO R3
- LED
- 2 x rezistor
- Tlačítko

Postup

1. Pedagog vysvětlí funkci digitálních vstupů a výstupů.
2. Pedagog vysvětlí funkci tlačítka.
3. Žáci zapojí obvod dle přiloženého schématu. Mohou vytvořit zcela nový obvod nebo navázat na obvod z předcházející lekce. Oproti předchozího zapojení přibylo tlačítko s dalším rezistorem.
4. Žáci nastaví odpor rezistoru před LED na 470 Ω.
5. Žáci nastaví odpor rezistoru před tlačítkem na 10 kΩ.
6. Žáci naprogramují obvod tak, aby po dobu stisku tlačítka svítila LED.
7. Pedagog společně se studenty zkontroluje a vyhodnotí správnou funkci programu.

Schéma zapojení



Obrázek 21: Arduino, schéma 2

Zdrojový kód

Obrázek 23: Arduino, bloky 2

```
// C++ code
void setup()
{
  pinMode(1, INPUT);
  pinMode(10, OUTPUT);
}

void loop()
{
  if (digitalRead(1) == 1) {
    analogWrite(10, 1);
  } else {
    analogWrite(10, 0);
  }
  delay(10); // Delay a little bit to improve simulation performance
}
```

Obrázek 22: Arduino, program 2

1. Blok představuje podmínku. Pokud se na digitálním kolíku č. 1 objeví logická 1 (při zmáčknutí tlačítka) vykoná se blok č. 2 a pokud se objeví logická 0, vykoná se blok č. 3.
2. Blok způsobí rozsvícení diody.
3. Blok způsobí zhasnutí diody.

Poznámky

- Žáci mohou přepnout blokově napsaný program do textové podoby a vidět tak svůj výsledek práce v opravdovém programovacím jazyce. Případní zájemci mohou zkoušet textově napsaný program upravovat. Tato možnost se však hodí spíše pro pokročilejší žáky.
- Žáci mohou použít jiné piny pro LED i pro tlačítko, avšak tuto skutečnost je nutno zohlednit v programu.
- Žáci mají jako nápovědu k dispozici na pracovním listě seznam použitých bloků.

3.1.3 Lekce 3

Název

Blikání s více LED

Cíl

Cílem lekce je procvičit práci s více digitálními výstupy a procvičit cykly v programu. Výsledkem bude zapojení čtyř LED, které budou postupně blikat. První dioda (úplně vlevo) blikne jednou, druhá dioda dvakrát, třetí třikrát a poslední blikne čtyřikrát po sobě.

Časová náročnost

30 minut

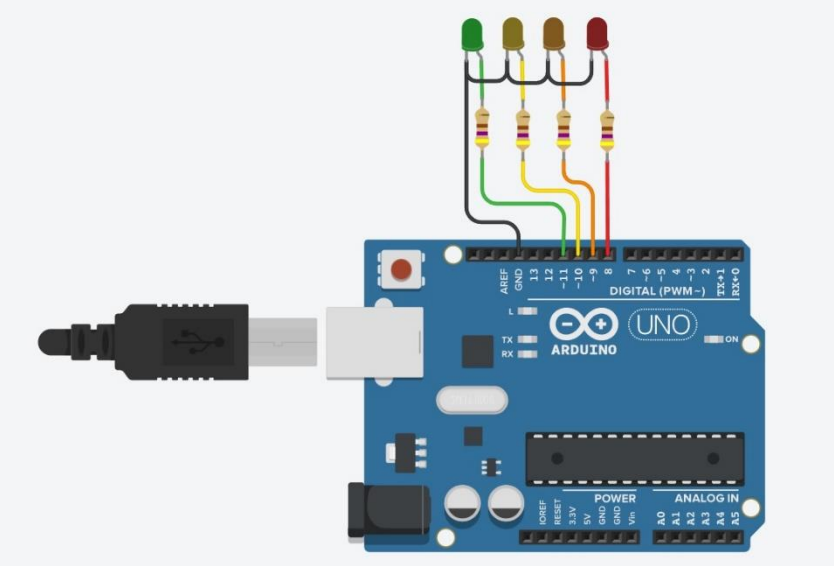
Použité součástky

- Arduino UNO R3
- 4x rezistor
- 4 x LED

Postup

1. Pedagog zopakuje funkci základní algoritmické konstrukce, cyklu.
2. Žáci vytvoří nový obvod a zapojí jej dle přiloženého schématu.
3. Žáci nastaví velikost rezistorů na 470 Ω.
4. Žáci nastaví barvy LED dle schématu.
5. Žáci naprogramují obvod tak, aby diody blikali postupně z leva doprava. První dioda blikne jednou a počet bliknutí se bude s každou další diodou zvyšovat o jedno. Interval blikání je 1 sekunda.
6. Pedagog společně se studenty zkontroluje a vyhodnotí správnou funkci programu.

