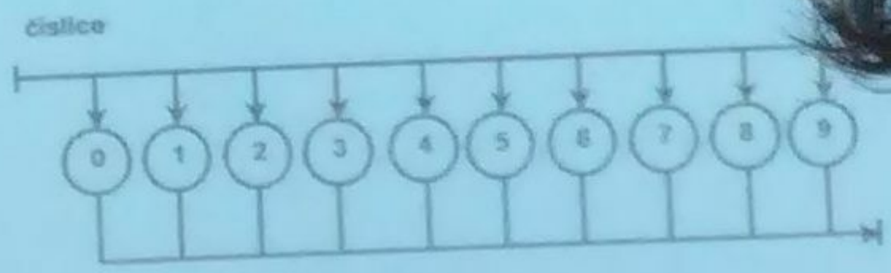
IZP – vzorové řešení

A, Příklad 1: Vysvětlete pojem syntaxe programovacího
věty). Uveďte příklad syntaxe **číslice** (pro číslice des
soustavy)

Řešení: Syntaxe: *Soubor pravidel udávající přípustné*
programů. Popisuje formální strukturu programu.
slova, identifikátory, čísla a další programové enti
způsob, jak je lze kombinovat. Na základě syntak
lze posoudit, zda určitý text je či není korektním
v daném jazyce.

Syntaxe číslice: *BNF: <číslice> = 0|1|2|3|4|5|6*

Syntaktický diagram:



IZP – vzorové řešení

B, Příklad 1: Vysvětlete pojem syntaxe programovacího jazyka. (1–3 věty). Uveďte příklad syntaxe **písmeno** (pro velká písmena anglické abecedy). [2

Řešení: Syntaxe: *Soubor pravidel udávající přípustné konstrukce programů. Popisuje formální strukturu programu. Definuje klíčová slova, identifikátory, čísla a další programové entity a určuje způsob, jak je lze kombinovat. Na základě syntaktických pravidel lze posoudit, zda určitý text je či není korektním zápisem programu v daném jazyce.*

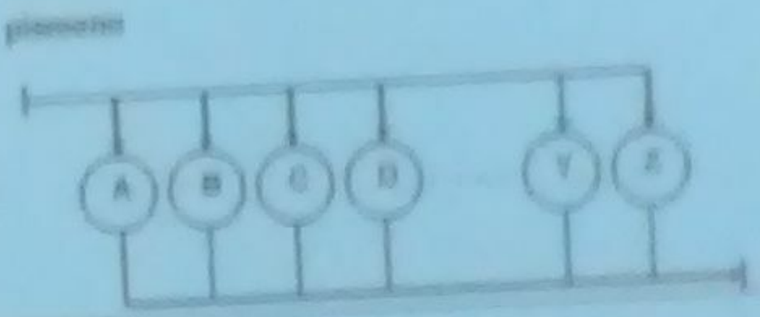
Syntaxe písmeno :

BNF:

<písmeno> =

A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z

Syntaktický diagram:



IZP – vzorové řešení

A, Příklad 2: Popište, co je výsledkem použití operátoru size

Řešení:

Výsledkem operátoru sizeof (int) je velikost typu int *(v po*
vývojovém prostředí) v bajtech.

B, Příklad 2: Vysvětlete pojmy: „ukazatel", „reference" a „dereference". [3 bod

Řešení:

ukazatel: *datový typ, jehož hodnota vyjadřuje místo/umístění dat*
v paměti (1b)
reference: *hodnota pro nepřímý přístup k proměnné.*
dereference: *přístup k hodnotě proměnné přes její referenci*
také:
operátor reference & – získání ukazatele na data v p
*operátor dereference * – získání hodnoty dat pře*
*(samotné citování & a * à 0b).*

IZP – vzorové řešení

A, Příklad 3: Je dán následující kód:

```
#include <stdio.h>
int main(void) {
  for(int x=0; x<10;)
  x++;
  printf("x je rovno  %d ", x);  return 0;
```

kompiluje se uvedený program? Pokud ne, proč ne? Poku
se zobrazilo při spuštění programu?

ešení: Program se nepřeloží. Proměnná x v příkazu printf("x je rovn
x); není definovaná.

