Перегрузка операторов

Перегрузка операторов

- используется для улучшения читаемости и упрощения кода
- общий вид:
  - тип имя_класса::operator#(параметры) {…}
- Можно перегружать все кроме

  :: .* ?
- Операторная функция может быть членом класса или «другом»

С использованием функций-членов

- Операторная функция принимает на один параметр меньше
- Объект, стоящий слева от оператора вызывает операторную функцию и передается ей не явно (через this)
- Объект, стоящий справа передается через параметр

Пример

```
1.  class three_d {
2.    int x, y, z; // 3-D coordinates
3.  public:
4.    three_d() { x = y = z = 0; }
5.    three_d(int i, int j, int k) {x = i; y = j; z = k; }
6.    three_d operator+(three_d op2); // op1 is implied
7.    three_d operator=(three_d op2); // op1 is implied
8.    void show() {   cout << x << ", ";   cout << y << ", ";   cout << z << "\n";
    }
9.  };
10. three_d three_d::operator+(three_d op2) {
11.   three_d temp;
12.   temp.x = x + op2.x;
13.   temp.y = y + op2.y;
14.   temp.z = z + op2.z;.
15.   return temp;
16. }
17. three_d three_d::operator=(three_d op2) {
18.   x = op2.x;   y = op2.y;    z = op2.z;
19.   return *this;
20. }
21. void main() {
22.   three_d a(1, 2, 3), b(10, 10, 10), c;
23.   a.show();   b.show();
24.   c = a + b;   c.show();
25.   c = a + b + c;   c.show();
26.   c = b = a;
27.   c.show();   b.show();
28. }
```

Перегрузка унарных операторов

```
1.  class three_d {
2.    int x, y, z; // 3-D coordinates
3.  public:
4.    three_d() { x = y = z = 0; }
5.    three_d(int i, int j, int k) {x = i; y = j; z = k; }
6.    three_d operator+(three_d op2);
7.    three_d operator=(three_d op2);
8.    three_d operator++(); // prefix version of ++
9.    void show() {   cout << x << ", ";   cout << y << ", ";
10.                   cout << z << "\n"; }
11. } ;
12. three_d three_d::operator++() {
13.   x++;    y++;    z++;
14.   return *this;
15. }
16. void main() {
17.   three_d a(1, 2, 3), b(10, 10, 10), c;
18.   a.show();         b.show();
19.   c = a + b;        c.show();
20.   c = a + b + c;  c.show();
21.   c = b = a;        c.show();   b.show();
22.   ++c;
23.   c.show();
24. }
```

Постфиксная форма

```
1. three_d three_d::operator++(int notused) {
2.   three_d temp = *this; // сохраняем
3.   x++;
4.   y++;
5.   z++;
6.   return temp; // возвращаем не измененную
7. }
```

Операторная функция – не член класса

- Бинарные операции имеют 2 параметра
- Унарные операции имеют 1 параметр
- Часто объявляются «друзьями»
- Нельзя перегружать:

  = () [] ->

Пример

```
1.  class three_d {
2.    int x, y, z; // 3-D coordinates
3.  public:
4.    three_d() { x = y = z = 0; }
5.    three_d(int i, int j, int k) { x = i; y = j; z = k;}
6.    friend three_d operator+(three_d op1, three_d op2);
7.    three_d operator=(three_d op2); // op2 is implied
8.    void show() {   cout << x << ", ";   cout << y << ", ";
9.                    cout << z << "\n";}
10. } ;
11. three_d operator+(three_d op1, three_d op2) {
12.   three_d temp;
13.   temp.x = op1.x + op2.x;    temp.y = op1.y + op2.y;
14.   temp.z = op1.z + op2.z;   return temp;
15. }
16. void main() {
17.   three_d a(1, 2, 3), b(10, 10, 10), c;
18.   a.show();   b.show();
19.   c = a + b;
20.   c.show();
21.   c = a + b + c;
22.   c.show();
23.   c = b = a; c.show();   b.show();
24. }
```

Перегрузка

```
1.  class CL {
2.  public:
3.    int count;
4.    CL operator=(CL obj);
5.    friend CL operator+(CL ob, int i);
6.    friend CL operator+(int i, CL ob);
7.  };

8.  CL CL::operator=(CL obj) {   count = obj.count;   return *this; }
9.  CL operator+(CL ob, int i) {  CL temp;   temp.count = ob.count + i; return temp; }
10. CL operator+(int i, CL ob) {  CL temp;   temp.count = ob.count + i;   return temp; }

11. int main() {
12.   CL O;
13.   O.count = 10;
14.   cout << O.count << " "; // 10
15.   O = 10 + O;
16.   cout << O.count << " "; // 20
17.   O = O + 12;
18.   cout << O.count;       // 32

19.   return 0;
20. }
```