Schéma zapojení



Obrázek 24: Arduino, schéma 3

Zdrojový kód

Obrázek 25: Arduino, bloky 3

```
// C++ code
//
int counter;

int counter2;

int counter3;

void setup()
{
  pinMode(11, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(8, OUTPUT);
}

void loop()
{
  digitalWrite(11, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(11, LOW);
  delay(1000); // Wait for 1000 millisecond(s)
  for (counter = 0; counter < 2; ++counter) {
    digitalWrite(10, HIGH);
    delay(1000); // Wait for 1000 millisecond(s)
    digitalWrite(10, LOW);
    delay(1000); // Wait for 1000 millisecond(s)
  }
  for (counter2 = 0; counter2 < 3; ++counter2) {
    digitalWrite(9, HIGH);
    delay(1000); // Wait for 1000 millisecond(s)
    digitalWrite(9, LOW);
    delay(1000); // Wait for 1000 millisecond(s)
  }
  for (counter3 = 0; counter3 < 4; ++counter3) {
    digitalWrite(8, HIGH);
    delay(1000); // Wait for 1000 millisecond(s)
    digitalWrite(8, LOW);
    delay(1000); // Wait for 1000 millisecond(s)
  }
}
```

Obrázek 26: Arduino, program 3

1. Blok rozsvítí zelenou diodu.
2. Blok znamená setrvání v tomto stavu po dobu 1 sekundy.
3. Blok zhasne zelenou diodu.
4. Blok znamená setrvání v tomto stavu po dobu 1 sekundy.
5. Blok představuje cyklus, který se zopakuje dvakrát (uvnitř cyklu se nachází blok č. 6, 7, 8, 9).
6. Blok rozsvítí žlutou diodu.
7. Blok znamená setrvání v tomto stavu po dobu 1 sekundy.
8. Blok zhasne žlutou diodu.
9. Blok znamená setrvání v tomto stavu po dobu 1 sekundy.
10. Blok představuje cyklus, který se zopakuje třikrát (uvnitř cyklu se nachází blok č. 11, 12, 13, 14).
11. Blok rozsvítí oranžovou diodu.
12. Blok znamená setrvání v tomto stavu po dobu 1 sekundy.
13. Blok zhasne oranžovou diodu.
14. Blok znamená setrvání v tomto stavu po dobu 1 sekundy.
15. Blok představuje cyklus, který se zopakuje čtyřikrát (uvnitř cyklu se nachází blok č. 16, 17, 18, 19).
16. Blok rozsvítí červenou diodu.
17. Blok znamená setrvání v tomto stavu po dobu 1 sekundy.
18. Blok zhasne červenou diodu.
19. Blok znamená setrvání v tomto stavu po dobu 1 sekundy.

Poznámky

- Žáci mohou přepnout blokově napsaný program do textové podoby a vidět tak svůj výsledek práce v opravdovém programovacím jazyce. Případní zájemci mohou zkoušet textově napsaný program upravovat. Tato možnost se však hodí spíše pro pokročilejší žáky.
- Žáci mohou použít jiné piny pro LED, avšak tuto skutečnost je nutno zohlednit v programu.
- Barvy jednotlivých diod nejsou podstatné, a tak je mohou žáci zvolit dle svého uvážení.
- Žáci mají jako nápovědu k dispozici na pracovním listě seznam použitých bloků.

3.1.4 Lekce 4

Název

Regulace jasu LED

Cíl

Cílem je seznámit žáky s analogovými vstupy a převodem analogové hodnoty na digitální a procvičit práci s proměnnými. Výsledkem bude zapojení LED a potenciometru. Při otáčení potenciometru se bude měnit jas LED. Když bude potenciometr natočen úplně vlevo, dioda nebude svítit. S postupným otáčením vpravo se jas bude zvyšovat a na konci otáčení bude LED svítit plným jasem.

