IZP – vzorové řešení – RT_1

Příklad 1: [2 body] Vysvětlete pojmy syntaxe a sémantika programovacích jazyků (každý pojem 1–3 věty).

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.*

Sémantika: *Určuje logický význam jednotlivých výrazů jazyka. Ze sémantiky plyne, jaký má daná konstrukce význam.*

IZP – vzorové řešení – RT_1

Příklad 2: [2 body] Je definováno inicializované pole:

```
int a [11] = {7,6,4,[7]=10,9,8,5};
```

Jakou hodnotu bude mít prvek a[3]?

Řešení: *Prvek* a[3] *bude mít hodnotu 0. (2b)*

IZP – vzorové řešení – RT_1

Příklad 3: [2 body] Je dán následující kód:

```
unsigned int a = 4, b = 0, c;
c = a & b || a && b;
```

Jakou hodnotu bude mít proměnná c po provedení tohoto kódu?

Řešení: *Proměnná c bude mít hodnotu 0. (2b)*

IZP – vzorové řešení – RT_1

Příklad 4: [4 body] Jaká bude hodnota proměnné sum po provedení následujícího kódu?

```
int sum = 3;
 for (int i = 0; i < 10; i++)  {
   switch (i)  {
     case 1: case 3: case 4: case 7:  sum++;
     default: continue;
     case 5: break;
   }
 break;
 }
```

Řešení: *Proměnná sum bude mít hodnotu 6. (4b)*

IZP – vzorové řešení – RT_1

Příklad 5: [4 body] Jsou dány definice:

```
int a = 2, b = 2, c = 1, d = 0, e = 4;
```

Jaké budou hodnoty následujících výrazů? *(4b)*

```
(a++  /  ++c  *  --e)
```

Hodnota výrazu bude: *3*

```
(--b  *  c++  -  a)
```

Hodnota výrazu bude: *-1*

IZP – vzorové řešení – RT_1

Příklad 6: [4 body] Uveďte a stručně popište alespoň dva způsoby zpracování chyb v programu. *(4b)*

Řešení

Způsoby zpracování chyb v programu:

– ***Prevence*** = *návrh programu tak, aby k chybám nedošlo.*

– ***Propagace*** = *program/část programu počítá s možností chyby, zotavení nechává na volajícím.*

– ***Terminace*** = *bez zotavení, při chybě ihned ukončuje činnost.*

IZP – vzorové řešení – RT_1

Příklad 7: [8 bodů] Upravte následující funkci tak, aby zobrazila všechny prvky **pod vedlejší diagonálou** čtvercové matice řádu n (po řádcích v původním pořadí, tj. v obou směrech ve směru rostoucích indexů) uložené ve dvourozměrném poli daném parametrem array. Výsledný algoritmus musí vykonat minimální možný počet iterací.

```
void printUnderSideDiag (int n, int array[n][n])
{ // parametr n je řád matice
  for (int i = 1; i < n; i++)
  {
    for (int j = n - i; j < n; j++)
      printf ("%d  ", array[i][j]);
    printf ("\n");
  }
}
```

IZP – vzorové řešení – RT_1

Příklad 8: [12 bodů] Je dáno:

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

Definujte funkci listFindMax, která vrací ukazatel na položku s maximální hodnotou složky pay v lineárním seznamu (daném parametrem funkce). V případě, že je seznam prázdný, funkce vrací NULL. V případě, že bude v lineárním seznamu více položek se shodnou maximální hodnotou složky pay, funkce vrátí ukazatel na první nalezenou položku s maximální hodnotou složky pay.

Řešení:

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

IZP – vzorové řešení – RT_1

Příklad 9: [10 bodů] Co se zobrazí po provedení následujícího programu? Uveďte přesně výsledek, který se zobrazí na standardní výstup.

```
#include <stdio.h>
void myPrint (int n)
{
  printf ("%d", n/2);
  if (n > 0)  //  Místo A
    myPrint (n - 2); // Místo B
  printf ("%d", n);
  return;
}
int main (void)
{
  int count = 4;
  myPrint (count);
  return 0;
}
```

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

Řešení:

Zobrazí se: *210024 (6b)*

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

Místo B: *rekurzívní volání funkce* myPrint.

IZP – vzorové řešení – RT_1

Příklad 10: [6 bodů] Uveďte a stručně popište tři možné typy dokumentace softwaru. U každého typu také uveďte komu zejména slouží.

Řešení

Dokumentace softwaru může být:

– ***Uživatelská příručka*** *- slouží pro uživatele, popisuje, jak se program používá.*

– ***Architektonický návrh*** *- slouží pro systémové inženýry, popisuje strukturu nebo chování systému.*

– ***Technická dokumentace*** *- slouží pro vývojáře, dokumentuje zdrojové soubory.*