Khoi Duong

Prof. Yang

CS360

8/16/2022

FINAL

1.

Source code:

```
// staken.cpp
// overloading functions in base and derived classes
#include <iostream>
#include <string>
#include <stack>
#include <utility>
using namespace std;
#include <cstdlib> //for exit()
////////////////////////////////////////////////////////////////
class Stack{
  protected: //NOTE: can't be private
    enum { MAX = 6 }; //size of stack array
    int st[MAX]; //stack: array of integers
    int top; //index to top of stack
  public:
    Stack() //constructor
    { top = -1; }
    void push(int var) //put number on stack
      { st[++top] = var; }
    int pop() //take number off stack
    { return st[top--]; }
};
////////////////////////////////////////////////////////////////
class Stack2 : public Stack
{
  public:
    void push(int var) //put number on stack
```

```
    {
      if(top >= MAX-1) //error if stack full
      { cout << "\nError: stack is full"; exit(1); }
      Stack::push(var); //call push() in Stack class
    }
    int pop() //take number off stack
    {
      if(top < 0) //error if stack empty
      { cout << "\nError: stack is empty\n"; exit(1); }
      return Stack::pop(); //call pop() in Stack class
    }
};

class pairStack : public Stack2
{
    public:
    void push(int x, int y) //put number on stack
    {
        Stack2::push(x); //call push() in Stack class
        Stack2::push(y); //call push() in Stack class
    }
    void pop()
    {
        int y = Stack2::pop(); //call pop() in Stack class
        int x = Stack2::pop(); //call pop() in Stack class
        cout << "{" << x << ", " << y << "}";
        cout << endl;
    }
};
////////////////////////////////////////////////////////////////
int main(){
    pairStack s2;
    s2.push(11,22); //push some values onto stack
    s2.push(33,44);
    s2.push(55,66);
    s2.pop(); //pop some values from stack
    s2.pop();
    s2.pop();
    s2.pop(); //oops, popped one too many...
    cout << endl;

  return 0;
}
```

Run program & result:

```
{55, 66}
{33, 44}
{11, 22}

Error: stack is empty
```

2.

Source code:

```
// englcon.cpp
// constructors, adds objects using member function
#include <iostream>
#include <fstream>
using namespace std;
////////////////////////////////////////////////////////////////
class Distance //English Distance class
{
  private:
    int feet;
    float inches;
  public: //constructor (no args)
    Distance() : feet(0), inches(0.0)
    { }
    //constructor (two args)
    Distance(int ft, float in) : feet(ft), inches(in)
    { }
    void getdist() //get length from user
    {
      cout << "\nEnter feet: "; cin >> feet;
      cout << "Enter inches: "; cin >> inches;
      if (inches >= 12.0) //adjust inches
      {
        feet += inches / 12;
        while (inches >= 12.0){
            inches -= 12.0;
        }
      }
    }
```

```
    void showdist() //display distance
    { cout << feet << "\'-" << inches << '\"'; }
    void add_dist( Distance, Distance ); //declaration
};
//--------------------------------------------------------------
//add lengths d2 and d3
void Distance::add_dist(Distance d2, Distance d3)
{
  inches = d2.inches + d3.inches; //add the inches
  feet = 0; //(for possible carry)
  if(inches >= 12.0) //if total exceeds 12.0,
  { //then decrease inches
    inches -= 12.0; //by 12.0 and
    feet++; //increase feet
  } //by 1
  feet += d2.feet + d3.feet; //add the feet
}
////////////////////////////////////////////////////////////////
int main()
{
  char ch;
  Distance dist;
  fstream file;
  //open for append
  file.open("DISTANCE.DAT", ios::app | ios::out |
  ios::in | ios::binary );
  do //data from user to file
  {
    cout << "\nEnter distance's data:";
    dist.getdist(); //get one distance's data
    //write to file
    file.write( reinterpret_cast<char*>(&dist), sizeof(dist) );
    cout << "Enter another distance (y/n)? ";
    cin >> ch;
  }
  while(ch=='y'); //quit on 'n'
    file.seekg(0); //reset to start of file
  //read first distance
  file.read( reinterpret_cast<char*>(&dist), sizeof(dist) );
  while( !file.eof() ) //quit on EOF
  {
    cout << "\nDistance:"; //display distance
    dist.showdist(); //read another distance
    file.read( reinterpret_cast<char*>(&dist), sizeof(dist) );
  }
```

```
  cout << endl;

  Distance dist1, dist3; //define two lengths
  Distance dist2(11, 6.25); //define and initialize dist2
  dist1.getdist(); //get dist1 from user
  dist3.add_dist(dist1, dist2); //dist3 = dist1 + dist2
  //display all lengths
  cout << "\ndist1 = "; dist1.showdist();
  cout << "\ndist2 = "; dist2.showdist();
  cout << "\ndist3 = "; dist3.showdist();
  cout << endl;
  return 0;
}
```

