Η ΓΛΩΣΣΑ C++

Μάθημα 6:

Κλάσεις και Υπερφόρτωση Τελεστών

Δημήτρης Ψούνης

www.psounis.gr

Περιεχόμενα Μαθήματος

Α. Νέα Στοιχεία της C++

1. Υπερφόρτωση Τελεστών

1. Γενικά

- Η υπερφόρτωση τελεστών είναι ένα σημαντικό στοιχείο της C++
  - Μπορούμε να υπερφορτώσουμε συνηθισμένους αριθμητικούς τελεστές όπως το +, το -, το * και το /
  - Αλλά και τελεστές όπως το ++ και το --
    - ώστε να προσθέσουμε λειτουργικότητα σε μία κλάση.
    - π.χ. να κατασκευάσουμε μία κλάση «μιγαδικός» μέσω της οποίας να μπορούμε να προσθέτουμε δύο μιγαδικούς αριθμούς, γράφοντας απλά a+b
    - αλλά και να γράφουμε στην κλάση «μάγος» που κατασκευάσαμε σε προηγούμενο μάθημα a+=10 και να αυξάνει το δείκτη μαγείας του κατά 10.
- Μπορούμε
  - να υπερφορτώσουμε στις κλάσεις μας (σχεδόν) οποιοδήποτε τελεστή της C++
  - όπως ο <<, ο [ ] (για πίνακες)
- Δεν μπορούμε:
  - Να υπερφορτώσουμε τελεστές για ενσωματωμένους τύπους δεδομένων
  - Δεν μπορούμε π.χ. να επαναορίσουμε το + όταν προστίθενται δύο πραγματικοί (double)

A. Νέα Στοιχεία της C++

2. Υπερφόρτωση Διθέσιων Αριθμητικών Τελεστών

1. Ο τελεστής +

- Κατασκευάζουμε μία στοιχειώδη κλάση «μιγαδικός», η οποία έχει ως μέλη δύο πραγματικές μεταβλητές που απεικονίζουν το πραγματικό (real) και το φανταστικό (imag) μέρος
- Βλέπουμε μια στοιχειώδη υλοποίηση με constructor και accessors:

```
class complex {
 public:
   complex();
   complex(double in_real, double in_imag);
   void set_real(double in_real);
   void set_imag(double in_imag);
   double get_real() const;
   double get_imag() const;
 private:
   double real
   double imag;
};
```

```
complex::complex()
{
   real = 0.0;
   imag = 0.0;
}
complex::complex(double in_real, double in_imag)
{
   real = in_real;
   imag = in_imag;
}
void complex::set_real(double in_real)
{
   real = in_real;
}
void complex::set_imag(double in_imag)
{
   imag = in_imag;
}
double complex::get_real() const
{
   return real;
}
double complex::get_imag() const
{
   return imag;
}
```

A. Νέα Στοιχεία της C++

2. Υπερφόρτωση Διθέσιων Αριθμητικών Τελεστών

1. Ο τελεστής +

- Για να προσθέσουμε δύο μιγαδικούς (π.χ. τους a και b) σε ένα νέο μιγαδικό c, θα έπρεπε να γράψουμε έναν κώδικα κάπως έτσι:

```
c.set_real(a.get_real()+b.get_real());
c.set_imag(a.get_imag()+b.get_imag());
```

- Ωστόσο, υπερφορτώνοντας τον τελεστή + θα μπορούμε να γράψουμε απευθείας τον πολύ κομψότερο κώδικα:

```
c = a + b;
```

A. Νέα Στοιχεία της C++

2. Υπερφόρτωση Διθέσιων Αριθμητικών Τελεστών

1. Ο τελεστής +

Η υπερφόρτωση του διθέσιου αριθμητίκού τελεστή + γίνεται
- Γράφοντας ως μέθοδο στην κλάση την:
  - **class_name operator+ (const class_name &right);**

π.χ. στην κλάση complex ενσωματώνουμε στην κλάση την δήλωση:
- complex operator+(const complex &right);

Ενώ στο σώμα της μεθόδου:
- και δεδομένου ότι η πρόσθεση γίνεται μεταξύ δύο αντικειμένων (π.χ. **left+right**)
- Η μέθοδος καλείται από το left με όρισμα το right
- Είναι σαν να κανουμε την κλήση **left.operator+(right)**
- και επιστρέφει ένα καινούργιο αντικείμενο, το οποίο είναι το αποτέλεσμα της πράξης.