Перегрузка унарных операторов

- Работать не будет:

```
three_d operator++(three_d op1)
{
  op1.x++;
  op1.y++;
  op1.z++;
  return op1;
}
```

Перегрузка унарных операторов

```
1.  class three_d {
2.    int x, y, z; // 3-D coordinates
3.  public:
4.    three_d() { x = y = z = 0; }
5.    three_d(int i, int j, int k) {x = i; y = j; z = k; }
6.    friend three_d operator+(three_d op1, three_d op2);
7.    three_d operator=(three_d op2);
8.    friend three_d operator++(three_d &op1);
9.    friend three_d operator++(three_d &op1, int notused);
10.   void show() ;
11. } ;
12. three_d operator++(three_d &op1) {
13.   op1.x++;
14.   op1.y++;
15.   op1.z++;
16.   return op1;
17. }
18. three_d operator++(three_d &op1, int notused) {
19.   three_d temp = op1;
20.   op1.x++;
21.   op1.y++;
22.   op1.z++;
23.   return temp;
24. }
```

Перегрузка оператора присваивания

```
1.  class sample {
2.    char *s;
3.  public:
4.    sample() { s = 0; }
5.    sample(const sample &ob) {  s = new char[strlen(ob.s)+1];
6.                                strcpy(s, ob.s);}
7.    ~sample( ) { if(s) delete [] s;}
8.    void show() { cout << s << "\n"; }
9.    void set(char *str) {  s = new char[strlen(str)+1];
10.                          strcpy(s, str);}
11.   };
12.   sample input(){
13.     char instr[80];
14.     sample str;
15.     cout << "Enter a string: ";   cin >> instr;
16.     str.set(instr);
17.     return str;
18.   }
19.   void main() {
20.     sample ob;
21.     ob = input();  // ошибка!!!!
22.     ob.show();
23.   }
```

Перегрузка оператора присваивания

- Решение проблемы: перегрузка оператора присваивания:

```
1. sample sample::operator=(sample &ob)
2. {
3.     if(strlen(ob.s) > strlen(s)) {
4.         delete [ ] s;
5.         s = new char[strlen(ob.s)+1];
6.     }
7.     strcpy(s, ob.s);
8.     return *this;
9. }
```

Перегрузка оператора [ ]

- Оператор [ ] перегружается как бинарный
- Только для класса
- Только как член класса
- Общий вид:
  - тип имя_класса::operator[](int индекс) {…}
- Для того, чтобы выражение с [] могло стоять слева от оператора присваивания, операторная функция должна возвращать ссылку на тип:
  - тип& имя_класса::operator[](int индекс) {…}

Пример: безопасный массив

```
1.  const int SIZE = 3;
2.  class atype {
3.    int a[SIZE];
4.  public:
5.    atype() {  register int i;  for(i=0; i<SIZE; i++)
    a[i] = i;  }
6.    int &operator[](int i);
7.  };
8.  int &atype::operator[](int i) {
9.    if( i<0 || i> SIZE-1) {
10.      cout << "\nIndex out-of-bounds.\n";  exit(1);  }
11.   return a[i];
12. }
13. void main() {
14.   atype ob;
15.   cout << ob[2]; cout << " ";
16.   ob[2] = 25;
17.   cout << ob[2];
18.   ob[3] = 44; // ошибка!!
19. }
```

Перегрузка оператора ( )

```
1.  class three_d {
2.    int x, y, z; // 3-D coordinates
3.  public:
4.    three_d() { x = y = z = 0; }
5.    three_d(int i, int j, int k) {x = i; y = j; z = k; }
6.    three_d operator()(int a, int b, int c);
7.    void show() {   cout << x << ", ";  cout << y << ", ";
    cout << z << "\n";}
8.  };
9.  three_d three_d::operator()(int a, int b, int c) {
10.   three_d temp;
11.   temp.x = x + a;   temp.y = y + b;   temp.z = z + c;
12.   return temp;
13. }
14. int main() {
15.   three_d ob1(1, 2, 3), ob2;
16.   ob2 = ob1(10, 11, 12); // вызов operator()
17.   cout << "ob1: ";   ob1.show();
18.   cout << "ob2: ";   ob2.show();
19.   return 0;
20. }
```

Практическое занятие

- Реализация карточной игры