Run program & result:

```
PS D:\VS CODE\C C++\CS360\Final> cd "d:\VS CODE\C C++\CS360\Final\" ;

Enter distance's data:
Enter feet: 12
Enter inches: 15
Enter another distance (y/n)? y

Enter distance's data:
Enter feet: 9
Enter inches: 10
Enter another distance (y/n)? y

Enter distance's data:
Enter feet: 25
Enter inches: 9
Enter another distance (y/n)? n

Distance:13'-3"
Distance:9'-10"
Distance:25'-9"

Enter feet: 7
Enter inches: 4

dist1 = 7'-4"
dist2 = 11'-6.25"
dist3 = 18'-10.25"
PS D:\VS CODE\C C++\CS360\Final>
```

3.

Source code:

```
// arrover3.cpp
// creates safe array (index values are checked before access)
// uses overloaded [] operator for both put and get
#include <iostream>
using namespace std;
#include <cstdlib> //for exit()
const int LIMIT = 100; //array size
////////////////////////////////////////////////////////////////
// template
template <typename T>

class safearay{
  private:
    T arr[LIMIT];
  public:
    T& operator [](int n) //note: return by reference
    {
      if( n< 0 || n>=LIMIT )
      { cout << "\nIndex out of bounds"; exit(1); }
      return arr[n];
    }
};
////////////////////////////////////////////////////////////////
int main(){
  safearay <int> sa1 = safearay <int>() ;
  safearay <int> sa2 = safearay <int>() ;
  for(int j=0; j<LIMIT; j++) //insert elements
    sa1[j] = j*10; //*left* side of equal sign
  for(int j=0; j<LIMIT; j++) //display elements of sa1 int array type
  {
    int temp = sa1[j]; //*right* side of equal sign
    cout << "Element " << j << " is " << temp << endl;
  }
  cout << endl;
  for (int j=0; j<LIMIT; j++) //insert elements
  {
    // create an array of type char
    sa2[j] = 'a' + j;
  }
  for(int j=0; j<LIMIT; j++) //display elements of sa2 int array type
  {
```

```
    char temp = sa2[j]; //*right* side of equal sign
    cout << "Element " << j << " is " << temp << endl;
  }
  return 0;
}
```

Run program & result:

(for sa1 with the datatype of *int*)

