Informatika – 10. 9. 2013

Jméno a příjmení	Rodné číslo

1) Napište algoritmus pro rychlé třídění (quicksort).

2) Napište algoritmus pro vložení položky na konec dvousměrného seznamu.

3) Napište algoritmus pro vyhledání položky v binárním stromu.

4) Vysvětlete využití výjimek (v jazyce C++).

5) Co je to destruktor, kdy je volán ?

6) Jaký má vliv první středník v následujících příkazech:

```
while (n < 10) ; n ++ ;
```

7) Vyberte vhodný příkaz pro vyhledávání hodnoty v seznamu:

```
a) while (p->key != value && p != NULL) p = p->next;
b) while (p != NULL && p->key != value) p = p->next;
```

Výběr zdůvodněte.

8) Napište deklaraci šablony třídy, která bude představovat prvek spojového seznamu.

9) Slovy popište význam následující deklarace:

```
int * f ( int, int * );
```

Na následující stránce jsou další tři příklady

10) Nechť jsou dány deklarace:

```
class A { };
class B : public A { };
class C : public B { };
A * p = new B;
```

Jaké budou typy a hodnoty následujících výrazů:

```
dynamic_cast < B * > (p)
dynamic_cast < C * > (p)
```

11) Nechť jsou dány deklarace:

```
class A
{
   public:
      virtual void f ();
      void g ();
};
class B : public A
{
   public:
      virtual void f ();
      void g ();
};
A * p = new B;
```

Které funkce budou zavolány následujícími příkazy ?

```
p->f ();
p->g ();
```

12) Nechť je dána procedura pro násobení matic:

```
const N = 100;
type pole = array [1..N, 1..N] of real;
procedure mult (var v: pole; var a, b: pole);
var i, j, k: integer;
    s: real;
begin
   for i := 1 to N do
      for j := 1 to N do begin
         s := 0;
         for k := 1 to N do
            s := s + a [i,k] * b [k,j];
         v [i,j] := s;
      end;
end;
```

Nalezněte chybu při použití této procedury v následující situaci:

```
mult (D, D, E);
```

Navrhněte vhodnou opravu.

Informatika – 10.9. 2013

Příklad 1:

```
Const n = 100;
Var   a: array [1..n] of real;

Procedure QuickSort (ii, jj: integer);
Var  i, j, s: integer;
       h, t:  real;
Begin
    i := ii;
    j := jj;
    s := (i + j) div 2;
    h := a [s]; (* pivot *)

    while i <= j do begin
       while a[i] < h do i := i + 1;
       while a[j] > h do j := j - 1;

       if i <= j then begin
          t := a[i];
          a[i] := a[j];
          a[j] := t;
          i := i + 1;
          j := j - 1;
       end;
   end;

   if ii < j then QuickSort (ii, j);
   if i < jj then QuickSort (i, jj);
end;

Begin
        QuickSort (1, n);
End.
```

Příklad 2:

```
class Node
{
    public:
       int key;
       Node * prev;
       Node * next;
};

Node * first, * last;

void insert_last (int new_key)
{
```

```
    Node * n = new Node;
    n->key = new_key;
    n->next = NULL;
    n->prev = last;

    if (last == NULL)
       first = n;
    else
        last->next = n;
}
```

Příklad 3:

```
class Node
{
   public:
     int key;
     Node * left, right;
};

Node * root;

Node * search (int value)
{
    Node * p = root;
    while (p != NULL && p->key != value)
    {
        if (p->key < value) p = p->left; else p = p->right;
    }
    return p; // ukazatel na nalezenou polozku, nebo NULL pokud hodnota nebyla nalezena
}
```

Příklad 4:

Výjimky dovolují přerušit výpočet a pokračovat v místě pro ošetření chyby či neobvyklé situace.

Výjimku lze vyvolat příkazem **throw** výraz; nebo výjimka vznikne chybou v programu (dělení nulou, použití nulového ukazatele, …)

Příkaz **try** zachycuje výjimky vzniklé v příkazu následujícím po klíčovém slově try.
Příkaz **try** obsahuje jednu nebo více částí **catch** pro zpracování vzniklých výjimek. Při zachycení výjimky se vyhledá část **catch** jejíž parametr typově odpovídá vzniklé výjimce. Pokud vhodná část **catch** neexistuje šíří se výjimka dále.

Pokud není výjimka ošetřena v původní funkci, šíří se výjimka do funkcí, které původní funkci volaly. (Pokud se pro výjimku nenalezne vhodná konstrukce **catch**, je zavolána standardní funkce terminate.)

Příklad 5:

Destruktor je metoda třídy volaná v okamžiku zániku instance.
Příklad deklarace

```
class C
{
   private:
      int * p;
   public:
      C () { p = NULL; } // konstruktor
      ~C () { if (p != NULL) delete p; } // destruktor
};
```

Instance dynamické proměnné zanikne při volání operátoru delete.

```
    C * u;  u = new C; …… delete u;
```

Instance lokální proměnné zanikne při opuštění oblasti viditelnosti ve které byla proměnná definována.

```
    { C v; .....  /* zde je volán destruktor */ }
```

Globální a statické proměnné zanikají při ukončení programu.

Příklad 6:

První středník reprezentuje prázdný příkaz, který je tělem cyklu:

```
     while (n < 10) ;
```

Pokud je n < 10 je tento cyklus nekonečný.
Pokud je n >= 10 neproběhne tělo cyklu ani jednou.

Po cyklu následuje příkaz pro zvětšení proměnné n.

Příklad 7:

```
while (p != NULL && p->key != value) p = p->next;
```

V okamžiku kdy zpracováváme poslední prvek seznamu (nebo pokud je seznam prázdný), proměnná p obsahuje nulový ukazatel, a proto nelze vyčíslit výraz `p->key`.

Zvolím variantu, která nejprve testuje ukazatel, zda není nulový.
Pokud je první operand před && nepravdivý, druhý operand se nevyhodnocuje.

Příklad 8:

```
    template <class T>
         class Node
         {
             T value;
             Node <T> * next;
         };
```

Příklad 9:

Funkce f vracející jako výsledek ukazatel na celé číslo (int).
Funkce má parametry typu int a ukazatel na int.

Příklad 10:

dynamic_cast <B*> (p) je typu **B*** a ukazuje (po přetypování) na objekt **p**
dynamic_cast <C*> (p) je typu **C*** a má nulovou hodnotu

Příklad 11:

p->f () volá funkci **B::f** (funkci f ze třídy B), neboť f je virtuální.
p->g () volá funkci **A::g**, neboť g není virtuální.

Příklad 12:

V případě volání procedury `mult (D, D, E)`
odkazuje parametr **v** a parametr **a** na stejnou proměnnou.
Všechny parametry jsou předávány odkazem.

Při uložení první hodnoty do výsledného pole **v** dojde změně levého operandu **a**.

Tomu lze zabránit tím, že vstupní operandy budeme předávat hodnotu.

Řádku
```
procedure mult (var v: pole; var a, b: pole);
```
nahradíme řádkou
```
procedure mult (var v: pole; a, b: pole);
```