**ALGORITMUS**

Algoritmus je základný elementárny pojem informatiky. Je to konečná postupnosť dobre definovaných inštrukcií na splnenie určitej úlohy. Ide o prepis, návod, realizáciou ktorého získame zo zadaných vstupných údajov požadované výsledky. V stručnosti sa dá povedať, že algoritmus je presný návod k zvládnutiu určitej činnosti.

|  |  |  |  |
| --- | --- | --- | --- |
| NÁRODNÁ NORMA | STN | STN 9030 | www.sutn.sk |

Vlastnosti

1.*Determinovanosť*: Realizácia algoritmu nesmie byť podmienená inými podmienkami ako tými, ktoré sú v ňom uvedené, nesmie byť závislá od riešiteľa a prostredia, v ktorom sa algoritmus realizuje. Je to proces, ktorý môže byť kedykoľvek a *kýmkoľvek* *opakovaný s rovnakým výsledkom*.

2.*Hromadnosť*: Algoritmus neslúži na riešenie jednej konkrétnej úlohy, ale na *riešenie celej triedy úloh*. Je použiteľný na ľubovoľné vstupné údaje spĺňajúce požadované podmienky.

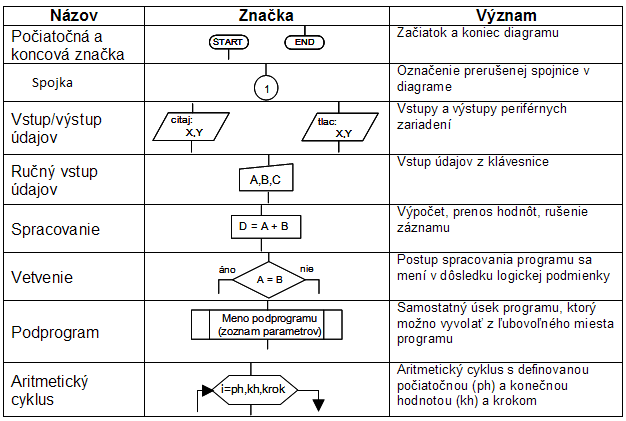
3.*Rezultatívnosť*: Transformácia predpísaná algoritmom *dá po konečnom počte krokov výsledok* t.j. vyžaduje, aby sa postup, použitý na riešenie ľubovoľnej úlohy danej triedy, po konečnom počte krokov zastavil.

4.*Efektivita*: Všeobecne požadujeme, aby *každá operácia požadovaná algoritmom bola dostatočne jednoduchá* na to, aby mohla byť aspoň v princípe prevedená v konečnom čase iba s použitím ceruzky a papiera.

5.*Všeobecnosť*: Algoritmus *nerieši jeden konkrétny problém* (napr. „ako vypočítať 3×7“), ale rieši všeobecnú triedu obdobných problémov (napr. „ako vypočítať súčin dvoch celých čísel“).

6.*Postupnosť*: Kroky algoritmu sa *vykonávajú postupne za sebou*, ak nie je explicitne dané iné poradie.

**VÝVOJOVÝ DIAGRAM**



**PREMENNÉ**

Premenné majú typ podľa druhu dát, ktoré do nich zapisujeme. Premenná je objekt, ktorý má určitý typ, má priradené meno a hodnotu, kt. môže v priebehu operácie zmeniť.

Typy premenných:

*-celé číslo (INT)*

*-desatinné číslo (FLOAT, DOUBLE)*

*-znak (CHAR)*

*-text (STRING)*

*-boolean (BOOL)*

S premennými robíme operácie pomocou operandov. Tie rozdeľujem na:

1.*Aritmetické*: \* , / , + , -, =, ++, --, %

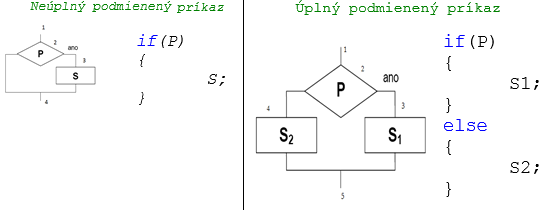
2.*Logické*: negácia, logický súčin, logický súčet

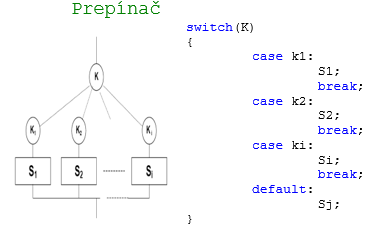
3.*Relačné*: <, <=, ==, !=, >=, >

4.*Binárne*: bitový súčet, súčin, vylúčenie, negácia, posun

**VETVENIE**

Vetvenie je v algoritmizácii reprezentované podmienkou, ktorá predstavuje možnosť rozhodnúť sa podľa „PRAVDIVOSTI“ podmienky. Ak je podmienka splnená, pokračuje sa vykonávaním vetvy označenej ako „+“ , v opačnom prípade sa spracúvajú príkazy vo vetve „-“.





