

CZP – vzorové řešení

Zadání A, příklad 1: Vysvětlete jednou větou, co znamená instrumentace programů. [2 body]

Řešení:

Instrumentace programů spočívá v doplnění kontrolních tisků do programu pro účely ladění.

Zadání A, příklad 2: Je definováno inicializované pole: [2 body]

`int a[10] = {1, 2, 3, [8]=8, 10};`

Jakou hodnotu bude mít prvek `a[10]`?

Řešení: Není definováno.



IZP – vzorové řešení

Zadání A, příklad 3: Je dán následující kód:

```
unsigned int a = 1, b = 2, c;
```

```
c = a & b || a && b;
```

Jakou hodnotu bude mít proměnná c po provedení tohoto kódu? [2 body]

Řešení: Proměnná c bude mít hodnotu 1.

IZP – vzorové řešení

Zadání A, příklad 4: Jaká bude hodnota proměnné sum po provedení [6]
následujícího kódu?

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

Řešení: Proměnná sum bude mít hodnotu 2.

IZP – vzorové řešení

Zadání A, příklad 5: Vysvětlete stručně (2-3 věty) princip manipulace s daty abstraktního datového typu zásobník (LIFO). [3 body]

Řešení:

Pro zásobník platí, že data uložená do zásobníku jako poslední, budou čtena jako první. Pro manipulaci s uloženými datovými položkami se udržuje tzv. ukazatel zásobníku, který udává relativní adresu poslední přidané položky, tzv. vrchol zásobníku.

IZP - vzorové řešení

Zadání A, příklad 6: Upravte následující funkci tak, aby zobrazila všechny prvky pod vedlejší diagonálou čtvercové matice řádu n (po řádcích v původním pořadí) uložené ve dvourozměrném poli, daném parametrem array. Výsledný algoritmus musí vykonat minimální možný počet iterací. Úpravu vyznačte přímo ve funkci pouze na řádcích označených XX. [8 bodů]

Řešení:

```
#include <stdio.h>

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

1	2	3	8
6	7	8	9
4	5	9	

Zadání A, příklad 7: Je dáno:

```
typedef struct  
{char name[11]; char surname[16]; int pay;} tdata;  
typedef struct item titem;  
struct item { tdata data; titem *next; };
```

```
typedef struct { titem *head; titem *tail; } tlist;  
Definujte funkci countItemDifferentPay, která vrací počet položek lineárního  
seznamu (daného prvním parametrem funkce), ve kterých je hodnota složky  
pay různá od hodnoty dané parametrem payValue. Předpokládejte, že seznam  
obsahuje minimálně jednu položku.
```

Řešení:

```
int countItemDifferentPay (const tlist *list, int payValue)  
{  
    int count = 0; // inicializace počítadla  
    for (titem *tmp = list->head; tmp != NULL; tmp = tmp->next)  
        if (tmp->data.pay != payValue)  
            count++;  
    return count;  
}
```



IZP – vzorové řešení

Zadání A, příklad S: Analyzujte následující fragment programu a za předpokladu, že $-10 \leq x \leq 2$, uvedte platné vstupní podmínky vzhledem k proměnné x pro příkazy bloku S1 a pro příkazy bloku S2 [4 body]

```
float x;  
// inicializace x  
// v tomto místě platí pro proměnnou x:  $-10.0 \leq x \leq 2.0$   
if (x ≤ 0.0)  
{  
    // příkazy bloku S1  
} else if (x ≥ 2.0)  
{  
    // příkazy bloku S2  
}
```

Rešení:

/stupní podmínka bloku S1: $-10 \leq x \leq 0.0$ (1+1b)
/stupní podmínka bloku S2: $x = 2.0$ (2b)

IZP – vzorové řešení

Zadání A, příklad 9: Co se zobrazí po provedení následujícího programu? Uveďte přesně výsledek, který se zobrazí na standardní výstup. [12 bodů]

```
#include <stdio.h>
unsigned long vypocet (unsigned long a, unsigned long b)
{ if (a * b == 0) return 0; // Místo A
  if (a < b)
    { a ^= b; b ^= a; a ^= b; // Místo B
      // ^ je operátor XOR - eXclusive OR }
    return (a % b > 0) ? vypocet(b, a % b) : b; // Místo C
    // % je operace modulo }
int main(void)
{ unsigned long x = 204; unsigned long y = 114;
  printf("%lu", vypocet(x, y)); return 0; }
```

Popište co funkce vypocet ve skutečnosti dělá? Popište, jaký význam má kód v místech označených jako A, B a C.

[obrazí se: 6 (4b)]

Jaký problém funkce řeší?: Funkce vypočítá největší společný dělitel dvou přirozených čísel. (2b)

Místo A: koncová podmínka rekurze, pokud bude některá z proměnných nulová, funkce vrátí nulu. (2b)

Místo B: swap, neboli výměna hodnot proměnných a a b, aby bylo před rekursivním voláním zaručeno, že $a \geq b$. (2b).

Místo C: rekursivní volání funkce vypocet. (2b)

IZP – vzorové řešení

Zadání A, příklad 10: Stručně vysvětlete základní vlastnosti metod řazení (1-2 věty pro každou vlastnost). [6 bodů]

Řešení:

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

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

Prostorová složitost: Označuje míru prostoru potřebnou k realizaci daného algoritmu. U řadicích algoritmů ji označujeme jako funkci počtu n řazených prvků. (1b)

Přirozenost: Je vlastnost algoritmu, která vyjadřuje, že doba potřebná k řazení je seřazené množiny údajů je menší než doba pro seřazení náhodně uspořádané množiny a ta je menší než doba pro seřazení opačně seřazené množiny údajů. (1b)

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

In situ: Metoda pracuje *in situ* znamená, že metoda nepožaduje výrazně větší prostor pro řazení, než zabírá původní řazená struktura. (1b)