Časová náročnost

30 minut

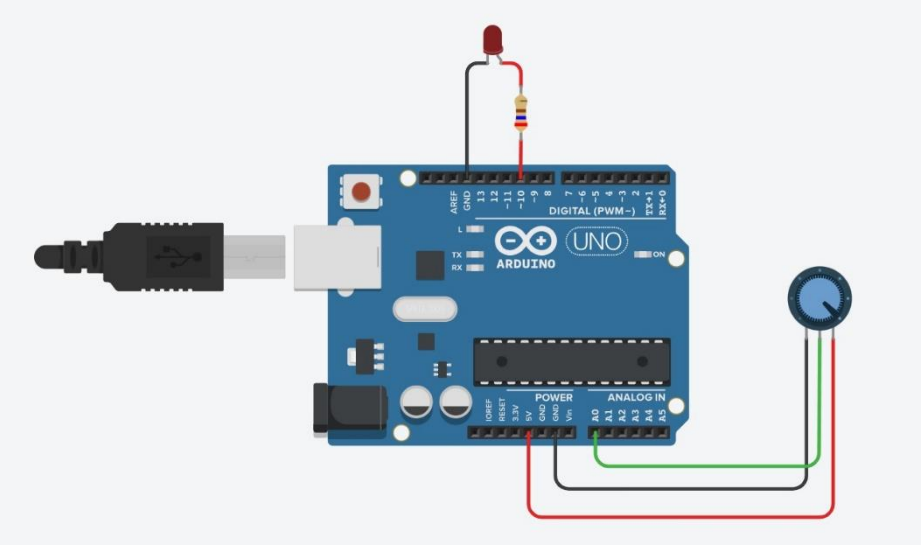
Použité součástky

- Arduino UNO R3
- LED
- Rezistor
- Potenciometr

Postup

1. Pedagog vysvětlí, co je to proměnná a vysvětlí její význam v programu.
2. Pedagog názorně ukáže tvorbu vlastní proměnné. Žáci si následně tvorbu vlastní proměnné sami vyzkouší. Také si vyzkouší uložení hodnoty do proměnné.
3. Pedagog seznámí žáky s funkcí a zapojení potenciometru.
4. Pedagog seznámí žáky s převodem analogové hodnoty na digitální.
5. Žáci vytvoří nový obvod a zapojí jej dle přiloženého schématu.
6. Žáci nastaví velikost odporu rezistoru na 260 Ω.
7. Žáci nastaví velikost potenciometru na 10 kΩ.
8. Žáci naprogramují obvod tak, aby se při otáčení proměnného odporu měnil jas LED. Když bude potenciometr natočen úplně vlevo, dioda nebude svítit. S postupným otáčením vpravo se jas bude zvyšovat a na konci otáčení bude LED svítit plným jasem.
9. Pedagog společně se studenty zkontroluje a vyhodnotí správnou funkci obvodu a programu.

Schéma zapojení



Obrázek 27: Arduino, schéma 4

Zdrojový kód

Obrázek 29: Arduino, bloky 4

```
// C++ code
//
int prevod = 0;

void setup()
{
  pinMode(A0, INPUT);
  pinMode(10, OUTPUT);
}

void loop()
{
  prevod = (analogRead(A0) / 4);
  analogWrite(10, prevod);
  delay(10); // Delay a little bit to improve simulation performance
}
```

Obrázek 28: Arduino, program 4

1. Blok nastaví proměnnou převod na čtení analogového kolíku A0 a přečtenou hodnotu vydělí čtyřmi.
2. Blok nastaví diodu na hodnotu proměnné převod. Měnící se hodnota (hodnota se mění při otáčení potenciometru) bude způsobovat měnící se jas.

Poznámky

- Žáci mohou přepnout blokově napsaný program do textové podoby a vidět tak svůj výsledek práce v opravdovém programovacím jazyce. Případní zájemci mohou zkoušet textově napsaný program upravovat. Tato možnost se však hodí spíše pro pokročilejší žáky.
- Žáci mohou použít jiné piny pro LED i pro proměnný rezistor, avšak tuto skutečnost je nutno zohlednit v programu.
- Žáci mají jako nápovědu k dispozici na pracovním listě seznam použitých bloků.

3.1.5 Lekce 5 – bonusová úloha

V této lekci budou žáci pracovat s více proměnnými a program se tak stává složitějším. Právě pro svou náročnost je úloha označena jako bonusová. Je určena především pro pokročilejší zájemce, případně pro kroužky programování, robotiky apod.

Název

Zaznamenání stavu tlačítka

Cíl

Cílem je procvičit práci s více proměnnými, práci s více podmínkami a operátory. Dalším cílem je naučit žáky pracovat s tzv. sériovým monitorem, který slouží jako stavový řádek, kde lze vypisovat například aktuální hodnoty proměnných.