**CYKLUS**

Cyklus poskytuje prostriedok umožňujúci opakovať činnosť alebo činnosti.

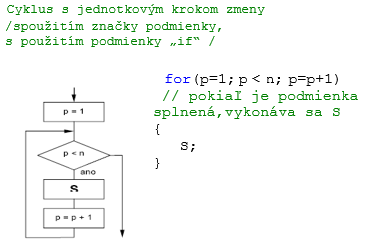
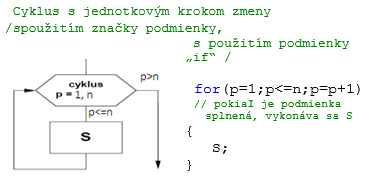
Iterácia je jedno vykonanie cyklu – napr. cyklus čo sa opakuje 4x, vykonáva 4 iterácie.

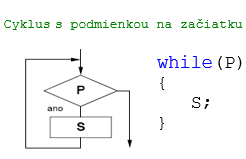
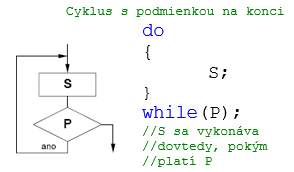
V závislosti od vzťahu medzi telom a podmienkou cyklu môžeme cykly rozdeliť na:

*-cyklus so známym počtom opakovaní*

*-cyklus s podmienkou na začiatku*

*-cyklus s podmienkou na konci*

**POJMY**

Program je sekvencia inštrukcií, vykonaním ktorej počítač vyrieši zadanú úlohu. Postup riešenia úlohy vyjadruje algoritmus úlohy.

Programy sa delia na:

1.*Systémové* – vytvárajú operačný systém počítača alebo firmware

2.*Aplikačné* – slúžia užívateľovi, riešia určité praktické úlohy

Program tvoríme tak, že hlavný cieľ rozdelíme na viacero menších úloh a tie na ešte jednoduchšie atď. Tieto úlohy sú tvorené algoritmami. Využívame dekompozíciu – rozdelenie problému na podproblémy, ktoré sú z určitého hľadiska jednoduchšie a teda pre nás viac známe a riešiteľné.

Program zapisujeme: *slovne, vývojovým diagramom, pseudokódom alebo programovacím jazykom*.

Jednotky informácii: b/B , k/M/G/T/P

Procesor – vykonáva inštrukcie a teda program

Operačná pamäť – uchováva dáta, s ktorými pracuje procesor (RAM)

Inštrukcia – elementárny príkaz pre procesor (napr. zvýš o 1, sčítaj, zapíš, porovnaj rovnosť, ...)

Strojový kód – program v tvare, ktorému rozumie procesor (0 a 1)

Zdrojový kód – program v tvare, v ktorom ho zapisuje programátor (čitateľnejší / syntax prog.jazyka)

Kompilátor – program, ktorý transformuje zdrojový kód na strojový

Vývojové prostredie – program, v ktorom programujeme (editor textu + kompilátor + simulátor, ....)

**GIT / GITHUB**:

Ide o verzovací systém, čiže program/službu, ktorá zálohuje a spravuje rôzne verzie súborov.

Github je konkrétny príklad/implementácia GITu. Firmy majú vlastné GITy.

Github je možné spravovať cez web, aplikáciu alebo doplnok do programu.