Στο παράδειγμα το σώμα της συνάρτησης είναι:

```
complex complex::operator+ (const complex &right)
{
   complex result;

   result.real = real+right.real;
   result.imag = imag+right.imag;
   return result;
}
```

A. Νέα Στοιχεία της C++

2. Υπερφόρτωση Διθέσιων Αριθμητικών Τελεστών

1. Ο τελεστής +

Ενσωματώνουμε τον κώδικα της υπερφόρτωσης του +, και βλέπουμε ολοκληρωμένο το παράδειγμα:

```
/*cpp6.operator_plus.cpp Υπερφόρτωση του + */

#include <iostream>
using namespace std;

class complex {
 public:
   complex();
   complex(double in_real, double in_imag);
   void set_real(double in_real);
   void set_imag(double in_imag);
   double get_real() const;
   double get_imag() const;
   complex operator+ (const complex &right);
 private:
   double real
   double imag;
};
```

```
int main()
{
   complex a(1.0,1.0);
   complex b(2.0,3.0);

   complex c;

   c = a + b;

   cout<<c.get_real()<<" "<<c.get_imag();

   return 0;
}
```

```
...
complex complex::operator+ (const complex &right)
{
   complex result;

   result.real = real+right.real;
   result.imag = imag+right.imag;
   return result;
}
```

Α. Νέα Στοιχεία της C++

2. Υπερφόρτωση Διθέσιων Αριθμητικών Τελεστών

1. Ο τελεστής +

- Μπορούμε να υπερφορτώσουμε τον τελεστή + για να δουλεύει με οποιονδήποτε τύπο δεδομένων στο δεξί μέρος.
- Π.χ. να δουλεύει σωστά αν κάνουμε μια πράξη μεταξύ μιγαδικού και ακεραίου:

```
c = a + 5;
```

  - Θα δούμε πως γίνεται η υπερφόρτωση ώστε να γίνεται πρόσθεση με ακέραιο.

- Προσοχή! Δεν μπορούμε να κάνουμε υπερφόρτωση με τον τρόπο που μάθαμε και να εκτελείται μία πράξη ως

```
c = 5 + a;
```

  - Ωστόσο μέσω των φιλικών συναρτήσεων (επόμενο μάθημα) θα μπορούμε να κάνουμε και αυτήν την υπερφόρτωση.

A. Νέα Στοιχεία της C++

2. Υπερφόρτωση Διθέσιων Αριθμητικών Τελεστών

1. Ο τελεστής +

Η υπερφόρτωση του διθέσιου αριθμητίκού τελεστή + με ακέραιο γίνεται

- Γράφοντας ως μέθοδο στην κλάση την:
  - **class_name operator+ (int right);**

π.χ. στην κλάση complex ενσωματώνουμε στην κλάση την δήλωση:

- complex operator+(int right);

Ενώ στο σώμα της μεθόδου:

- και δεδομένου ότι η πρόσθεση γίνεται μεταξύ αντικειμένου και ακεραίου(π.χ. **left+right**)
- Η μέθοδος καλείται από το left με όρισμα το right
- Είναι σαν να κανουμε την κλήση **left.operator+(right)**
- και επιστρέφει ένα καινούργιο αντικείμενο, το οποίο είναι το αποτέλεσμα της πράξης.

Στο παράδειγμα το σώμα της συνάρτησης είναι:

```
complex complex::operator+ (int right)
{
   complex result;

   result.real = real+right;
   result.imag = imag+right;
   return result;
}
```

A. Νέα Στοιχεία της C++

2. Υπερφόρτωση Διθέσιων Αριθμητικών Τελεστών

1. Ο τελεστής +

Ενσωματώνουμε τον κώδικα της υπερφόρτωσης του +, και βλέπουμε ολοκληρωμένο το παράδειγμα:

```
/*cpp6.operator_plus_int.cpp Υπερφόρτωση του + */

#include <iostream>
using namespace std;

class complex {
 public:
   …
   complex operator+ (const complex &right);
   complex operator+ (int right);
 private:
   double real
   double imag;
};
```

```
int main()
{
   complex a(1.0,1.0);
   complex b(2.0,3.0);

   complex c;

   c = a + 5;

   cout<<c.get_real()<<" "<<c.get_imag();

   return 0;
}
```

```
...
complex complex::operator+ (int right)
{
   complex result;

   result.real = real+right;
   result.imag = imag+right;
   return result;
}
```

