Programování v C++

Alokace a struktury 7

Pavel Fibich

pavel.fibich@prf.jcu.cz

3-11-2010

Dynamická alokace

Jak alokovat pamět když potřebnou velikost zjistím až za běhu?

```
#include <iostream>
#include <new>
using namespace std;
int main () {
  int i,n, *p;
  cout << "Kolik zadate cisel?";
  cin >> i;
  p= new int[i];
  for (n=0; n<i; n++) {
    cout << "Vloz cislo: ";
    cin >> p[n];
  }
  cout << "Vlozena cisla: ";
  for (n=0; n<i; n++) cout << p[n] << ", ";
  delete[] p;
  return 0;
}
```

Datové struktury

```
struct ovoce {
  int kusu;
  float cena;
} ;
product jablka, hrusky;

struct zelenina {
  int kusu;
  float cena;
} okurka, paprika;

jablka.kusu = 5;

float nakup;
nakup = jablka.cena + hrusky.cena;

ovoce * ukaz;
ukaz = &jablka;
ukaz->cena = 5.7; // to same jako (*ukaz).cena = 5.7;
```

Datové struktury

```
#include <iostream>
#include <string>
#include <sstream>
using namespace std;
#define POLOZEK 3
struct polozka {
  string nazev; int ks;
} polozky [POLOZEK];
int main () {
  string mystr; int n;
  for (n=0; n<POLOZEK; n++)  {
    cout << "Zadej nazev: ";getline (cin,polozky[n].nazev);
    cout << "Zadej pocet ks: "; getline (cin,mystr);
    stringstream(mystr) >> polozky[n].ks;
  }
  cout << "\nZadane polozky :\n";
  for (n=0; n<POLOZEK; n++)
    cout << polozky[n].ks << "x "<< polozky[n].nazev << ↩
        endl;
  return 0; }
```

typedef, union, enum

```
typedef unsigned int WORD;
typedef char * pChar;
typedef char pole [50];
...
WORD prom1;
pChar prom2;
pole prom3;

union mix_t {
  long l;
  char c[4];
} mix;

enum barvy {cervena, modra, zelena, zluta, bila};
barvy mojebarva;

mojebarva = zluta;
if (mojebarva == bila) mojebarva = zelena;
```

Třídy

- Třída (typ) - základní struktura, umí držet data i funkce.
- Objekt (proměnná) - instance třídy.

```
#include <iostream>
using namespace std;
class Ctverec {
    int x, y;
  public:
    void nastav_xy (int,int);
    int plocha () {return (x*y);}
};
void Ctverec::nastav_xy (int a, int b) {
  x = a; y = b;
}
int main () {
  Ctverec ctv;
  ctv.nastav_xy (3,4);
  cout << "plocha: " << ctv.plocha();
  return 0;}
```

Konstruktory

```
...
  public:
    Ctverec ();
    Ctverec (int,int);
    int plocha () {return (x*y);}
...
Ctverec::Ctverec () {
  x = 3;y = 8;
}
Ctverec::Ctverec (int a, int b) {
  x = a;y = b;
}
...
int main () {
  Ctverec ctv (5,6);
  Ctverec ctv2; // NE Ctverec ctv2();
...
```

Vypište plochu obou čtverců.

Ukazatele na třídy

& reference (adresa, address of)

* dereference (hodnota na kterou ukazuje, value pointed by)

```
int main () {
  Ctverec a, *b, *c;
  Ctverec * d = new Ctverec[2];
  b= new Ctverec;
  c= &a;
  a.nastav_xy (1,2);
  b->nastav_xy (3,4); // stejne jako (*b).nastav_xy (3,4);
  d->nastav_xy (5,6);
  d[1].nastav_xy (7,8);
  cout << "a plocha: " << a.plocha() << endl;
  cout << "*b plocha: " << b->plocha() << endl;
  cout << "*c plocha: " << c->plocha() << endl;
  cout << "d[0] plocha: " << d[0].plocha() << endl;
  cout << "d[1] plocha: " << d[1].plocha() << endl;
  delete[] d;
  delete b;
  return 0;
}
```

Přetěžování (overloading) operátorů

```
#include <iostream>
using namespace std;
class Vektor {
  public:
    int x,y;
    Vektor () {};
    Vektor (int a,int b) {x = a;  y = b;};
    Vektor operator + (Vektor);
};
Vektor Vektor::operator+ (Vektor param) {
  Vektor temp;
  temp.x = x + param.x;
  temp.y = y + param.y;
  return (temp);
}
int main () {
  Vektor a (3,5); Vektor b (1,2);
  Vektor c;  c = a + b;
  cout << c.x << "," << c.y;
  return 0; }
```

Další operátory a this

this je ukazatel na aktualní objekt

Následující operátor je generován implicitně:

```
Vektor& Vektor::operator= (const Vektor& param)
{
  x=param.x;
  y=param.y;
  return *this;
}
```

Přidejte do operátoru výpis o provedení a přidejte ho do předchozího programu (nezapomeňte na definici třídy).