Základným pojmom je: repozitár, commit, push, pull, fork, clone, merge, branch

Návod a viac info: **https://github.com/SPSKNM-cvicenia/Github-Introduction**

*Príklad: Vytvorte VD pre zadanie čísla. Pokiaľ bude v rozsahu 1 až 5, napíše sa „známka“, inak „chyba“. (IF)*

*Príklad: Vytvorte VD pre kalkulačku. Zadajú sa dve čísla a znak +,-,\*,/. Podľa znaku vykoná operáciu a vypíše výsledok. (SWITCH)*

*Príklad: Vytvorte VD pre štafetový beh. Program bude 4x náhodné číslo v rozsahu 8-11. Nakoniec vypíše súčet týchto časov a priemerný čas. (FOR)*

*Príklad: Vytvorte VD pre hľadanie čísla. Program opakovane porovnáva zadávané číslo s vopred vygenerovaným a hlási, či je číslo väčšie alebo menšie. Pokiaľ je rovné, program sa ukončí.*

*Príklad: Vytvorte VD pre pravidelné generovanie čísla, ktoré bude iné, ako predošlé. Program generuje v 1sek. Intervaloch náhodné číslo od 1 do 10 a vypisuje ho.*

*Príklad: Vytvorte VD pre získanie max. veľkosti. V úvode sa zadá 7 čísel, následne sa zistí, ktoré z nich je najväčšie.*

**BINÁRNA SÚSTAVA:**

Obsahuje len dva znaky (0/1), z ktorých sú tvorené čísla. V binárnej sústave existujú logické (súčet, súčin) a aritmetické (+, -, \*, /) operácie.



Násobky 2n sú: 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096

|  |  |  |
| --- | --- | --- |
| **DEC (10)** | **BIN (2)** | **HEX (16)** |
| 0 | 0 | 0 |
| 1 | 1 | 1 |
| 2 | 10 | 2 |
| 3 | 11 | 3 |
| 4 | 100 | 4 |
| 5 | 101 | 5 |
| 6 | 110 | 6 |
| 7 | 111 | 7 |
| 8 | 1000 | 8 |
| 9 | 1001 | 9 |
| 10 | 1010 | A |
| 11 | 1011 | B |
| 12 | 1100 | C |
| 13 | 1101 | D |
| 14 | 1110 | E |
| 15 | 1111 | F |
| 16 | 10000 | 10 |

|  |  |  |
| --- | --- | --- |
| **+** | **b=0** | **b=1** |
| **a=0** | **0** | **1** |
| **a=1** | **1** | **0 (1)** |

Aritmetický súčet:

+1011

+1001

10100

|  |  |  |
| --- | --- | --- |
| **\*** | **b=0** | **b=1** |
| **a=0** | **0** | **0** |
| **a=1** | **0** | **1** |

Aritmetický súčin:

1011\*1001 = ?

1011

0000

0000

1011

1100011

|  |  |  |
| --- | --- | --- |
| **|** | **b=0** | **b=1** |
| **a=0** | **0** | **1** |
| **a=1** | **1** | **1** |

Logický súčet:

+ 10110

| 10011

10111

|  |  |  |
| --- | --- | --- |
| **&** | **b=0** | **b=1** |
| **a=0** | **0** | **0** |
| **a=1** | **0** | **1** |

Logický súčin:

+ 10110

& 10011

10010

**PREVODY MEDZI ČÍSELNÝMI SÚSTAVAMI:**

**Prevod 2->10:**

**110101112 = ?10**

**110101112** = 1\*27 + 1\*26 + 0\*25 + 1\*24 + 0\*23 + 1\*22 + 1\*21 + 1\*20 =

1\*128 + 1\*64 + 0\*32 + 1\*16 + 0\*8 + 1\*4 + 1\*2 + 1\*1 = **21510**

**Prevod 10->2:**

**21510 = ?2**

215 : 2 = 107 ; zv. 1

107 : 2 = 53 ; zv. 1

53 : 2 = 26 ; zv. 1

26 : 2 = 13 ; zv. 0

13 : 2 = 6 ; zv. 1

6 : 2 = 3 ; zv. 0

3 : 2 = 1 ; zv. 1

1 : 2 = 0 ; zv. 1 **= 110101112**

Iný spôsob:

215 ; 128 ; nachádza **1x** ; zv. 87

87 ; 64 ; nachádza **1x** ; zv. 23

23 ; 32 ; nenachádza

23 ; 16 ; nachádza **1x** ; zv. 7

7 ; 8 ; nenachádza

7 ; 4 ; nachádza **1x** ; zv. 3

3 ; 2 ; nachádza **1x** ; zv. 1

1 ; 1 ; nachádza **1x** ; zv. 0

**Prevod 16->2: Prevod 2->16:**

(Musí platiť vzťah: **M = kn** , kde *n* určuje počet cifier čísla *k* z rozkladu číslice *M)*