A. Νέα Στοιχεία της C++

2. Υπερφόρτωση Διθέσιων Αριθμητικών Τελεστών

2. Ο τελεστής -

Η υπερφόρτωση του διθέσιου αριθμητίκού τελεστή - γίνεται με ίδιο τρόπο

- Γράφοντας ως μέθοδο στην κλάση την:
  - **class_name operator- (const class_name &right);**

π.χ. στην κλάση complex ενσωματώνουμε στην κλάση την δήλωση:

- complex operator-(const complex &right);

Ενώ στο σώμα της μεθόδου:

- και δεδομένου ότι η πρόσθεση γίνεται μεταξύ δύο αντικειμένων (π.χ. **left-right**)
- Η μέθοδος καλείται από το left με όρισμα το right
- Είναι σαν να κανουμε την κλήση **left.operator-(right)**
- και επιστρέφει ένα καινούργιο αντικείμενο, το οποίο είναι το αποτέλεσμα της πράξης.

Στο παράδειγμα το σώμα της συνάρτησης είναι:

```
complex complex::operator- (const complex &right)
{
   complex result;

   result.real=real-right.real;
   result.imag=imag-right.imag;
   return result;
}
```

A. Νέα Στοιχεία της C++

2. Υπερφόρτωση Διθέσιων Αριθμητικών Τελεστών

2. Ο τελεστής -

- Εντελώς αντίστοιχη είναι η υπερφόρτωση και των άλλων διθέσιων τελεστών (*,/,%)
  - Βλέπουμε την υπερφόρτωση του -

```
/*cpp6.operator.plus.cpp Υπερφόρτωση του + */

#include <iostream>
using namespace std;

class complex {
 public:
   …
   complex operator+ (const complex &right);
   complex operator+ (int right);
   complex operator- (const complex &right);
 private:
   …
};
```

```
int main()
{
   complex a(1.0,1.0);
   complex b(2.0,3.0);

   complex c;

   c = a - b;

   cout<<c.get_real()<<" "<<c.get_imag();

   return 0;
}
```

```
…
complex complex::operator- (const complex &right)
{
   complex result;

   result.real=real-right.real;
   result.imag=imag-right.imag;

   return result;
}
```

A. Νέα Στοιχεία της C++

2. Υπερφόρτωση Διθέσιων Αριθμητικών Τελεστών

3. Παρατηρήσεις

- Βλέπουμε και έναν εναλλακτικό συντομότερο τρόπο για να κάνουμε την ίδια δουλειά:
- Π.χ. το σώμα του τελεστή + για την κλάση μπορεί να γραφεί αντί για τον τρόπο που είδαμε:

```
complex complex::operator+ (const complex &right)
{
   complex result;

   result.real=real+right.real;
   result.imag=imag+right.imag;

   return result;
}
```

- ως εξής:

```
complex complex::operator+ (const complex &right)
{
   return complex(real+right.real, imag+right.imag);
}
```

  - όπου επιστρέφεται ένα νέο αντικείμενο, κατάλληλα αρχικοποιημένο από τον κατασκευαστή.

Α. Νέα Στοιχεία της C++

2. Υπερφόρτωση Διθέσιων Αριθμητικών Τελεστών

3. Παρατηρήσεις

- Με τον ίδιο τρόπο υπερφορτώνουμε οποιονδήποτε διθέσιο αριθμητικό τελεστή και μπορούμε να κάνουμε πράξεις μεταξύ των αντικειμένων (+,-,*,/,%)
- Αλλά και να υπερφορτώνουμε τους τελεστές ώστε να δουλεύουν:
  - Με αριστερό μέρος οπωσδήποτε αντικείμενο της κλάσης
  - Με δεξί μέρος όποιον τύπο δεδομένων θέλουμε
- Στο επόμενο μάθημα θα δούμε και υπερφόρτωση
  - ώστε να δουλεύουν με αριστερό μέρος όποιον τύπο δεδομένων θέλουμε
  - και δεξί μέρος αντικείμενο κλάσης.