Časová náročnost

45 minut

Použité součástky

- Arduino UNO R3
- LED
- 2 x Rezistor
- Tlačítko

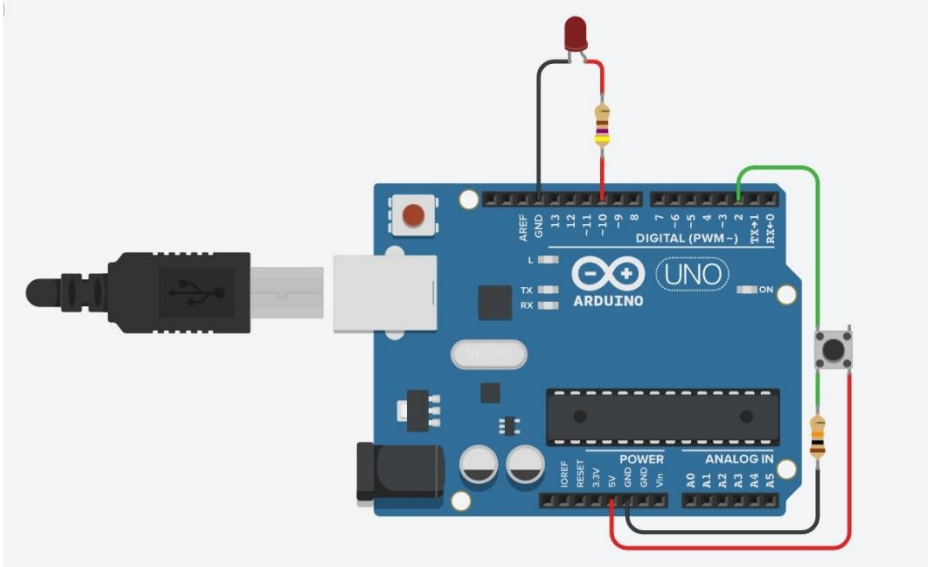
Postup

1. Pedagog zopakuje význam základních algoritmických konstrukcí a proměnných v programu.
2. Žáci vytvoří nový obvod a zapojí jej dle přiloženého schématu, které je stejné jako u lekce 2.
3. Žáci nastaví velikost odporu rezistoru před LED na 470 Ω.
4. Žáci nastaví velikost odporu rezistoru před tlačítkem na 10 kΩ.
5. Žáci naprogramují obvod tak, aby každý stisk tlačítka byl zaznamenán do proměnné pocitadlo. Na začátku je stav tlačítka nastaven na hodnotu 0 a s každým dalším stiskem tlačítka se hodnota zvýší o 1. Jestli je tlačítko aktuálně stisknuto či nikoliv se zaznamenává do proměnné stav_tlacitka. Pokud je počet stisknutí větší než 3, proměnná pocitadlo se vynuluje. Počítadlo tak může nabývat čtyř hodnot (0-3) a s každou hodnotou dojde k nějaké akci:
    a) 0 – LED nebude svítit
    b) 1 – LED se rozsvítí

c) 2 – LED opět zhasne
d) 3 – LED začne blikat v intervalu 0,5 sekund

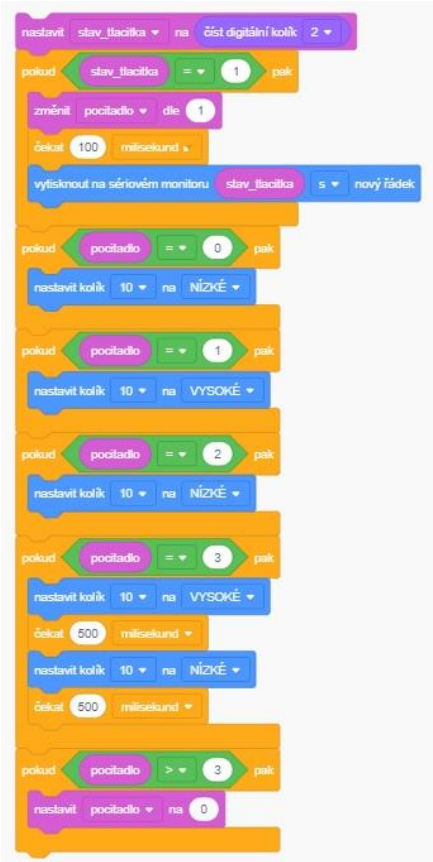
6. Pedagog společně se studenty zkontroluje a vyhodnotí správnou funkci obvodu a programu.

Schéma zapojení



Obrázek 30: Arduino, schéma 5

Zdrojový kód



Obrázek 32: Arduino, bloky 5

```
// C++ code
//
int stav_tlacitka = 0;

int pocitadlo = 0;

void setup()
{
  pinMode(2, INPUT);
  Serial.begin(9600);

  pinMode(10, OUTPUT);
}

void loop()
{
  stav_tlacitka = digitalRead(2);
  if (stav_tlacitka == 1) {
    pocitadlo += 1;
    delay(100); // Wait for 100 millisecond(s)
    Serial.println(stav_tlacitka);
  }
  if (pocitadlo == 0) {
    digitalWrite(10, LOW);
  }
  if (pocitadlo == 1) {
    digitalWrite(10, HIGH);
  }
  if (pocitadlo == 2) {
    digitalWrite(10, LOW);
  }
  if (pocitadlo == 3) {
    digitalWrite(10, HIGH);
    delay(500); // Wait for 500 millisecond(s)
    digitalWrite(10, LOW);
    delay(500); // Wait for 500 millisecond(s)
  }
  if (pocitadlo > 3) {
    pocitadlo = 0;
  }
}
```

Obrázek 31: Arduino, program 5

1. Blok nastaví proměnnou stav_tlacitka jako čtení digitálního kolíku č. 2
2. Blok představuje podmínku, která se vykoná, jestliže proměnná stav_tlacitka se bude rovnat logické jedničce (tak se stane, pokud bude stisknuto tlačítko). Pokud bude podmínka splněna, vykonají se bloky č. 3, 4, 5.
3. Blok zvětší hodnotu proměnné pocitadlo o hodnotu 1.
4. Blok znamená setrvání v tomto stavu po dobu 100 ms (kvůli správné funkci tlačítka).
5. Blok vypíše na sériovém monitoru hodnotu proměnné stav tlačítka (zjistíme tak, jestli je tlačítko stisknuto či nikoliv).
6. Blok je podmínka, která se vykoná, jestliže se proměnná pocitadlo rovná hodnotě 0. Pokud bude podmínka splněna, vykoná se blok č. 7.
7. Blok zhasne diodu.
8. Blok je podmínka, která se vykoná, jestliže se proměnná pocitadlo rovná hodnotě 1. Pokud bude podmínka splněna, vykoná se blok č. 9.
9. Blok rozsvítí diodu.
10. Blok je podmínka, která se vykoná, jestliže se proměnná pocitadlo rovná hodnotě 2. Pokud bude podmínka splněna, vykoná se blok č. 11.

11. Blok zhasne diodu.
12. Blok je podmínka, která se vykoná, jestliže se proměnná pocitadlo rovná hodnotě 3. Pokud bude podmínka splněna, vykoná se blok č. 13, 14, 15, 16.
13. Blok rozsvítí diodu.
14. Blok znamená setrvání v tomto stavu po dobu 500 ms.
15. Blok zhasne diodu.
16. Blok znamená setrvání v tomto stavu po dobu 500 ms.
17. Blok je podmínka, která se vykoná, jestliže se proměnná pocitadlo je větší než 3. Pokud bude podmínka splněna, vykoná se blok č. 18.
18. Blok vynuluje proměnnou pocitadlo.