**3AE16 = ?2**

3 A C

0011 1010 1100

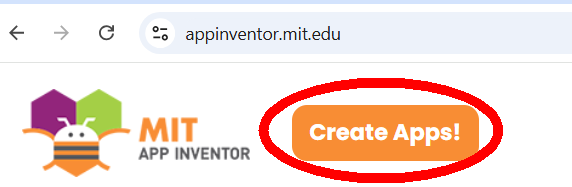
**1100111101012 = ?16**

1100 1111 0101

C F 5

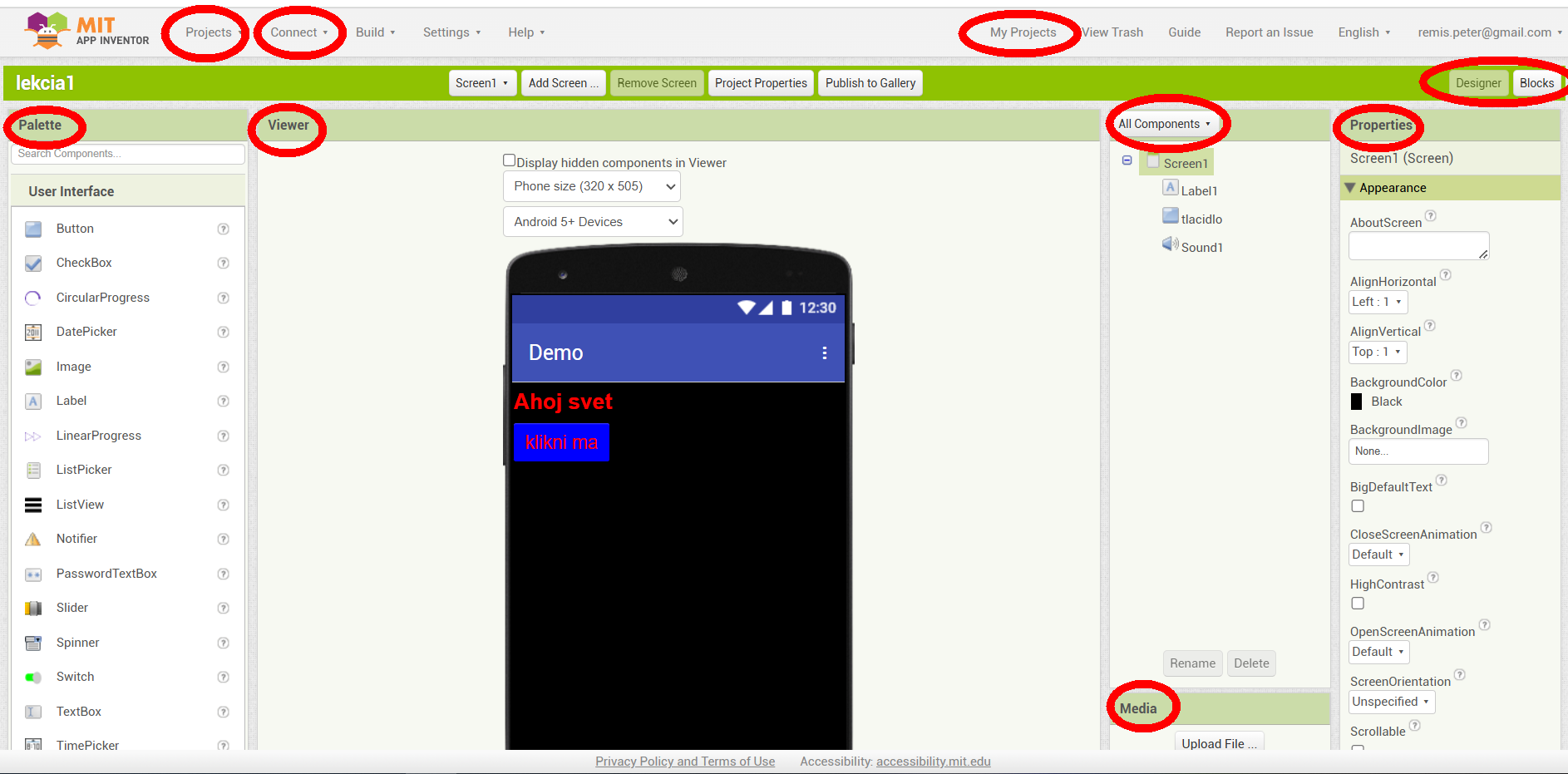
**Programovanie mobilných aplikácii pomocou MIT App Inventor**

Vyžaduje sa registrácia a prihlásenie. Na cloude sa ukladajú automatický všetky projekty.