Α. Νέα Στοιχεία της C++

3. Υπερφόρτωση Μονοθέσιων Αριθμητικών Τελεστών

1. Ο μεταθεματικος τελεστής ++

- Ξέρουμε ότι ο τελεστής ++ (μεταθεματικός):
  - Πρώτα εκτελεί την εντολή στην οποία ενσωματώνεται.
  - Έπειτα αυξάνει κατά 1, τη μεταβλητή στην οποία επενεργεί.
- Π.χ. οι εντολές

```
int x=1;
cout<<x++;
cout<<x;
```

- Θα τυπώσουν:

```
1
2
```

- Η υπερφόρτωσή μας θα πρέπει να υποστηρίζει και τη συνήθη λειτουργία:

```
a = x++;
```

  - όπου με το πέρας της εκτέλεσης, θα πρέπει x=2 και a=1

- Θα δούμε πως μπορούμε να υπερφορτώσουμε τον τελεστή ++ στην κλάση μιγαδικού ώστε να προσθέτει μια μονάδα και στο πραγματικό και στο φανταστικό μέρος.

A. Νέα Στοιχεία της C++

3. Υπερφόρτωση Μονοθέσιων Αριθμητικών Τελεστών

1. Ο μεταθεματικός τελεστής ++

Η υπερφόρτωση του μονοθέσιου μεταθεματικού τελεστή ++ γίνεται

- Γράφοντας ως μέθοδο στην κλάση την:
  - **class_name operator++ (int right);**

π.χ. στην κλάση complex ενσωματώνουμε στην κλάση την δήλωση:

- const complex operator++(int right);

- το όρισμα right δεν θα χρησιμοποιηθεί (σηματοδοτεί μόνο ότι είναι μεταθεματικός)
- και δεδομένου ότι ο τελεστής επενεργεί πάνω σε ένα αντικείμενο (π.χ. **left++**)
- Είναι σαν να κανουμε την κλήση **left.operator++(right)**
- Πρέπει:
  - Να αυξάνει κατά ένα το αντικείμενο left
  - Να δημιουργεί ένα νέο αντικείμενο και να το επιστρέφει με την αρχική τιμή του left

Στο παράδειγμα το σώμα της συνάρτησης είναι:

```
const complex complex::operator++ (int right)
{
   complex result = *this;
   this->real++;
   this->imag++;
   return result;
}
```

A. Νέα Στοιχεία της C++

3. Υπερφόρτωση Μονοθέσιων Αριθμητικών Τελεστών

1. Ο μεταθεματικός τελεστής ++

- Εντελώς αντίστοιχη είναι η υπερφόρτωση και των άλλων διθέσιων τελεστών (*,/,%)
  - Βλέπουμε την υπερφόρτωση του -

```
/* cpp6.operator_plusplus_post.cpp Υπερφόρτωση του ++ (μεταθεματικός) */

#include <iostream>
using namespace std;

class complex {
 public:
   …
   complex operator+ (const complex &right);
   complex operator- (const complex &right);
   complex operator+ (int right);
   complex operator++(int right);
 private:
   …
};
```

```
int main()
{
   complex a(1.0,1.0);
   complex c;

   c=a++;

   cout<<"a="<<a.get_real()<<","<<a.get_imag()<<endl;
   cout<<"c="<<c.get_real()<<","<<c.get_imag();

   return 0;
}
```

```
…
complex complex::operator++(int right)
{
   complex result=*this;

   real++;
   imag++;

   return result;
}
```

Α. Νέα Στοιχεία της C++

3. Υπερφόρτωση Μονοθέσιων Αριθμητικών Τελεστών

2. Ο προθεματικος τελεστής ++

- Ξέρουμε ότι ο τελεστής ++ (προθεματικός):
  - Πρώτα αυξάνει κατά 1, τη μεταβλητή στην οποία επενεργεί.
  - Μετά εκτελεί την εντολή στην οποία ενσωματώνεται.
- Π.χ. οι εντολές

```
int x=1;
cout<<++x;
cout<<x;
```

- Θα τυπώσουν:

```
2
2
```

- Η υπερφόρτωσή μας θα πρέπει να υποστηρίζει και τη συνήθη λειτουργία:

```
a = ++x;
```

  - όπου με το πέρας της εκτέλεσης, θα πρέπει x=2 και a=2

- Θα δούμε πως μπορούμε να υπερφορτώσουμε τον τελεστή ++ στην κλάση μιγαδικού ώστε να προσθέτει μια μονάδα και στο πραγματικό και στο φανταστικό μέρος.

