SAE EDUCATION

Intro to C++ | 1.3

Classes, Structs

Akis Zachariadis - Games Dept. Coordinator

Classes

- A user defined datatype which groups together related pieces of information.
- Examples
  - Name
  - Vector
  - Stats
  - etc

Class Declaration

- Line 6 | Declare class with the name Vector
- Line 7 | Create private variables that are accessible only from the class itself
- Line 9 | Create public variables that are accessible from the script

- By default in C++ classes are private. So we don't need to write the private:

```
5
6   class Vector {
7   private:
8       char* name;
9   public:
10      double xStart;
11      double xEnd;
12      double yStart;
13      double yEnd;
14  };
```

Constructors and Init

- Line 15: Empty Constructor
- Main
- Vector vec -> Create an instance of the Vector class with name vec and call the default/empty constructor
- Print the vec.xStart -> Output: 1

- In the c# we used the keyword new with the type of constructor that we want. Here we just create the class and the init at the same time

```
int main()
{
    Vector vec;
    std::cout << vec.xStart << std::endl;
}
```

```
5   //Class Declaration
6   class Vector {
7   private:
8       const char* name;
9   public:
10      double xStart;
11      double xEnd;
12      double yStart;
13      double yEnd;
14      //Constructor
15      Vector() {
16          name = "Hello";
17          xStart = 1.0,
18          xEnd = 2.0;
19          yStart = 3.0;
20          yEnd = 4.0;
21      }
```

Multiple Constructors

- Line 16 | Default Constructor
- Line 24 | Second Constructor with overloading
- Line 41 | Create and initialize a new instance of vector class with name vec2 using the second constructor
- Line 42 | Printing the name of the second vector

```
16      Vector() {
17          name = "Hello";
18          xStart = 1.0,
19          xEnd = 2.0;
20          yStart = 3.0;
21          yEnd = 4.0;
22      }
23
24      Vector(const char* newName) {
25          name = newName;
26          xStart = 1.0,
27          xEnd = 2.0;
28          yStart = 3.0;
29          yEnd = 4.0;
30      }
31  };
```

```
34  int main()
35  {
36      Vector vec;
37      std::cout << vec.xStart << std::endl;
38
39      const char* Name = "NewVector";
40      //Create a new instance and use the second constructor
41      Vector vec2(Name);
42      std::cout << vec2.name << std::endl;
43  }
```

Default Copy Constructor

- We copy all the values of the vec2 to the variables of vec
- The cout will print now the name of NewVector which was the value of vec2

```
34  int main()
35  {
36      Vector vec;
37      std::cout << vec.xStart << std::endl;
38
39      const char* Name = "NewVector";
40      //Create a new instance and use the second constructor
41      Vector vec2(Name);
42      std::cout << vec2.name << std::endl;
43
44      vec = vec2; //Default copy constructor
45
46      std::cout << vec.name << std::endl;
47
48      return 0;
```

Custom Copy Constructor

- We can create a custom copy instructor in order to customize how we copy one class instance to another.

```
41  int main()
42  {
43      Vector vec;
44      std::cout << vec.xStart << std::endl;
45
46      const char* Name = "NewVector";
47      //Create a new instance and use the second constructor
48      Vector vec2(Name);
49      std::cout << vec2.name << std::endl;
50
51      vec = vec2; //Custom Copy Constructor
52
53      std::cout << vec.name << std::endl;
54      //This will print HelloAgain as we specified on our custom copy contructor
55
56      return 0;
```

```
16      Vector() {
17          name = "Hello";
18          xStart = 1.0,
19          xEnd = 2.0;
20          yStart = 3.0;
21          yEnd = 4.0;
22      }
23
24      Vector(const char* newName) {
25          name = newName;
26          xStart = 1.0,
27          xEnd = 2.0;
28          yStart = 3.0;
29          yEnd = 4.0;
30      }
        Vector(Vector& other) {
32          name = "HelloAgain",
33          xStart = other.xStart;
34          xEnd = other.xEnd;
35          yStart = other.yStart;
36          yEnd = other.yEnd;
37      }
38  };
```

Passing classes to functions

- We can pass our classes as a value to a function
- If we want to manipulate our classes value inside the function we need to pass it as a reference by using the reference type &
- In the first example our program will print 2.0
- In the second example our program will print 4.0

```
44  int main()
45  {
46      Vector vec;
47      GetVector(vec);
48
49      return 0;
50  }
51
52  int GetVector(Vector newVec) {
53      std::cout << newVec.xEnd << std::endl;
54  }
```

```
44  int main()
45  {
46      Vector vec;
47      GetVector(vec);
48
49      return 0;
50  }
51
52  int GetVector(Vector &newVec) {
53      newVec.xEnd += 2;
54      std::cout << newVec.xEnd << std::endl;
55  }
```

Structs

- Same as classes but with some differences
- Structs are by default a by value variable
- Our structs are allocating memory in sequence
- Classes on the other hand are variables that references a template class somewhere in the memory.
- Classes are allocating random positions in the memory

Destructor

- When we create a new class by using the new keyword we allocate space in the heap memory.
- In order to free memory when the class or memory goes out of scope we need to use a destructor.
- Destructor is called when the class instance gets de-allocated.

```
27      Vector(Vector& other) {
28          name = "HelloAgain",
29          xStart = other.xStart;
30          xEnd = other.xEnd;
31          yStart = other.yStart;
32          yEnd = other.yEnd;
33      }
34      ~Vector() {
35          std::cout << "Destructor Invoked" << std::endl;
36      }
37  };
38
41  int main()
42  {
43      Vector* vec = new Vector; //Creating
44      delete vec;
45
46      return 0;
47  }
```

Example #1

Να δημιουργηθεί πρόγραμμα που να δημιουργεί στο runtime έναν integer array με μέγεθος που θα δίνεται από τον χρήστη.

