Practice 8
Class/Vector

2018 second semester
Computer Engineering Programming
CONTENTS

Inline Function
Static & Constant
Vector
Midterm Commentary
Inline Function

• Inline Function code actually inserted in place of call.
  – When normal functions are called, a program executes functions jumping to their address (pointer).

• Inline functions are efficient by eliminating overhead, but must be used for very short functions only.

• Definition example:

```cpp
inline getName()
{
    cout<"Get Name is Performed."
    return name;
}
```
Static & Constant

• **Static member variables**
  – All objects of class "share" one copy
  – One object changes it → all see change
    ex) static int a;

• **Static Functions**
  – If no access to object data needed
  – And still "must" be member of the class
  – Can then be called outside class
  – Can only use static data and functions!

• **Constant member variables**
  – Cannot be changed after defined
  – If no need for function modifications
  – Protect parameter with const
    ex) const int b;
Static & Constant Example

```cpp
#include <iostream>
#include <string>

using namespace std;

class Student {
    public:
        Student(string s, string i, string n, int a, int c);
        Student(string s, string i, string n);
        Student();

        void print_studentInfo();
        static void print_studentNum();
        static inline static void inc_stu_num()
        {
            ++student_num;
        }

    private:
        const string school;
        string student_id;
        string name;
        int age;
        static int student_num;
};

int main() {
    Student s1("SNU", "2017312313", "KIM SU MIN", 20, 88);
    Student::inc_stu_num();
    Student::print_studentNum();
    Student s2("SNU", "2015313313", "LEE Gwang Sun");
    Student::inc_stu_num();
    Student::print_studentNum();
    s2.print_studentInfo();
    s2.print_studentNum();
    s2.print_studentInfo();
    s2.print_studentNum();
    return 0;
}

Student::Student(string s, string i, string n, int a, int c)
    : school(s),
      student_id(i),
      name(n),
      age(a),
      score(c)
{
}

void Student::print_studentInfo() {
    cout << school << " 	" << student_id << " 	" << name << " 	" << age << " 	" << score << "<endl;}

void Student::print_studentNum() {
    cout << "Student Number is " << student_num << endl;}
```
Static & Constant Example

Static variable **student_num** is shared via every object. All objects of class “share” one copy.

```
hsherlcok@uni:~/18 fall_cpp$ ./static
Student Number is 1
Student Number is 2
SKKU  2017312313  KIM SU MIN  20  80
Student Number is 2
SKKU  2015313313  LEE GWANG SU  20  0
Student Number is 2
```
Vectors are arrays with dynamic size.

- can grow and shrink during program execution
- Recall that arrays are fixed size!!.

Vectors are used similar to array, but differently declared.

- vector<**Base_Type**> variable_name
  - Any type can be "plugged in" to Base_Type (int, double ...)
  - Example
    - vector<int> v;
#include <iostream>
#include <vector>

using namespace std;

int main() {
    vector<int> v_int;
    int num;
    int i = 0;
    
    while(1) {
        cout << "Enter a list of positive numbers (if you want to stop, enter -1) : ";
        cin >> num;
        if (num != -1) {
            v_int.push_back(num);
            cout << v_int[i] << " is inserted in Vector." << endl;
            cout << "Current Vector Size is " << v_int.size() << " endl; 
            i++;
        }
        else break;
    }
    cout << "Making list is completed..." << endl;
    cout << "Check the list! : " << endl;
    for (i = 0; i < v_int.size(); i++) {
        cout << v_int[i] << " ";
    }
    cout << endl;
    return 0;
}
Exercise

- Make a program that construct Student objects and store in vector. And you should implement vector searching function that search corresponding student from vector.

```cpp
// Get index of corresponding name from vector.
– Get_index_by_name(vector<Student> stdlist, string name)
// Get name of corresponding index from vector.
– Get_name_by_index(vector<Student> stdlist, int idx)
```
#include <iostream>
#include <string>
#include <vector>

using namespace std;

class Student
{
   public:
      Student(string i, string n, int a, int c);
      string GetName();
   
   private:
      string student_id;
      string name;
      int age;
      int score;
};

int Get_index_by_name(vector<Student> stdlist, string name);
string Get_name_by_index(vector<Student> stdlist, int idx);

int main() {
   vector<Student> stds;
   Student s1("2017123543", "LEE GWANG SU", 30, 15);
   Student s2("2014312123", "KIM MIN SU", 24, 89);

   stds.push_back(s1);
   stds.push_back(s2);

   cout << "Student KIM MIN SU's index is " << Get_index_by_name(stds, "KIM MIN SU") << endl;
   cout << "Second student of stds vector is " << Get_name_by_index(stds, 1) << endl;
}
Exercise

hsherlcok@uni:~/18_fall_cpp$ ./exec
Student KIM MIN SU's index is 1
Second student of stds vector is KIM MIN SU