A. Νέα Στοιχεία της C++

3. Υπερφόρτωση Μονοθέσιων Αριθμητικών Τελεστών

2. Ο προθεματικός τελεστής ++

Η υπερφόρτωση του μονοθέσιου προθεματικού τελεστή ++ γίνεται

- Γράφοντας ως μέθοδο στην κλάση την:
  - **class_name &operator++ ();**

π.χ. στην κλάση complex ενσωματώνουμε στην κλάση την δήλωση:

- complex &operator++();

Στο σώμα της μεθόδου:

- δεδομένου ότι ο τελεστής επενεργεί πάνω σε ένα αντικείμενο (π.χ. **++left**)
- Είναι σαν να κανουμε την κλήση **left.operator++()**
- Πρέπει:
  - Να αυξάνει κατά ένα το αντικείμενο left
  - Να επιστρέφει αναφορά στο ίδιο το αντικείμενο

Στο παράδειγμα το σώμα της συνάρτησης είναι:

```
complex &complex::operator++ ()
{
   real++;
   imag++;
   return *this;
}
```

A. Νέα Στοιχεία της C++

3. Υπερφόρτωση Μονοθέσιων Αριθμητικών Τελεστών

2. Ο προθεματικός τελεστής ++

- Βλέπουμε την υπερφόρτωση

```
/*cpp6.operator_plusplus_pre.cpp Υπερφόρτωση του ++ (προθεματικός) */

#include <iostream>
using namespace std;

class complex {
 public:
   …
   complex operator+ (const complex &right);
   complex operator+ (int right);
   complex operator- (const complex &right);
   complex operator++(int right);
   complex &operator++();
 private:
   …
};
```

```
int main()
{
   complex a(1.0,1.0);
   complex c;

   c=++a;

   cout<<"a="<<a.get_real()<<","<<a.get_imag()<<endl;
   cout<<"c="<<c.get_real()<<","<<c.get_imag();

   return 0;
}
```

```
…
complex &complex::operator++()
{
   real++;
   imag++;

   return *this;
}
```

Α. Νέα Στοιχεία της C++

3. Υπερφόρτωση Μονοθέσιων Αριθμητικών Τελεστών

3. Παρατηρήσεις

- Η κύρια διαφορά των δύο τελεστών είναι:
  - Στον μεταθεματικό τελεστή (x++):
    - Πρέπει να επιστρέφουμε άλλο (καινούργιο) αντικείμενο, το οποίο κρατάει την προηγούμενη τιμή.
    - Γι' αυτό και δεν υπάρχει αναφορά στην επιστροφή
  - Στον προθεματικό τελεστή (++x):
    - Επιστρέφουμε το ίδιο το αντικείμενο, μιας και η αύξηση γίνεται αρχικά
    - Χρησιμοποιούμε αναφορά για να μην δημιουργηθεί νέο αντικείμενο (κατά την επιστροφή) και να γλιτώσουμε αυτόν τον φόρτο εκτέλεσης
- Ο διαχωρισμός των δύο τελεστών γίνεται με βάση το όρισμα που δέχεται.
  - Χωρίς όρισμα είναι ο προθεματικός.
  - Με όρισμα ακέραιο είναι ο μετάθεματικος.
    - Έχει οριστεί έτσι από τη γλώσσα, δεν θα δούλευε π.χ. αν θέταμε όρισμα έναν πραγματικό.
    - Δεν είναι και ο πιο «φυσικός» τρόπος, αλλά με κάποιον τρόπο έπρεπε να γίνει αυτός ο διαχωρισμός.
- Εντελώς αντίστοιχα μπορούμε να κάνουμε και υπερφόρτωση του τελεστή --

A. Νέα Στοιχεία της C++

4. Υπερφόρτωση του τελεστή ανάθεσης

1. Εξήγηση του =

- Ο τελεστής ανάθεσης
  - Αντιγράφει το αποτέλεσμα της έκφρασης που είναι στο δεξί μέρος του στο αριστερό.