Poznámky

- Žáci mohou přepnout blokově napsaný program do textové podoby a vidět tak svůj výsledek práce v opravdovém programovacím jazyce. Případní zájemci mohou zkoušet textově napsaný program upravovat. Tato možnost se však hodí spíše pro pokročilejší žáky.
- Žáci mají jako nápovědu k dispozici na pracovním listě seznam použitých bloků.
- Žáci mohou použít jiné piny pro LED i pro tlačítko, avšak tuto skutečnost je nutno zohlednit v programu.
- Všechny proměnné lze pojmenovat libovolně. V případě vlastního pojmenování je potřeba tuto skutečnost zohlednit v celém programu.
- Na sériovém monitoru lze vypisovat také hodnoty jiných proměnných. Například aktuální hodnotu proměnné pocitadlo.

3.2 Nedigitální aktivity

Následující lekce jsou zaměřeny na rozvoj algoritmizace a algoritmického myšlení bez použití digitálních pomůcek. Na rozdíl od digitálních aktivit jsou lekce vhodné především pro prezenční výuku či samostatné procvičování. K realizaci aktivit je zapotřebí pouze pracovní list a kartičky s příkazy. Žádné další pomůcky nejsou potřeba.

Cílem aktivit je dostat postavičku Rudolfa do cíle. Rudolf a jeho cíl se nachází na trojúhelníkové síti. Postavička se může pohybovat pouze po stranách trojúhelníka a nesmí procházet skrze jeho vrcholy. K úspěšnému přesunu postavičky do cíle je zapotřebí sestavit algoritmus, složený ze základních algoritmických konstrukcí v podobě papírových kartiček, které do sebe musí zapadat. První čtyři lekce jsou zaměřeny na sestavení algoritmu. Poslední dvě lekce se zaměřují na porozumění již sestaveným algoritmům.

Na rozdíl od digitálních aktivit nemusí vyučující disponovat vědomostmi z jiných oborů. Stačí pouze základní znalosti z oblasti algoritmizace.

3.2.1 Lekce 1

Název

Seznámení s Rudolfem

Cíl

Cílem lekce je procvičit práci s jednoduchými příkazy a porozumět pohybu postavičky po trojúhelníkové síti. Výsledkem bude jednoduchý algoritmus, který dovede postavičku Rudolfa do jeho cíle (postavička dojde na trojúhelník s jablkem).

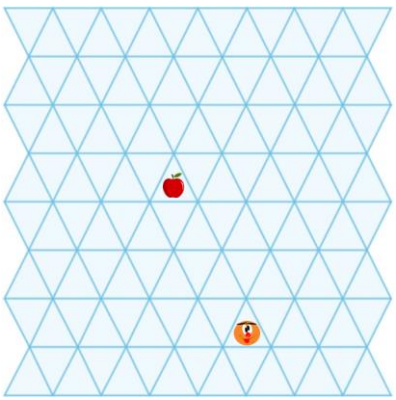
Časová náročnost

15 minut

Postup

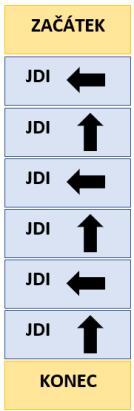
1. Pedagog seznámí žáky s cílem aktivity.
2. Pedagog seznámí žáky s pravidly pohybu po trojúhelníkové síti.
3. Pedagog seznámí žáky s pravidly skládání algoritmu pomocí kartiček.
4. Žáci obdrží pracovní listy a kartičky s příkazy.
5. Žáci samostatně sestaví algoritmus složený z jednoduchých směrových příkazů v podobě papírových kartiček tak, aby postavička Rudolf došla úspěšně do cíle (na trojúhelník s jablkem).
6. Žáci společně s pedagogem vyhodnotí správnost řešení.
7. Diskuse se žáky.

Zadání



Obrázek 34: Rudolf, síť 1

Řešení



Obrázek 33: Rudolf, bloky 1

Poznámky

- Lekci lze rozšířit tak, že žáci dostanou další směrové příkazy a budou hledat jiné možné řešení problému.

3.2.2 Lekce 2

Název

Opakování s Rudolfem

Cíl

Cílem je procvičit další algoritmickou konstrukci – opakování. S využitím opakování budou žáci zjednodušovat algoritmus z předchozí lekce. Výsledek bude totožný jako v minulé lekci, ale postup řešení se značně zjednoduší.