Příklad 3: Je dán následující kód:

```
float f = 0.0;
for (int i=0; i<10; i++) f = f + 0.1;
if (f == 1.0f) printf ("Vysledek je: %6.4f", f); // XX
```

tlete, proč podmínka uvedená na řádku ozna
splněna.

*Typ float je pouze aproximací reálných čísel. 10
e přibližuje hodnotě 1.0 (1b), ostrá rovnost*

IZP – vzorové řešení

A, Příklad 4: Uveďte příklady (název) datových strukt
struktury také uveďte definici proměnné pro uv
struktury.

Řešení: homogenní: pole, **heterogenní:** záznam
dynamická: lineární seznam

Statická: pole

B, Příklad 4: Uveďte základní řídící struktury v programovac
každé řídící struktury uveďte příklad.

Řešení:

sekvence (posloupnost): (1b)

```
{
 podil = a / b;
 zbytek = a % b;
}
```

selekce (větvení): (1b)

```
if(vyraz_podminky)
  printf("X\n");
else
  printf("Y\n");
```

terace (cyklus): (1b)

```
for(int num = 1; num < 11; ++num)
  printf("%d ", num);
```

IZP – vzorové řešení

A, Příklad 5: Je dána definice: unsigned int r; // inicializace x

Napište **výraz** vyjadřující, zda rok uložený v proměnné r je pře
nebo není přestupný (false). Přestupné roky, podle Greg
kalendáře, jsou roky dělitelné 4, avšak roky dělitelné 10
pouze tehdy, pokud jsou dělitelné i 400. Nepoužívejte
operátor!

sú prestupné

ternárny

Řešení:

```
((r % 4 == 0 ) && ((r % 100 > 0) || (r % 400 == 0)))
```

nebo lépe: `(!(r % 4) && ((r % 100) || !(rok % 400)))`

B, Příklad 5: Je dáno: int x; // inicializace x

Napište **výraz**, který nabývá hodnoty true, když hodnota uložená v pr
není dělitelná žádným číslem z množiny {3,5,7}. Nepoužíve
operátor!

ternárny

Řešení:

```
((x % 3 != 0) && (x % 5 != 0) && (x % 7 !=
```

nebo lépe: `((x % 3 ) && (x % 5 ) && (x % 7 )`

IZP – vzorové řešení

A, Příklad 6: Je definována funkce:

```
#include <stdio.h>
void test(int a, int b)
{
  if (a = b)
    printf("a je rovno b\n"); //  XX
  else
    printf("a není rovno b\n");
}
```

Pro jaké hodnoty argumentů a a b, při volání této funkce, se **vždy** provede příkaz na řádku označeném XX?

Řešení: *na hodnotě a nezáleží, b ≠ 0*

B, Příklad 6: Je definována funkce:

Pro jaké hodnoty argumentů a a b, při volání této funkce, s**nikdy** neprovede příkaz na řádku označeném XX?

Řešení: *na hodnotě a nezáleží, b = 0*

IZP – vzorové řešení

A, Příklad 7: Upravte následující funkci tak, aby zobrazila všechny prv
hlavní diagonále čtvercové matice řádu n **po řádcích v opačné**
uložené ve dvojrozměrném poli dané parametrem array.

```
void printMainDiagRev (int n, int array[n][n])
// parametr n je řád matice

                    0     < n        ++
  for (int i = 1; i <= n+1; i--)

                                n-i-1      n-i-1
    printf ("%d \n", array[n-1][n-1]);
```

IZP – vzorové řešení

B, Příklad 7: Upravte následující funkci tak, aby zobrazila všechny p
hlavní a na vedlejší diagonále čtvercové matice řádu n v po
doprava, shora dolů uložené ve dvojrozměrném poli dané para
array. V případě, že některý prvek matice leží současně na h
vedlejší diagonále, zobrazí se v odpovídajícím řádku pouze

```
void printMainSideDiag (int n, int array[n][n])
// parametr n je řád matice
{
  for (int i = 0; i < n-1; i++)
   if (i == n-i-1
    printf ("%d \n", array[i][i]);
   else
    printf ("%d %d \n",array[i][i<n/2 ? i:n-i-1],
                       array[i][i<n/2 ? n-i-1:i]
```

IZP – vzorové řešení

AB, Příklad 8: Je dáno:

```
typedef struct { …; int pay; } tdata;
struct item {  tdata data;  titem *next;};
typedef struct item titem;
typedef struct {  titem *head; … } tlist;
```

listFindMin

Definujte funkci listFindMax, která vrací ukazatel na položku s největší hodnotou složky pay v lineárním seznamu (daném parametrem). V případě, že je seznam prázdný, funkce vrací NULL.