- Π.χ. οι εντολές:

```
int a, b;
a=1;
b=a;
```

- Έχουν δύο εφαρμογές του τελεστή ανάθεσης:
  - Εδώ θα ασχοληθούμε μόνο με τη 2η, όπου ανατίθεται σε μια μεταβλητή, μία άλλη του ίδιου τύπου δεδομένων
- Και μια «κρυφή» λειτουργία
  - Η εντολή b=a επιστρέφει το αριστερό μέρος της (το b)
  - Γι' αυτό μπορούμε να γράφουμε διαδοχικές εντολές ανάθεσεις π.χ.

```
a=b=c;
```

A. Νέα Στοιχεία της C++

4. Υπερφόρτωση του τελεστή ανάθεσης

2. Υπερφόρτωση του =

Η υπερφόρτωση του τελεστή ανάθεσης γίνεται
- Γράφοντας ως μέθοδο στην κλάση την:
  - **class_name &operator= (const class_name &right);**

π.χ. στην κλάση complex ενσωματώνουμε στην κλάση την δήλωση:
- complex &operator=(const complex &right);

Στο σώμα της μεθόδου:
- δεδομένου ότι ο τελεστής επενεργεί πάνω σε δύο αντικείμενο (π.χ. **left=right**)
- Είναι σαν να κανουμε την κλήση **left.operator=(right)**
- Πρέπει:
  - Να αντιγράφει το αντικείμενο right στο αντικείμενο left.
  - Να λαμβάνει υπόψη αν το όρισμα είναι ίδιο με τον εαυτό του.

Στο παράδειγμα το σώμα της συνάρτησης είναι:

```
complex &complex::operator= (const complex &right)
{
   if (this==&right) return *this;
   real=right.real;
   imag=right.imag;
   return *this;
}
```

A. Νέα Στοιχεία της C++

4. Υπερφόρτωση του τελεστή ανάθεσης

2. Υπερφόρτωση του =

- Βλέπουμε την υπερφόρτωση του = στην κλάση των μιγαδικών αριθμών:

```
/*cpp6.operator_equal.cpp Υπερφόρτωση του = */

#include <iostream>
using namespace std;

class complex {
 public:
   …
   complex operator+ (const complex &right);
   complex operator+ (int right);
   complex operator- (const complex &right);
   complex operator++(int right);
   complex &operator++();
   complex &operator= (const complex &right);
 private:
   …
};
```

```
int main()
{
   complex a(1.0,1.0);
   complex c;

   c=a;

   cout<<"a="<<a.get_real()<<","<<a.get_imag()<<endl;
   cout<<"c="<<c.get_real()<<","<<c.get_imag();

   return 0;
}
```

```
…
complex &complex::operator= (const complex &right)
{
   if (this==&right) return *this;

   real=right.real;
   imag=right.imag;

   return *this;
}
```

Α. Νέα Στοιχεία της C++

4. Υπερφόρτωση του τελεστή ανάθεσης

2. Υπερφόρτωση του =

- Αντίστοιχα μπορούμε να υπερφορτώσουμε τον τελεστή = ώστε να δουλεύει με δεξί μέρος οποιονδήποτε τύπο δεδομένων.
- Η δήλωση της μεθόδου θα είναι (π.χ. για ακέραιο):

```
complex &operator=(int right);
```

- και στο σώμα βάζουμε τις επιθυμητές ενέργειες, π.χ.

```
complex &complex::operator= (int right)
{
   real=right;
   imag=right;
   return *this;
}
```

A. Νέα Στοιχεία της C++

5. Υπερφόρτωση του [ ]

1. Υπερφόρτωση του [ ]

Η υπερφόρτωση του τελεστή δείκτη πίνακα [ ] γίνεται
- Γράφοντας ως μέθοδο στην κλάση την:
  - **type &operator[ ] (int i);**

π.χ. στην κλάση ARRAY που κατασκευάσαμε (μάθημα 4: άσκηση 2) η δήλωση θα είναι
- int &operator[ ] (int index);
  - Δέχεται ως όρισμα το δείκτη της θέσης του πίνακα
  - και επιστρέφει αναφορά στο στοιχείο που βρίσκεται αποθηκευμενο σε αυτήν την θέση.