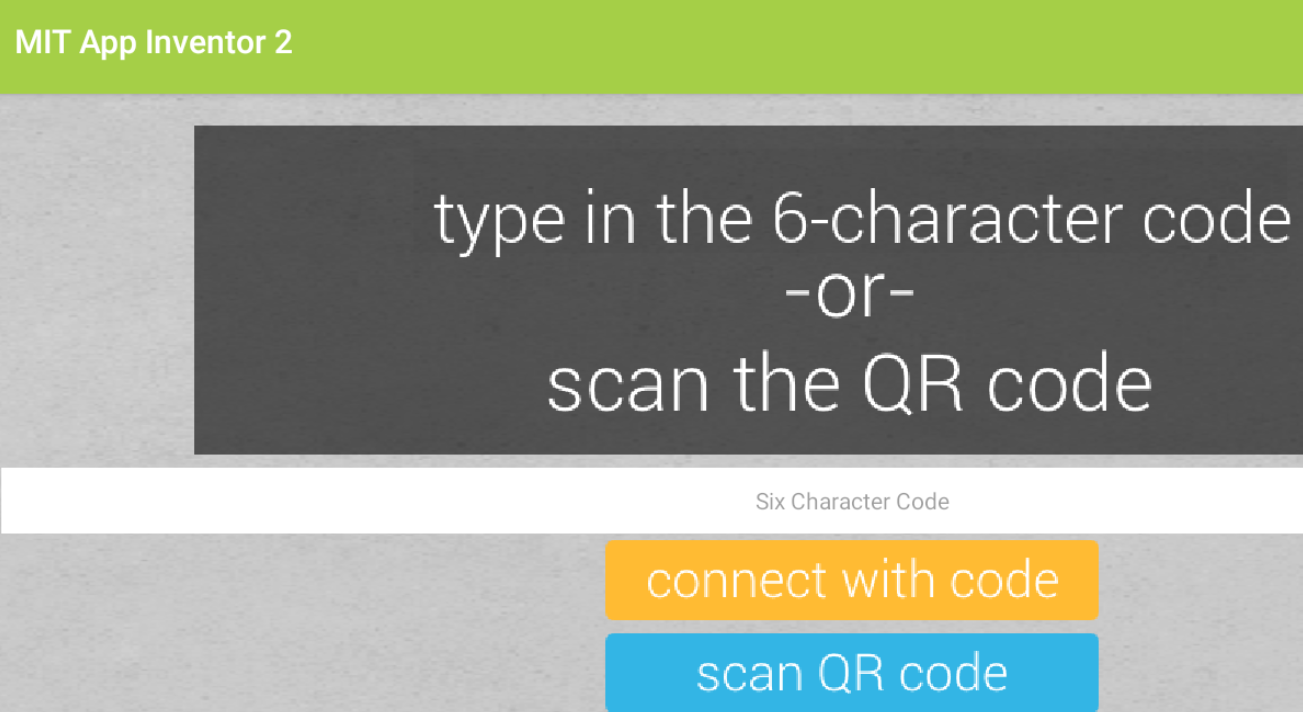


Rozhranie obsahuje:

* menu pre spravovanie projektu, upload, tvorbu APK, zoznam projektov atď.
* okno palety obsahujúce zoznam komponentov, ktoré vkladáme do aplikácie
* okno zobrazenia, ktoré zobrazuje usporiadanie komponentov
* okno komponentov v jednotlivých obrazovkách
* okno vlastností jednotlivých komponentov
* prepínanie medzi obrazovkou užívateľského rozhrania a programom/logikou



Spustenie aplikácie je možné buď na reálnom mobile, alebo v simulátore/emulátore. Simulátor je vstavaný (ale pomalý), externý (virtualizovaný). Kvôli rýchlosti a jednoduchosti je možné použiť aplikáciu BlueStacks, ktorá emuluje Android aplikácie. Pokiaľ užívateľ chce použiť reálne zariadenie, je ho možné pripojiť cez USB alebo cez sieť pomocou aplikácie MIT AI2 COMPANION. V nej sa potvrdí aplikácia buď zadaním 6 miestneho kódu, alebo pomocou QR kódu z MENU/Connect/AI Companion.

i 

Príklad: vytvorte kalkulačku, ktorá sčíta dve zadané čísla po stlačení tlačidla.