Řešení:

```
titem *listFindMax (tlist *list)
{
  titem *maxitem = list->head;
  for (titem *tmp = list->head; tmp != NULL; tmp = tmp->next)
    if (tmp->data.pay > maxitem->data.pay)
      maxitem = tmp;
  return maxitem;
}
```

Diskuze: V případě, že bude v lineárním seznamu více položek s největší hodnotou složky pay, funkce vrátí ukazatel na první z nich s největší hodnotou složky pay.

IZP – vzorové řešení

A, Příklad 9: Co se zobrazí po provedení následujícího program
Uveďte přesně výsledek, který se zobrazí na standardní vý
[10 bo

```
#include <stdio.h>
void myPrint (int n) {
  printf ("%d", n/2);
  if (n > 0)  //  Místo A
    myPrint (n - 1); // Místo B
  printf ("%d", n);
}
int main (void) { int count = 3; myPrint (count); return
```

Popište, jaký význam má kód v místech označených jako A a

Řešení:

Zobrazí se: *11000123 (6b)*

Místo A: *podmínka ukončení rekurze. (2b)*

Místo B: *rekurzívní volání funkce* myPrint. (

IZP – vzorové řešení

A, Příklad 10: Stručně charakterizujte základní vlastnosti me
věty pro každou vlastnost). [

Řešení:

Sekvenčnost: *Je vlastnost, která vyjadřuje, že řadící algoritmus pra vstupními údaji i s datovými meziprodukty v tom pořadí v jaké lineárně uspořádány v datové struktuře. (1b)*

Časová složitost: *Označuje míru času potřebnou k realizaci dané algoritmu. U řadicích algoritmů ji označujeme jako funkci počt řazených prvků. (1b)*

Prostorová složitost: *Označuje míru prostoru potřebnou k realiza algoritmu. U řadicích algoritmů ji označujeme jako funkci počt řazených prvků. (1b)*

Přirozenost: *Je vlastnost algoritmu, která vyjadřuje, že doba potřeb řazení již seřazené množiny údajů je menší než doba pro seřaz náhodně uspořádané množiny a ta je menší než doba pro seřaz seřazené množiny údajů. (1b)*

Stabilita: *Je vlastnost řazení, která vyjadřuje, že alg vzájemné pořadí údajů se shodnými klíči. (1b)*

situ: *Metoda pracuje **in situ** znamená, že metoda n prostor pro řazení, než zabírá původní řazená str*

IZP – vzorové řešení

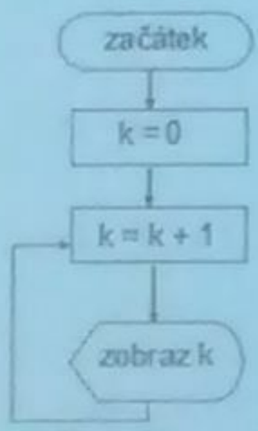
B, Příklad 10: Stručně charakterizujte základní vlastnosti algoritmů (1–
pro každou vlastnost). Uveďte příklad algoritmu, který není konečn
příklad algoritmu, který není hromadný. [6 bodů]

Řešení:

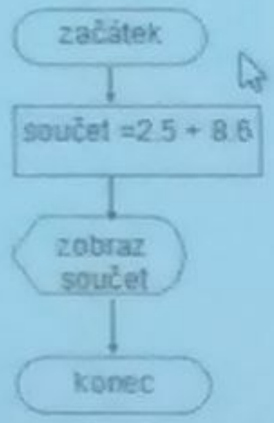
Konečnost (rezultativnost): *má zaručit vyřešení úlohy po konec*
kroků. (1b)

Hromadnost: *jedním algoritmem lze řešit celou třídu úloh stejného*

***Příklad** (není konečný)*:



***Příklad** (není hromadný):*



novanost: *algoritmus je zadaný ve formě konečného poč*
značných pravidel. (1b)

na správný průběh programu nemá žádný vliv,
val co nejkratší dobu. (1b)