Στο σώμα της μεθόδου:
- δεδομένου ότι ο τελεστής επενεργεί πάνω σε ένα αντικείμενο (π.χ. **arr[i]**)
- Είναι σαν να κανουμε την κλήση **arr.operator[ ](i)**

Στο παράδειγμα το σώμα της συνάρτησης είναι:

```
int &ARRAY::operator[] (int i)
{
   if (i>=0 && i<n)
      return p[i];
   else
      cout<<"Index Error";
}
```

A. Νέα Στοιχεία της C++

5. Κάνοντας την κλάση πίνακα

1. Υπερφόρτωση του [ ]

- Βλέπουμε την κλάση ARRAY στη νέα της έκδοση:

```
/* cpp6.array_with_index_overloading.cpp*/

#include <iostream>
using namespace std;

class ARRAY {
 public:
   ARRAY(int in_n);
   ARRAY(const ARRAY &ob);
   ~ARRAY();
   int get_n() const;
   int &operator[] (int i);
   void print();
 private:
   int *p;
   int n;
};
```

```
int main()
{
   int n=10;
   ARRAY pin(n);

   for (int i=0; i<n; i++)
      pin[i]=i*i;

   pin.print();

   return 0;
}
```

```
...
int &ARRAY::operator[] (int i)
{
   if (i>=0 && i<n)
      return p[i];
   else
      cout<<"Index Error";
}
```

A. Νέα Στοιχεία της C++

5. Κάνοντας την κλάση πίνακα

2. Παρατηρήσεις

- Παρατηρήστε ότι στο παράδειγμα της κλάσης ARRAY, έχουν αντικατασταθεί οι accessors get_i και set_i με την πρόσβαση μέσω των [ ].
- Η κλάση μας είναι σχεδόν έτοιμη. Είναι ένας περιτυλιχτής των πινάκων της C:
  - Το πρόγραμμα πλέον δεν «σκάει» όταν γίνεται πρόσβαση εκτός του πίνακα (κλασσικό προγραμματιστικό λάθος)
  - Το μόνο που απομένει είναι να κάνουμε μια πιο αξιοπρεπή διαχείριση των λαθών
    - μέσω των «εξαιρέσεων» που θα μάθουμε σε επόμενο μάθημα.

Α. Νέα Στοιχεία της C++

6. Άλλες Υπερφορτώσεις

1. Διαθέσιμοι Τελεστές

- Μπορούμε να υπερφορτώσουμε οποιονδήποτε από τους γνωστούς τελεστές της C++:
  - π.χ. τους +=, -=
  - τους && και ||
  - τους ^ και &
  - κ.α.
- και δεν μπορούμε να υπερφορτώσουμε:
  - τους ::, .*, . και ?:

Α. Νέα Στοιχεία της C++

6. Άλλες Υπερφορτώσεις

2. Σύνοψη και επεκτάσεις

- Σε αυτό το μάθημα:
  - Είδαμε πως κάνουμε υπερφορτώσεις όπου ο τελεστής:
    - είτε επενεργεί σε ένα αντικείμενο
    - είτε κάνει μία ενέργεια μεταξύ δύο αντικειμένων
    - είτε κάνει μια ενέργεια μεταξύ ενός αντικειμένου (αριστερά) και άλλου ΤΔ (δεξιά)
- Στο επόμενο μάθημα θα δούμε πως μπορούμε:
  - Να κάνουμε πράξεις μεταξύ μιας μεταβλητής κάποιου τύπου δεδομένων (αριστερά) και ενός αντικειμένου (δεξιά)
  - Και να υπερφορτώσουμε π.χ το + για να γράφουμε:

```
5 + a;
```

- Αλλά και να υπερφορτώσουμε τον τελεστή << ώστε να γράφουμε:

```
cout<<a;
```

Β. Ασκήσεις

Άσκηση 1.1: Επέκταση της κλάσης STRING με τον τελεστή =

Επεκτείνετε την κλάση STRING της άσκησης 3 του «Μάθημα 4: Κλάσεις και Αναφορές» με υπερφόρτωση του τελεστή =.

Β. Ασκήσεις

Άσκηση 1.2: Επέκταση της κλάσης STRING με τον τελεστή [ ]

Επεκτείνετε την κλάση STRING της προηγούμενης άσκησης, ώστε να υποστηρίζει τον τελεστή [], επιστρέφοντας το χαρακτήρα που βρίσκεται στην αντίστοιχη θέση του πίνακα χαρακτήρων.

Β. Ασκήσεις

Άσκηση 1.3: Επέκταση της κλάσης STRING με <,>,==

Επεκτείνετε την κλάση STRING της προηγούμενης άσκησης, ώστε να υποστηρίζει τους τελεστές <,>,== οι οποίοι επιστρέφουν true/false, συγκρίνοντας δύο συμβολοσειρές.