Μόλις δημιουργηθεί από τον χρήστη να δημιουργείτε ένας πίνακας ακόμα και να γίνεται αντιγραφή του πρώτου στο δεύτερο.

Για την δημιουργία του πίνακα να χρησιμοποιείτε μια κλάση με έναν constructor.

Challenge #3

Για την συγκεκριμένη άσκηση θα μεταβάλετε το πρόγραμμα phonebook της προηγούμενης εβδομάδας έτσι ώστε να λειτουργεί αντικειμενοστρεφώς.

Συγκεκριμένα, θα δημιουργήσετε κλάση Person η οποία θα περιέχει το όνομα και το τηλέφωνο του κάθε ατόμου και θα αντικαταστήσετε τους πίνακες string[] και int[] με έναν μοναδικό πίνακα Person[] ο οποίος θα περιέχει όλες τις καταχωρήσεις αλλά σε μορφή αντικειμένων της κλάσης Person σε αυτή την περίπτωση.

Ο υπόλοιπος κώδικας θα προσαρμοστεί ώστε να είναι συμβατός με αυτή την βασική αρχή. Κατά τα άλλα, η λειτουργία του προγράμματος και η είσοδος/έξοδός του δεν θα αλλάξουν καθόλου σε σχέση με την μη αντικειμενοστρεφή έκδοση.

Challenge #4

Για την άσκηση αυτή θα δημιουργήσετε την βασική υποδομή ενός RPG σε μορφή console εφαρμογής και, πιο συγκεκριμένα, μόνο την λειτουργικότητα μάχης μεταξύ δύο παικτών όπως αυτή περιγράφεται παρακάτω (combat simulator).

Αναλυτικότερα ο κάθε παίκτης θα πρέπει να περιέχει τα εξής χαρακτηριστικά:

- Όνομα
- Health μεταξύ του διαστήματος [800,1000]
- Attack Power μεταξύ του διαστήματος [100, 300]
- Defence Rating μεταξύ του διαστήματος [0.25, 1]

Τα στατιστικά, εκτός από το όνομα, θα υπολογίζονται αυτόματα από το πρόγραμμα. Το όνομα του κάθε ήρωα θα δίνεται από τον χρήστη.

Το πρόγραμμα θα περιέχει μια μέθοδο Attack μέσω της οποίας ο κάθε παίκτης θα επιτίθεται στον άλλον παίκτη. Η μέθοδος θα μειώνει κατάλληλα το health του αμυνόμενου παίκτη κατά ένα ποσό damage τύπου int το οποίο και θα επιστρέφει για σκοπούς εκτύπωσης από το πρόγραμμα στην κονσόλα. Ο υπολογισμός του damage γίνεται ως εξής:

Έστω r1 και r2 τυχαίοι αριθμοί στο διάστημα [0, 1]

- a = AttackPower * r 1
- d = 1 – (DefenceRating * r 2)
- damage = a * d

Βασικές Λειτουργίες

Συγκεκριμένα το πρόγραμμα όταν εκτελείται θα λειτουργεί με την παρακάτω δομή:

- Welcome message: Εμφανίζεται μήνυμα καλωσορίσματος του παίκτη με βασικές πληροφορίες για το παιχνίδι, την έκδοσή του, τον δημιουργό του, κ.τ.λ.
- Main menu: Εμφανίζεται menu με τις παρακάτω επιλογές: Help, Start, Exit.
- Η επιλογή Help εμφανίζει πλήρεις οδηγίες και περισσότερες πληροφορίες για το παιχνίδι.
- Με την επιλογή Start ξεκινάει μία παρτίδα του παιχνιδιού, κατά τη διάρκεια της οποίας συμβαίνουν τα παρακάτω:
    - Το παιχνίδι ενημερώνει τον χρήστη για την έναρξη της παρτίδας και ζητάει τα ονόματα των δύο παικτών.
    - Το παιχνίδι αρχικοποιεί με τυχαίες - αλλά έγκυρες και λογικές - τιμές τα hit points, το attack power και το defence rating των δύο παικτών και ενημερώνει τον χρήστη εμφανίζοντας κατάλληλο output στην οθόνη.
    - Το παιχνίδι επιλέγει τυχαία έναν από τους δύο παίκτες.
    - Ο επιλεγμένος παίκτης επιτίθεται - με χρήση της μεθόδου Attack - στον άλλο παίκτη.
    - Το παιχνίδι ελέγχει αν ένας από τους δύο παίκτες έχει σκοτωθεί.
        - Αν ναι, ανακοινώνει ως νικητή τον άλλο παίκτη και επιστρέφει στο βήμα 2.
        - Αν όχι, συνεχίζεται η μάχη.
- Η επιλογή Exit έχει την προφανή λειτουργία.