UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

COS110 Practical 2

Total marks: 60

Due date: 22 August 2016

1 General instructions

- This practical should be completed individually.
- Be ready to upload your practical well before the deadline as no extension will be granted.
- If your code does not compile you will be awarded a mark of 0. The output of your program will be primarily considered for marks, although internal structure may also be tested (eg. the presence of certain functions or classes).
- Read the entire practical specification thoroughly before you start coding.
- To ensure that you did not plagiarize, your code will be inspected with the help of dedicated software.
- Note that plagiarism is considered a very serious offence. Plagiarism will not be tolerated and disciplinary action will be taken against offending students. Please refer to the University of Pretoria's plagiarism page at `http://www.ais.up.ac.za/plagiarism/index.htm`.

2 Overview

In this practical you will practice the concepts of static variables, classes, the copy constructor and overloading the assignment operator.

3 Code Provided for Practical

Download the file `practical2.zip` from the course website under the folder `Practicals/Prac 2`.

In this zip file, the following header file is provided for the practical:

```
#ifndef ADVENTURER_H
#define ADVENTURER_H
#include <string>
using namespace std;

class Adventurer{
private:
    string*** items;
    string name;
    double maxCarryWeight;
    double currentCarryWeight;
    int currentNumberOfItems;
    int maxNumberOfItems;
    double health;
    static int numberOfAdventurers;

public:
    Adventurer(); //default constructor
    Adventurer(const Adventurer& a);  //copy constructor
    ~Adventurer();
    bool pickUpItem(string it, double weight);
    bool dropItem(string it);
    bool dropItem(int index);
    void setName(string n);
    string getName() const;
    void setMaxCarryWeight(double w);
    double getMaxCarryWeight() const;
    void setCurrentCarryWeight(double w);
    double getCurrentCarryWeight() const;
    void setMaxNumberOfItems(int n);
    int getMaxNumberOfItems() const;
    void setCurrentNumberOfItems(int n);
    int getCurrentNumberOfItems() const;
    int getNumberOfAdventurers() const;
    void setHealth(double h);
    double getHealth() const;
    string** getItem(int index) const;
    Adventurer& operator = (const Adventurer& a);
};

#endif  /* ADVENTURER_H */
```

The class `Adventurer` has the following member variables:

- `string items`: A dynamically allocated 2-dimensional array that stores the items that an adventurer has picked up. Each element in the array is a pointer to a string. Its two values for each item are strings, where the first value gives a description of the item and the second value gives the weight of the item.
- `string name`: Contains the name of the adventurer.
- `double maxCarryWeight`: The maximum weight of items that the adventurer is allowed to carry.
- `double currentCarryWeight`: The weight of the items that the adventurer is currently carrying.
- `int maxNumberOfItems`: Maximum number of items that the adventurer is allowed to carry.
- `int currentNumberOfItems`: The number of items that the adventurer is currently carrying.
- `double health`: Health of the adventurer. It indicates a percentage, where 100 is perfect health.
- `static int numberOfAdventurers`: Keeps track of the number of adventurers that are currently in existance.

The class `Adventurer` has the following member functions:

- `Adventurer()`: Default constructor. Assign default values that make sense and initialise what is necessary.
- `Adventurer(const Adventurer& a)`: Copy constructor.
- `~Adventurer()`: Destructor.
- `bool pickUpItem(string it, double weight)`: Determines whether an adventurer can pick up an item and if he can, updates the list of items and all other variables that are affected by adding the new item. To check whether an adventurer can pick up an item, refer to the description of the variables above.
- `bool dropItem(string it)`: Determines whether the adventurer can drop the specific item and if he can, updates the list of items and all other variables that are affected by the removal of the item. `it` provides the description of the item that should be dropped.
- `bool dropItem(int index)`: Similar to the previous function, except the index of the item that should be removed from the list of items are provided instead of the description of the item.
- `void setName(string n)`: Sets the name of the adventurer.

- `string getName() const`: Returns the name of the adventurer.
- `void setMaxCarryWeight(double w)`: Sets the maximum weight of items that the adventurer is allowed to carry.
- `double getMaxCarryWeight() const`: Returns the maximum weight of items that the adventurer is allowed to carry.
- `void setCurrentCarryWeight(double w)`: Sets the total weight of the items that the adventurer is currently carying.
- `double getCurrentCarryWeight() const`: Returns the total weight of the items that the adventurer is currently carying.
- `void setMaxNumberOfItems(int n)`: Sets the maximum number of items that the adventurer is allowed to carry.
- `int getMaxNumberOfItems() const`: Returns the maximum number of items that the adventurer is allowed to carry.
- `void setCurrentNumberOfItems(int n)`: Sets the number of items that the adventurer is currently carrying.
- `int getCurrentNumberOfItems() const`: Returns the number of items that the adventurer is currently carrying.
- `int getNumberOfAdventurers() const`: Returns the number of adventurers that are in existance.
- `void setHealth(double h)`: Sets the health of the adventurer.
- `double getHealth() const`: Returns the health of the adventurer.
- `string getItem(int index) const`: Returns the item that is at the specified index of the list of the item.
- `Adventurer& operator = (const Adventurer& a)`: Assignment operator.

4 Your Task

For this practical your task is the following:

1. Implement all the functions that are declared in the header file of the Adventurer class.
2. Take care that you update the variables in the functions where appropriate. Refer to the description of the class's variables.
3. Test your code properly - test for various cases.
4. Take note: Your are **NOT** allowed to change any of the functions or variables in the header file. You are allowed to add your own helper functions.

However, remember that the provided functions will be called to test your code. You are also **NOT** allowed to change the name of the provided filenames (`Adventurer.h`, `Adventurer.cpp` and `main.cpp`). Remember that Linux is case sensitive, so make sure that the filenames and file extensions conform to the files that are provided for the practical.

5 Marks

An approximate break down of the marks are as follows:

- 6 marks for get and set functions
- 20 marks for `pickUpItem`
- 10 marks for `dropItem`
- 10 marks for the assignment operator
- 10 marks for the copy constructor
- 10 marks for the assignment operator
- 2 marks for the static variable
- 2 marks for the destructor

Take note that when a function is tested, it is not tested in isolation. Variables that are affected by the function are also tested.

6 Submission

Once you are satisfied that your program is working, compress all of your code, including a working `Makefile`, into a single archive (either .tar.gz, .tar.bz2 or .zip) and submit it for marking to the appropriate upload link (Practical 2) before the deadline. Your `Makefile` should compile your code to an executable named `main`. Remember that your `Makefile` must have a run rule to execute your compiled program. That is:

```
run:
    ./main
```

In addition, remember that your Makefile should use the `static` flag to compile and link the code. Furthermore, your zip file should **NOT** contain any folders.