Β. Ασκήσεις

Άσκηση 2: Επέκταση της κλάσης ARRAY με τον τελεστή +=

Επεκτείνετε την κλάση ARRAY όπως την αφήσαμε στο μάθημα όταν είδαμε τον τελεστή [ ]:

- με τον τελεστή =
- με τον τελεστή +=
  - Συγκεκριμένα, επιθυμούμε χρήση όπως η ακόλουθη:

```
ob += 5;
```

    - Η οποία θα επεκτείνει τις θέσεις του πίνακα κατά 5

Β. Ασκήσεις

Άσκηση 3.1: Επέκταση της κλάσης «Μάγος»

Επεκτείνετε την κλάση Μάγος (Άσκηση 3 του «Μαθήματος 2: Εισαγωγή στις Κλάσεις») με τα εξής στοιχεία:

- Μία ιδωτική μεταβλητή (health) που απεικονίζει την υγεία του μάγου (0-100)
  - Να αρχικοποιείται σε 100.
- Υπερφόρτωση του τελεστή +=, ώστε να αυξάνει την υγεία του μάγου με τον ακέραιο που ακολουθεί τον τελεστή.
  - Να μην υπερβαίνει το 100
- Υπερφόρτωση του τελεστή -=, ώστε να μειώνει την υγεία του μάγου με τον ακέραιο που ακολουθεί τον τελεστή.
  - Αν η υγεία πέσει κάτω από 0, να τυπώνει ένα μήνυμα «Wizard Dead!»
- Τροποποίηση των μεθόδων:
  - Lightning: να επιστρέφει έναν ακέραιο από 10 έως 20
  - Fireball: να επιστρέφει έναν ακέραιο από 50 έως 70

Ελέγξτε την ποιότητα της υλοποίησης σας με ένα κατάλληλο σενάριο που θα σκεφτείτε.

Σημείωση: Για την παραγωγή τυχαίων αριθμών, συμβουλευτείτε το βίντεο «Γλώσσα C, Μάθημα 7, Βίντεο Θεωρίας 4/4: Τυχαίοι Αριθμοί»

Β. Ασκήσεις

Άσκηση 3.2: Κλάση Ανθρωποειδές (humanoid)

Ορίστε στο ίδιο πρόγραμμα με το Μάγο, την κλάση Ανθρωποειδές (humanoid) η οποία μοντελοποιεί ένα αναλώσιμο ζωντανό, το οποίο θα πολεμήσει το μάγο. Ορίστε:

- Μία ιδωτική μεταβλητή (health) που απεικονίζει την υγεία του μάγου (0-100)
  - Να αρχικοποιείται σε 100 μέσω default κατασκευαστή.
- Υπερφόρτωση του τελεστή -=, ώστε να μειώνει την υγεία του ανθρωποειδούς με τον ακέραιο που ακολουθεί τον τελεστή.
  - Αν η υγεία πέσει κάτω από 0, να τυπώνει ένα μήνυμα «Humanoid Dead!»
- Να έχει μία δημόσια μέθοδο attack:
  - Η οποία θα επιστρέφει έναν τυχαίο αριθμό ζημιάς από 1 έως 5.

Β. Ασκήσεις

Άσκηση 3.3: Ανθρωποειδές εναντίον Μάγου

- Προσθέστε στις δύο κλάσεις μία δημόσια μέθοδο (check_dead) η οποία
  - θα ελέγχει αν το αντίστοιχο ον είναι νεκρό
- Και ένα σενάριο γύρων στην συνάρτηση main:
- Επαναληπτικά σε κάθε γύρο:
  - Παίζει πρώτα ο μάγος
    - Επιλέγει τυχαία lightning ή fireball
    - Αφαιρούνται οι πόντοι ζημιάς από την υγεία του ανθρωποειδούς
  - Γίνεται έλεγχος αν πέθανε το ανθρωποειδές (αν ναι, το πρόγραμμα τερματίζεται)
  - Παίζει έπειτα το ανθρωποειδές
    - Κάνει επίθεση
    - Αφαιρούνται οι πόντοι ζημιάς από την υγεία του μάγου
  - Γίνεται έλεγχος αν πέθανε ο μάγος (αν ναι, το πρόγραμμα τερματίζεται)