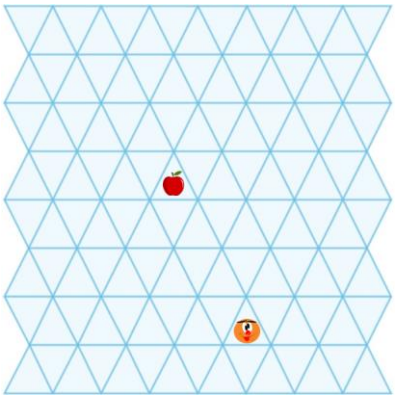
Časová náročnost

15 minut

Postup

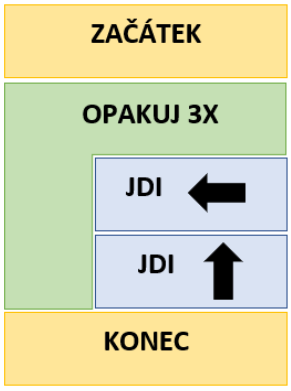
1. Pedagog seznámí žáky s cílem aktivity.
2. Pedagog vysvětlí význam a využití cyklu jako algoritmické konstrukce.
3. Žáci obdrží pracovní listy a kartičky s příkazy.
4. Žáci samostatně sestaví algoritmus složený z jednoduchých směrových příkazů a cyklu v podobě papírových kartiček tak, aby postavička Rudolf došla úspěšně do cíle (na trojúhelník s jablkem).
5. Žáci společně s pedagogem vyhodnotí správnost řešení.
6. Diskuse se žáky.

Zadání



Obrázek 36: Rudolf, síť 2

Řešení



Obrázek 35: Rudolf, bloky 2

Poznámky

- Lekci lze rozšířit tak, že žáci dostanou další směrové příkazy a cyklus, u kterého nastaví počet opakování a budou tak hledat jiné možné řešení problému.

3.2.3 Lekce 3

Název

Podmínky s Rudolfem

Cíl

Cílem je procvičit další algoritmické konstrukce – podmínky. S využitím příkazů a podmínek se budou žáci snažit dostat postavičku Rudolfa k cíli přes překážku (postavička dojde na trojúhelník s jablkem).

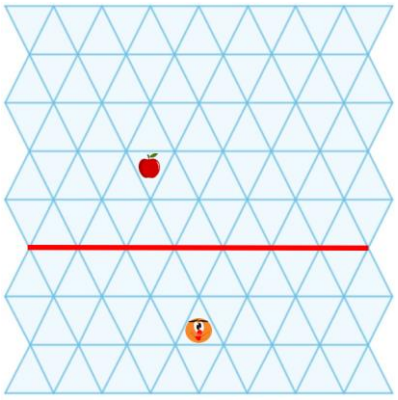
Časová náročnost

15 minut

Postup

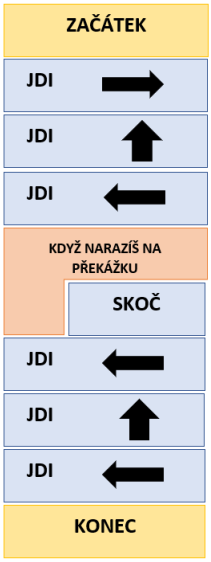
1. Pedagog seznámí žáky s cílem aktivity.
2. Pedagog vysvětlí význam a využití podmínky jako algoritmické konstrukce.
3. Žáci obdrží pracovní listy a kartičky s příkazy.
4. Žáci samostatně sestaví algoritmus složený z jednoduchých směrových příkazů a podmínky v podobě papírových kartiček tak, aby postavička Rudolf došla přes překážku úspěšně do cíle (na trojúhelník s jablkem).
5. Žáci společně s pedagogem vyhodnotí správnost řešení.
6. Diskuse se žáky.

Zadání



Obrázek 37: Rudolf, síť 3

Řešení



Obrázek 38: Rudolf, bloky 3

Poznámky

- Lekci lze rozšířit tak, že žáci dostanou další směrové příkazy a podmínku a budou hledat jiné možné řešení problému.

3.2.4 Lekce 4

Název

Přes překážky k jablku

Cíl

Cílem lekce je zopakovat algoritmické konstrukce – příkazy, podmínky a cykly v jednom algoritmu. S využitím příkazů, podmínky a cyklu se budou žáci snažit dostat postavičku Rudolfa k cíli přes překážky (postavička dojde na trojúhelník s jablkem)

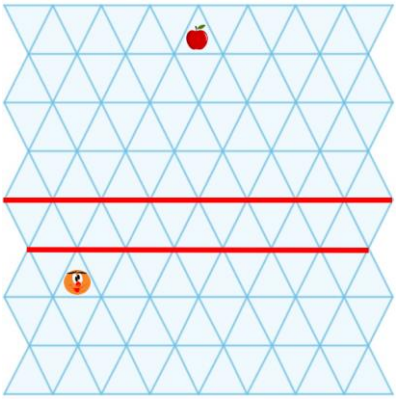
Časová náročnost

30 minut

Postup

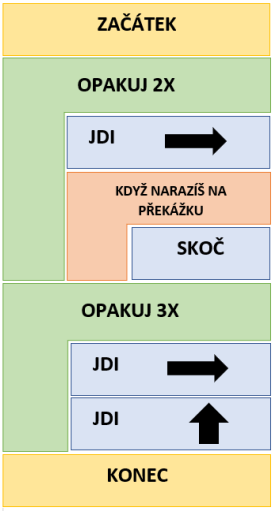
1. Pedagog seznámí žáky s cílem aktivity.
2. Pedagog zopakuje základní algoritmické konstrukce (příkazy, cykly, podmínky).
3. Žáci obdrží pracovní listy a kartičky s příkazy.
4. Žáci samostatně sestaví algoritmus složený z jednoduchých směrových příkazů, podmínky a cyklu v podobě papírových kartiček tak, aby postavička Rudolf došla přes překážky úspěšně do cíle (na trojúhelník s jablkem).
5. Žáci společně s pedagogem vyhodnotí správnost řešení.
6. Diskuse se žáky.

