云南大学数学与统计学院

上机实践报告

|  |  |  |
| --- | --- | --- |
| **课程名称**：数据结构与算法实验 | **年级**：2015级 | **上机实践成绩**： |
| **指导教师**：陆正福 | **姓名**：刘鹏 |  |
| **上机实践名称**：树结构实验 | **学号**：20151910042 | **上机实践日期**：2017-05-03 |
| **上机实践编号**：No.08 | **组号**： | **上机实践时间**：上午三、四节 |

# 一、实验目的

1. 熟悉与树和二叉树有关的数据结构与算法；

2. 熟悉主讲教材Chapter 8的代码片段。

# 二、实验内容

# 三、实验平台

Windows 10 Enterprise 中文版；

Python 3.6.0；

Wing IDE Professional 6.0.2-1集成开发环境。

# 四、实验记录与实验结果分析

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82 | # 2.3.1 Example: CreditCard Class  class CreditCard:  """A consumer credit card."""  def \_\_init\_\_(self,customer,bank,acnt,limit):  """Create a new credit card instance.    The initial balance is zero.    customer the name of the customer (e.g., 'John Bowman')  bank the name of the bank (e.g., 'California Savings')  acnt the acount identifier (e.g., '5391 0375 9387 5309')  limit credit limit (measured in dollars)  """  self.\_customer = customer  self.\_bank = bank  self.\_account = acnt  self.\_limit = limit  self.\_balance = 0  def get\_customer(self):  """Return name of the customer."""  return self.\_customer  def get\_bank(self):  """Return the bank's name."""  return self.\_bank  def get\_account(self):  """Return the card identifying number (typically stored as a string)."""  return self.\_account  def get\_limit(self):  """Return current credit limit."""  return self.\_limit  def get\_balance(self):  """Return current balance."""  return self.\_balance  def charge(self,price):  """Charge given price to the card, assuming sufficient credit limit.    Return True if charge was processed; False if charge was denied  """  if price + self.\_balance > self.\_limit: # if charge would exceed limit  return False # cannot accept charge  else:  self.\_balance += price  return True  def make\_payment(self,amount):  """Process customer payment that reduces balance."""  self.\_balance -= amount  #------------------------------ main function ------------------------------  if \_\_name\_\_ == '\_\_main\_\_':  wallet = []  wallet.append(CreditCard('John Bowman','California Savings',\  '5391 0375 9387 5309',2500))  wallet.append(CreditCard('John Bowman','California Fedoral',\  '3485 0399 3395 1954',3500))  wallet.append(CreditCard('John Bowman','California Finance',\  '5391 0375 9387 5309',5000))  for val in range(1,17):  wallet[0].charge(val)  wallet[1].charge(2\*val)  wallet[2].charge(3\*val)    for c in range(3):  print('Customer =',wallet[c].get\_customer())  print('Bank =',wallet[c].get\_bank())  print('Account =',wallet[c].get\_account())  print('Limit =',wallet[c].get\_limit())  print('Balance =',wallet[c].get\_balance())  while wallet[c].get\_balance() > 100:  wallet[c].make\_payment(100)  print('New balance =',wallet[c].get\_balance())  print() |

coordinates

# 五、实验体会

Translation：

Chapter 8 Trees

＊第八章 树

8.1 General Trees

＊8.1节 一般的树

Productivity experts say that breakthroughs come by thinking “nonlinearly.” In this chapter, we discuss one of the most important nonlinear data structures in computing – ***trees***. Tree structures are indeed a breakthrough in data organization, for they allow us to implement a host of algorithms much faster than when using linear data structures, such as array-based lists or linked lists. Trees also provide a natural organization for data, and consequently have become ubiquitous structures in file systems, graphical user interfaces, databases, Web sites, and other computer systems.

＊

END

# 六、参考文献

[1] Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser, *Data Structures and Algorithms in Python,* Chapter 4

[2] 实验教材：汪萍，陆正福等编著 数据结构与算法的问题与实验 第1章