Element 0 is 0

Element 1 is 10

Element 2 is 20

Element 3 is 30

Element 4 is 40

Element 5 is 50

Element 6 is 60

Element 7 is 70

Element 8 is 80

Element 9 is 90

Element 10 is 100

Element 11 is 110

Element 12 is 120

Element 13 is 130

Element 14 is 140

Element 15 is 150

Element 16 is 160

Element 17 is 170

Element 18 is 180

Element 19 is 190

Element 20 is 200

Element 21 is 210

Element 22 is 220

Element 23 is 230

Element 24 is 240

Element 25 is 250

Element 26 is 260

Element 27 is 270

Element 28 is 280

Element 29 is 290

Element 30 is 300

Element 31 is 310

Element 32 is 320

Element 33 is 330

Element 34 is 340

Element 35 is 350

Element 36 is 360

Element 37 is 370

Element 38 is 380

Element 39 is 390

Element 40 is 400

Element 41 is 410

Element 42 is 420

Element 43 is 430

Element 44 is 440

Element 45 is 450

Element 46 is 460

Element 47 is 470

Element 48 is 480

Element 49 is 490

Element 50 is 500

Element 51 is 510

Element 52 is 520

Element 53 is 530

Element 54 is 540

Element 55 is 550

Element 56 is 560

Element 57 is 570

Element 58 is 580

Element 59 is 590

Element 60 is 600

Element 61 is 610

Element 62 is 620

Element 63 is 630

Element 64 is 640

Element 65 is 650

Element 66 is 660

Element 67 is 670

Element 68 is 680

Element 69 is 690

Element 70 is 700

Element 71 is 710

Element 72 is 720

Element 73 is 730

Element 74 is 740

Element 75 is 750

Element 76 is 760

Element 77 is 770

Element 78 is 780

Element 79 is 790

Element 80 is 800

Element 81 is 810

Element 82 is 820

Element 83 is 830

Element 84 is 840

Element 85 is 850

Element 86 is 860

Element 87 is 870

Element 88 is 880

Element 89 is 890

Element 90 is 900

Element 91 is 910

Element 92 is 920

Element 93 is 930

Element 94 is 940

Element 95 is 950

Element 96 is 960

Element 97 is 970

Element 98 is 980

Element 99 is 990

(Picture on the next page)

```
Element 63 is 630
Element 64 is 640
Element 65 is 650
Element 66 is 660
Element 67 is 670
Element 68 is 680
Element 69 is 690
Element 70 is 700
Element 71 is 710
Element 72 is 720
Element 73 is 730
Element 74 is 740
Element 75 is 750
Element 76 is 760
Element 77 is 770
Element 78 is 780
Element 79 is 790
Element 80 is 800
Element 81 is 810
Element 82 is 820
Element 83 is 830
Element 84 is 840
Element 85 is 850
Element 86 is 860
Element 87 is 870
Element 88 is 880
Element 89 is 890
Element 90 is 900
Element 91 is 910
Element 92 is 920
Element 93 is 930
Element 94 is 940
Element 95 is 950
Element 96 is 960
Element 97 is 970
Element 98 is 980
Element 99 is 990
```

(for sa2 with the datatype of *char*)

Element 0 is a

Element 1 is b

Element 2 is c

Element 3 is d

Element 4 is e

Element 5 is f

Element 6 is g

Element 7 is h

Element 8 is i

Element 9 is j

Element 10 is k

Element 11 is l

Element 12 is m

Element 13 is n

Element 14 is o

Element 15 is p

Element 16 is q

Element 17 is r

Element 18 is s

Element 19 is t

Element 20 is u

Element 21 is v

Element 22 is w

Element 23 is x

Element 24 is y

Element 25 is z

Element 26 is {

Element 27 is |

Element 28 is }

Element 29 is ~

Element 30 is

Element 31 is Ç

Element 32 is ü

Element 33 is é

Element 34 is â

Element 35 is ä

Element 36 is à

Element 37 is å

Element 38 is ç

Element 39 is ê

Element 40 is ë

Element 41 is è

Element 42 is ï

Element 43 is î

Element 44 is ì

Element 45 is Ä

Element 46 is Å

Element 47 is É

Element 48 is æ

Element 49 is Æ

Element 50 is ô

Element 51 is ö

Element 52 is ò

Element 53 is û

Element 54 is ù

Element 55 is ÿ

Element 56 is Ö

Element 57 is Ü

Element 58 is ¢

Element 59 is £

Element 60 is ¥

Element 61 is Pts

Element 62 is *f*

Element 63 is á

Element 64 is í

Element 65 is ó

Element 66 is ú

Element 67 is ñ

Element 68 is Ñ

Element 69 is ª

Element 70 is º

Element 71 is ¿

Element 72 is ⌐

Element 73 is ¬

Element 74 is ½

Element 75 is ¼

Element 76 is ¡

Element 77 is «

Element 78 is »

Element 79 is ░

Element 80 is ▒

Element 81 is ▓

Element 82 is │

Element 83 is ┤

Element 84 is ╡

Element 85 is ╢

Element 86 is ╖

Element 87 is ╕

Element 88 is ╣

Element 89 is ║

Element 90 is ╗

Element 91 is ╝

Element 92 is ╩

Element 93 is ╛

Element 94 is ┐

Element 95 is └

Element 96 is ┴

Element 97 is ┬

Element 98 is ├

Element 99 is ─

(Picture on the next page)

```
Element 55 is ÿ
Element 56 is Ö
Element 57 is Ü
Element 58 is ¢
Element 59 is £
Element 60 is ¥
Element 61 is ₧
Element 62 is ƒ
Element 63 is á
Element 64 is í
Element 65 is ó
Element 66 is ú
Element 67 is ñ
Element 68 is Ñ
Element 69 is ª
Element 70 is º
Element 71 is ¿
Element 72 is ⌐
Element 73 is ¬
Element 74 is ½
Element 75 is ¼
Element 76 is ¡
Element 77 is «
Element 78 is »
Element 79 is ░
Element 80 is ▒
Element 81 is ▓
Element 82 is │
Element 83 is ┤
Element 84 is ╡
Element 85 is ╢
Element 86 is ╖
Element 87 is ╕
Element 88 is ╣
Element 89 is ║
Element 90 is ╗
Element 91 is ╝
Element 92 is ╜
Element 93 is ╛
Element 94 is ┐
Element 95 is └
Element 96 is ┴
Element 97 is ┬
Element 98 is ├
Element 99 is ─
PS D:\VS CODE\C C++\CS360\Final>
```