Zadání



Obrázek 40: Rudolf, síť 4

Řešení



Obrázek 39: Rudolf, bloky 4

Poznámky

- Lekci lze rozšířit tak, že žáci dostanou další směrové příkazy, podmínku a cykly, kde nastaví daný počet opakování. Budou tak hledat jiné možné řešení problému.

3.2.5 Lekce 5

Název

Jaké ovoce Rudolf získá?

Cíl

Cílem lekce je porozumět již hotovému algoritmu a zopakovat základní algoritmické konstrukce. Žáci budou mít za úkol z již napsaného algoritmu zjistit, do jakého cíle se postavička Rudolf dostane.

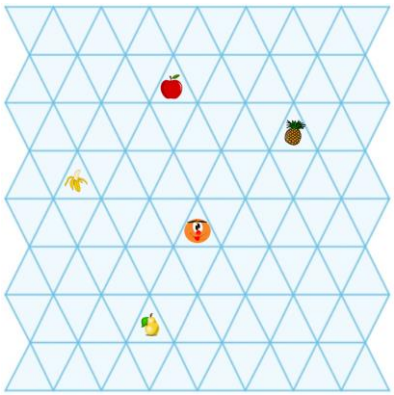
Časová náročnost

15 minut

Postup

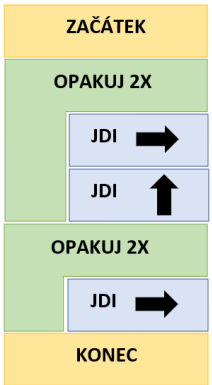
1. Pedagog seznámí žáky s cílem aktivity.
2. Pedagog zopakuje základní algoritmické konstrukce (příkazy, cykly, podmínky).
3. Žáci obdrží pracovní listy.
4. Žáci z již napsaného algoritmu v podobě bloků zjistí, k jakému cíli dojde postavička Rudolf.
5. Žáci společně s pedagogem vyhodnotí správnost řešení.
6. Diskuse se žáky.

Zadání



Obrázek 42: Rudolf, síť 5

Řešení



Obrázek 41: Rudolf, bloky 5

Podle algoritmu dojde postavička Rudolf do cíle v podobě trojúhelníku s ananasem.

Poznámky

- Lekci lze rozšířit tak, že žáci dostanou hotové algoritmy pro všechny dostupné cíle a budou mít za úkol určit, který algoritmus náleží příslušnému cíli.

3.2.6 Lekce 6

Název

Které ovoce Rudolf získá tentokrát?

Cíl

Cílem lekce je porozumět již hotovému algoritmu a zopakovat základní algoritmické konstrukce. Žáci budou mít za úkol z již napsaného algoritmu zjistit, do jakého cíle se postavička Rudolf dostane.

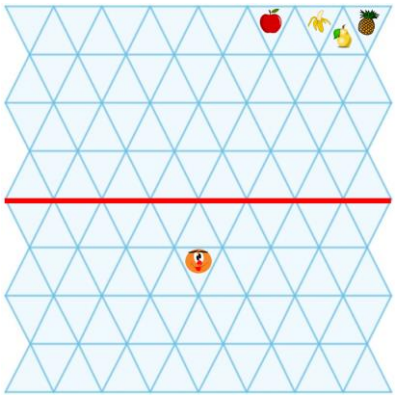
Časová náročnost

20 minut

Postup

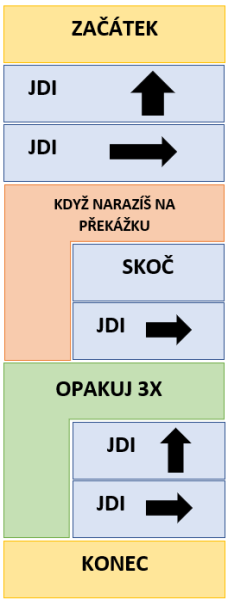
1. Pedagog seznámí žáky s cílem aktivity.
2. Pedagog zopakuje základní algoritmické konstrukce (příkazy, cykly, podmínky).
3. Žáci obdrží pracovní listy.
4. Žáci z již napsaného algoritmu v podobě bloků zjistí, k jakému cíli dojde postavička Rudolf.
5. Žáci společně s pedagogem vyhodnotí správnost řešení.
6. Diskuse se žáky.

Zadání



Obrázek 43: Rudolf, síť 6

Řešení



Obrázek 44: Rudolf, bloky 6

Podle algoritmu dojde postavička Rudolf do cíle v podobě trojúhelníku s banánem.

Poznámky

- Lekci lze rozšířit tak, že žáci dostanou hotové algoritmy pro všechny dostupné cíle a budou mít za úkol určit, který algoritmus náleží příslušnému cíli
- Oproti předcházející lekci je algoritmus ztížen, jelikož cíle se nacházejí blíže u sebe. Není tedy na první pohled zcela jasné, kam postavička dojde.