Statické proměnné třídy

```
#include <iostream>
using namespace std;

class Dummy {
  public:
    static int n;
    Dummy () { n++; };
    ~Dummy () { n--; };
};

int Dummy::n=0;
int main () {
  Dummy a;
  Dummy b[5];
  Dummy * c = new Dummy;
  cout << a.n << endl;
  delete c;
  cout << Dummy::n << endl;
  return 0;
}
```

Ukol 3. - podmínky

- odvezdání přes ▸ http://moodle.prf.jcu.cz/course/view.php?id=16
- 0-10 bodů
- za opisování jen jedno varování a mínus body pro oba účastníky
- odevzdáte jen komentovaný zdrojový soubor (.cpp), s nazvem podle Vašeho jména a příjmení napr. fibich_pavel.cpp
- na začátku .cpp souboru bude zakomentovaná a vyplněná následující hlavička

```
AUTOR: (jmeno prijmeni, osobni cislo zacinajici B)
EMAIL:
SKUPINA: (kam chodite a jste zapsani na cviceni)
VERZE G++: (ziskate pomoci prikazu g++ -v)
PRIKAZ PREKLADU: (jaky prikazem byl program prelozen)
OS: (operacni system, napr. Win 7 64b)
CISLO UKOLU:
```

Ukol 3. - zadání

- Napište program, který bude mít funkci `int main()` přesně viz. následující frame.
- Dva argumenty funkce `main()` určí počet prvků jednotlivých množin.
- Uživatel bude vyzván k zadání jednotlivých prvků obou množin. Prvky množiny budou jednoslovné string.
- Cílem je napsat třídu Mnozina, se 4 operátory a dalšími pomocnými funkcemi a proměnnými, tak aby program fungoval jako v příkladech.
- Operátory budou provádět jednoduché množinové operace (sjednocení, průnik, rozdíl) a porovnání množin.
- Výstup programu bude mít stejný formát jako v příkladech.
- Program **NE**bude používat kontejnery (Set,Vector,...).

Ukol 3. - funkce main

```
int main (int argc, char *argv[]) {
  if ( argc != 3) {
    cout << "Program chce dva short parametry" << endl;
    return 1;
        }
  cout << "Zadavejte prvky pro mnoziny a a b" << endl;
  Mnozina a(atoi(argv[1])); Mnozina b(atoi(argv[2]));
  Mnozina c, d, e, f;
  cout << "Prvky a: "; a.vypis(); cout << endl;
  cout << "Prvky b: "; b.vypis(); cout <<  endl;
  cout << "Sjednoceni c=a+b : ";
  c=a+b; c.vypis(); cout << " -"<< c.getVelikost() << endl;
  cout << "Prunik d=a*b : ";
  d=a*b; d.vypis(); cout << " -"<< d.getVelikost() << endl;
  cout << "Rozdil e=a-b : ";
  e=a-b; e.vypis(); cout << " -"<< e.getVelikost() << endl;
  cout << "Rozdil f=b-a : ";
  f=b-a; f.vypis(); cout << " -"<< f.getVelikost() << endl;
  cout << "Porovnani e==f : " <<((e==f)?"ANO":"NE")<< endl;
  return 0; }
```

Ukol 3. - příklady

- `ukol3.exe 2 3`

```
Zadavejte prvky pro mnoziny a a b
prvek 1/2 1
prvek 2/2 x
prvek 1/3 2
prvek 2/3 y
prvek 3/3 x
Prvky a: 1 x
Prvky b: 2 y x
Sjednoceni c=a+b : 2 y x 1 -4
Prunik d=a*b : x -1
Rozdil e=a-b : 1 -1
Rozdil f=b-a : 2 y -2
Porovnani e == f : NE
```

- `ukol3.exe 3 1`

```
Zadavejte prvky pro mnoziny a a b
prvek 1/3 jablka
prvek 2/3 hrusky
prvek 3/3 svestky
prvek 1/1 hrusky
Prvky a: jablka hrusky svestky
Prvky b: hrusky
Sjednoceni c=a+b : hrusky jablka svestky -3
Prunik d=a*b : hrusky -1
Rozdil e=a-b : jablka svestky -2
Rozdil f=b-a : -0
Porovnani e == f : NE
```

Ukol 3. - příklady

- `ukol3.exe 2 2`

```
Zadavejte prvky pro mnoziny a a b
prvek 1/2 pes
prvek 2/2 kocka
prvek 1/2 kocka
prvek 2/2 pes
Prvky a: pes kocka
Prvky b: kocka pes
Sjednoceni c=a+b : kocka pes -2
Prunik d=a*b : pes kocka -2
Rozdil e=a-b : -0
Rozdil f=b-a : -0
Porovnani e == f : ANO
```

- `ukol3.exe 4 5`

```
Zadavejte prvky pro mnoziny a a b
prvek 1/4 bila
prvek 2/4 cervena
prvek 3/4 zluta
prvek 4/4 zelena
prvek 1/5 modra
prvek 2/5 zluta
prvek 3/5 hneda
prvek 4/5 bila
prvek 5/5 oranzova
Prvky a: bila cervena zluta zelena
Prvky b: modra zluta hneda bila oranzova
Sjednoceni c=a+b : modra zluta hneda bila oranzova cervena zelena -7
Prunik d=a*b : bila zluta -2
Rozdil e=a-b : cervena zelena -2
Rozdil f=b-a : modra hneda oranzova -3
Porovnani e == f : NE
```