ZÁVĚR

Bakalářská práce je zaměřena na rozvoj algoritmizace u žáků druhého stupně základních škol v České republice. Cílem je popsat možnosti rozvoje algoritmizace a následně navrhnout konkrétní digitální a nedigitální aktivity, díky kterým mohou žáci rozvinout své schopnosti v oblasti tvorby algoritmů.

První kapitola teoretické části se věnuje problematice algoritmů v obecné rovině. Jsou zde popsány základní pojmy, které tvoří nezbytný teoretický základ pro pochopení celé problematiky. V druhé půli je pozornost upřena na výčet konkrétních aktivit. Z nepřeberného množství dostupných, jsou vyjmenovány pouze ty v praxi nejpoužívanější. Některé z nich jsou pouze digitální a jiné mají podobu fyzické učební pomůcky. Na základě druhé kapitoly teoretické části lze konstatovat, že první z cílů práce byl splněn.

Praktická část se skládá ze dvou podkapitol. První z nich obsahuje návrh digitálních aktivit. Výchozím prostředím je webová aplikace www.tinkercad.com. Jako nástroj zde byla zvolena výuková pomůcka Arduino, kterou lze blokově programovat. V tomto duchu bylo navrhnuto celkem 5 lekcí, z čehož 4 jsou určeny všem žákům. Poslední je vzhledem k vyšší obtížnosti označena jako bonusová a je vhodná například pro pokročilejší žáky. Druhá podkapitola praktické části je zaměřena na tvorbu aktivit nedigitálních. Na trojúhelníkové síti stojí postavička Rudolf a její cíl v podobě ovoce. Pomocí papírových kartiček s příkazy skládají žáci algoritmus, díky kterému se Rudolf dostane do svého cíle. Celkem bylo navrženo 6 lekcí. Tímto byl splněn i druhý cíl práce. A tedy návrh digitálních a nedigitálních aktivit.

Vzhledem k nově revidovanému Rámcovému vzdělávacímu programu pro základní vzdělávání, mohou vytvořené aktivity posloužit jako náplň výuky ve vzdělávací oblasti Informatika. Konkrétně v tematickém celku algoritmizace a programování. V tomto smyslu představuje obsah praktické části potencionální přínos pro praktické využití ve výuce.

SEZNAM POUŽITÉ LITERATURY

Blahuta, J. (2017). *Algoritmizace: studijní opora pro kombinované studium*. Moravská vysoká škola Olomouc.

Dohnal, P. (2009). *Programování na základních školách* (Diplomová práce). Masarykova univerzita, Pedagogická fakulta.

Futschek, G. (2006). Algorithmic Thinking: The Key for Understanding Computer Science. *Informatics Education – The Bridge between Using and Understanding Computers*, 159–168. https://doi.org/10.1007/11915355_15

Jakeš, T. š., Baťko, J., & Simbartl, P. (2020). *Učebnice LEGO robotiky*. Učebnice LEGO Robotiky. https://lego.zcu.cz/ucebnice/

Kostolányová, K. (2002). *Algoritmizace a řešení problémů*. Ostravská univerzita, Pedagogická fakulta

LEGO® MINDSTORMS® – Informace. (n.d.). LEGO® MINDSTORMS®. https://www.lego.com/cs-cz/themes/mindstorms/about

Ozobot | Robots to code and create with. (2022). Ozobot. https://ozobot.com/

Pšenčíková, J. (2009). *Algoritmizace*. Van Duuren Media.

Scottie Go! Gry do nauki programowania dla dzieci. (2021, October 15). *Scottie Go! - Scottie Go! Gry do nauki programowania dla dzieci.* Scottie Go! Gry do nauki programowania dla dzieci. - Poznaj Scottie Go! - edukacyjne gry do nauki programowania w technologii AR! Układaj komendy z bloczków i skanuj je telefonem lub tabletem! https://scottiego.com

Skalka, J., Cápay, M., Lovászová, G., Masárošová, M., & Palmárová, V. (2007). *Algoritmizácia a úvod do programovania*. Univerzita Konštantína Filozofa.

Šarmanová, J. (2014). *Algoritmizace a řešení problémů*. Ostravská univerzita, Pedagogická fakulta.

Štefan R. (2006): *Úvod do algoritmizace a programování*. OA Ostrava-Poruba.

T. (2016, March 19). *How to Explain Algorithms to Kids*. Tynker Blog. https://www.tynker.com/blog/articles/ideas-and-tips/how-to-explain-algorithms-to-kids/

Tichavová, B. (2018). *Co je IM*. Informatické myšlení. https://imysleni.cz/informaticke-mysleni/co-je-informaticke-mysleni

Umíme informatiku. (n.d.). Umíme informatiku. https://www.umimeinformatiku.cz/

Umíme to - Online procvičování školního učiva. (n.d.). Umíme to. https://www.umimeto.org/

Voda, Z. (2017). *Průvodce světem Arduina*. Martin Stříž.

SEZNAM POUŽITÝCH SYMBOLŮ

ZŠ	Základní škola
3D	Trojrozměrný
USB	Univerzální sériová sběrnice
IDE	Integrované vývojové prostředí
OS	Operační systém
LED	Světlo emitující dioda

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH