

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКО”

Факультет прикладної математики

Кафедра системного програмування і спеціальних комп’ютерних систем

**Лабораторна робота №1**

З дисципліни «Організація баз даних»

«Ознайомлення з базовими конструкціями мови Python. Спрощена база даних»

**Виконав:**

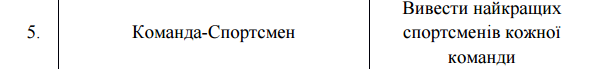
**студент III-го курсу**

**групи КВ-41**

**Горпинич-Радуженко Іван**

**Київ 2016**

**Варіант:**

****

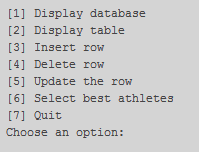


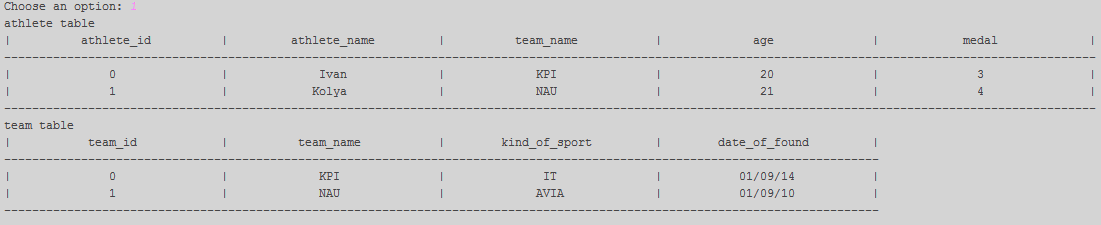
**Текст програми:**

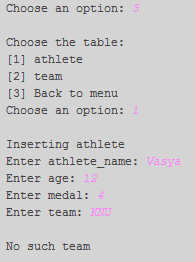
**class Main:**  
  
 def \_\_init\_\_ (self, fileProduct, fileOrder):  
 self.fileProduct = fileProduct  
 self.fileOrder = fileOrder  
  
 def main(self):  
 ui = GUI.GUI()  
 main = Engine.Engine(self.fileProduct, self.fileOrder)  
 choice = ui.menu()  
 while choice != 7:  
 if choice == 1:  
 self.show\_database(ui, main)  
 elif choice == 2:  
 self.show\_table(ui, main)  
 elif choice == 3:  
 self.insert(ui, main)  
 elif choice == 4:  
 self.delete(ui, main)  
 elif choice == 5:  
 self.update(ui, main)  
 elif choice == 6:  
 self.select(ui, main)  
 choice = ui.menu()  
 main.pack()  
 sys.exit(0)  
  
 def show\_database(self, ui, main):  
 ui.show\_table\_athlete('athlete', main.get\_athlete())  
 ui.show\_table\_team('team', main.get\_team())  
  
 def show\_table(self, ui, main):  
 table = self.what\_table(ui)  
 if table == 3:  
 return  
 if table == 1:  
 ui.show\_table\_athlete('athlete', main.get\_athlete())  
 else:  
 ui.show\_table\_team('team', main.get\_team())  
  
 def insert(self, ui, main):  
 table = self.what\_table(ui)  
 if table == 3:  
 return  
 if table == 1:  
 self.insert\_into\_athlete(ui, main)  
 else:  
 self.insert\_into\_team(ui, main)  
  
 def delete(self, ui, main):  
 table = self.what\_table(ui)  
 if table == 3:  
 return  
 if table == 1:  
 self.delete\_from\_athlete(ui, main)  
 else:  
 self.delete\_from\_team(ui, main)  
  
 def update(self, ui, main):  
 table = self.what\_table(ui)  
 if table == 3:  
 return  
 if table == 1:  
 self.update\_athlete(ui, main)  
 else:  
 self.update\_team(ui, main)  
  
 def select(self, ui, main):  
 main.select\_variant()  
  
 def insert\_into\_athlete(self, ui, main):  
 info = ui.insert\_athlete\_info()  
 if not info:  
 return  
 if not (info[0] and info[1]and info[2]and info[3]):  
 ui.error('Invalid input')  
 return  
 is\_error = main.insert\_into\_athlete(info[0], info[1], info[2], info[3])  
 ui.is\_successful(is\_error)  
  
 def insert\_into\_team(self, ui, main):  
 info = ui.insert\_team\_info()  
 if not info:  
 return  
 if not (info[0] and info[1] and info[2]):  
 ui.error('Invalid input')  
 return  
 is\_error = main. insert\_into\_team(info[0], info[1], info[2].strftime("%d/%m/%y"))  
 ui.is\_successful(is\_error)  
  
 def delete\_from\_athlete(self, ui, main):  
 info = ui.delete\_athlete\_info()  
 if info:  
 is\_error = main.delete\_from\_athlete(info[0])  
 ui.is\_successful(is\_error)  
  
 def delete\_from\_team(self, ui, main):  
 info = ui.delete\_team\_info()  
 if info:  
 is\_error = main.delete\_from\_team(info[0], info[1])  
 ui.is\_successful(is\_error)  
  
 def update\_athlete(self, ui, main):  
 old\_info = ui.update\_\_info\_athlete()  
 if not old\_info:  
 ui.error('Invalid input')  
 return  
 existing\_athlete = main.athlete\_to\_update(old\_info[0])  
 if type(existing\_athlete) == str:  
 ui.error(existing\_athlete)  
 return  
 new\_info = ui.update\_new\_info\_athlete()  
 if new\_info:  
 is\_error = main.update\_athlete(old\_info[0], new\_info[0], new\_info[1], new\_info[2], new\_info[3])  
 ui.is\_successful(is\_error)  
  
 def update\_team(self, ui, main):  
 old\_info = ui.update\_\_info\_team()  
 if not (old\_info and old\_info[0] and old\_info[1]):  
 ui.error('Invalid input')  
 return  
 existing\_team = main.team\_to\_change(old\_info[0], old\_info[1])  
 if type(existing\_team) == str:  
 ui.error(existing\_team)  
 return  
 new\_info = ui.update\_new\_info\_team()  
 if new\_info:  
 is\_error = main.update\_team(old\_info[0], old\_info[1], new\_info[0], new\_info[1], new\_info[2])  
 ui.is\_successful(is\_error)  
  
 def what\_table(self, ui):  
 table = ui.what\_table()  
 while not table:  
 table = ui.what\_table()  
 return table  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 c = Main('athlete.txt', 'team.txt')  
 c.main()

import pickle  
  
**class Engine:**  
  
 def \_\_init\_\_(self, fileteam, fileAthlete):  
 try:  
 self.fileteam = fileteam  
 self.fileAthlete = fileAthlete  
 DB = open(fileteam, 'rb')  
 self.athlete = pickle.load(DB)  
 DB.close()  
 DB = open(fileAthlete, 'rb')  
 self.team = pickle.load(DB)  
 DB.close()  
 except:  
 self.athlete = list()  
 self.team = list()  
  
 def get\_athlete(self):  
 return self.athlete  
  
 def get\_team(self):  
 return self.team  
  
 def insert\_into\_athlete(self, athlete\_name, age, medal, team):  
 if not self.team\_name\_in\_table(team):  
 return 'No such team'  
 if not self.athlete:  
 athlete\_id = 0  
 else:  
 athlete\_id = len(self.athlete)  
 if self.athlete\_name\_in\_table(athlete\_name):  
 return 'Such athlete already exists'  
 self.athlete.append({'athlete\_id': athlete\_id, 'athlete\_name': athlete\_name, 'age': age, 'medal': medal, 'team\_name': team})  
  
 def insert\_into\_team(self, team,kind\_of\_sport, found\_date):  
  
 if self.team\_name\_in\_table(team):  
 return 'Such team already exists'  
 if not self.team:  
 team\_id = 0  
 else:  
 team\_id = len(self.team)  
 self.team.append({'team\_id': team\_id, 'team\_name': team, 'kind\_of\_sport': kind\_of\_sport, 'date\_of\_found': found\_date})  
  
 def delete\_from\_athlete (self, athlete\_id):  
 existing\_athlete = self.athlete\_id\_in\_table(athlete\_id)  
 if not existing\_athlete:  
 return 'No such athlete'  
 self.athlete.remove(existing\_athlete[0])  
  
 def delete\_from\_team (self, team, kind\_of\_sport):  
 if filter(lambda x: x['team\_name'] == team, self.athlete):  
 return 'Cannot delete an team'  
 team = self.team\_to\_change(team, kind\_of\_sport)  
 '''  
 if type(team) == str:  
 return team  
 '''  
 self.team.remove(team)  
  
 def athlete\_to\_update(self, athlete\_id):  
 existing\_athlete = self.athlete\_id\_in\_table(athlete\_id)  
 if not existing\_athlete:  
 return 'No such athlete'  
  
 def update\_athlete(self, athlete\_id, new\_athlete\_name, new\_age, new\_medal, new\_team):  
  
 existing\_athlete = self.athlete\_id\_in\_table(athlete\_id)  
 if new\_athlete\_name:  
 existing\_athlete[0]['athlete\_name'] = new\_athlete\_name  
 if new\_age:  
 existing\_athlete[0]['age'] = new\_age  
 if new\_medal:  
 existing\_athlete[0]['medal'] = new\_medal  
 if new\_team:  
 existing\_athlete[0]['team\_name'] = new\_team  
  
 def team\_to\_change(self, team, kind\_of\_sport):  
 existing\_team = self.team\_name\_in\_table(team)  
 if not existing\_team:  
 return 'No such team'  
 if filter(lambda x: x['team\_name'] == team, self.athlete):  
 return 'Cannot update an team'  
 return existing\_team[0]  
  
 def update\_team(self, team, kind\_of\_sport, new\_team, new\_kind, new\_found):  
 if new\_team and new\_kind and self.team\_in\_table(new\_team, new\_kind):  
 return 'Such team already exists'  
 existing\_team = self.team\_in\_table(team, kind\_of\_sport)  
 if new\_team:  
 existing\_team[0]['team\_name'] = new\_team  
 if new\_kind:  
 existing\_team[0]['kind\_of\_sport'] = new\_kind  
 if new\_found:  
 existing\_team[0]['date\_of\_found'] = new\_found.strftime("%d/%m/%y")  
  
 def select\_variant(self):  
 for team in self.team:  
 print "team: ", team['team\_name']  
 team\_players = filter(lambda x: x['team\_name'] == team['team\_name'], self.athlete)  
 max\_medals, best\_athlete = 0, None  
 for athlete in team\_players:  
 if athlete['medal'] > max\_medals:  
 max\_medals = athlete['medal']  
 best\_athlete = athlete  
 if best\_athlete is not None:  
 print "Best athlete: ", best\_athlete['athlete\_name'], best\_athlete['medal']  
 print '-'\*30  
   
 def pack(self):  
 DB = open(self.fileteam, 'wb')  
 pickle.dump(self.athlete, DB)  
 DB.close()  
 DB = open(self.fileAthlete, 'wb')  
 pickle.dump(self.team, DB)  
 DB.close()  
  
 def team\_name\_in\_table(self, team\_name):  
 return filter(lambda x: x['team\_name'] == team\_name, self.team)  
  
 def athlete\_name\_in\_table(self, athlete\_name):  
 return filter(lambda x: x['athlete\_name'] == athlete\_name, self.athlete)  
  
 def athlete\_id\_in\_table(self, athlete\_id):  
 return filter(lambda x: x['athlete\_id'] == athlete\_id, self.athlete)  
  
 def team\_in\_table(self,team, kind\_of\_sport):  
 return filter(lambda x: x['team\_name'] == team and x['kind\_of\_sport'] == kind\_of\_sport, self.team)  
  
**class GUI:**  
 def menu(self):  
 print '\n[1] Display database'  
 print '[2] Display table'  
 print '[3] Insert row'  
 print '[4] Delete row'  
 print '[5] Update the row'  
 print '[6] Select best athletes'  
 print '[7] Quit'  
 try:  
 selection = int(raw\_input('Choose an option: '))  
 if not 1 <= selection <= 7:  
 raise ValueError  
 return selection  
 except ValueError:  
 self.error('Invalid input')  
 return None  
  
 def show\_table\_team(self, table\_name, table):  
 print '{:^10}'.format(table\_name + ' table')  
 if not table:  
 print '{:^10}'.format('empty')  
 else:  
 columns = table[0].keys()  
 print '|{:^30}|{:^30}|{:^30}|{:^30}|'.format(columns[1] ,columns[3], columns[0], columns[2])  
 print '-' \* 125  
 for row in table:  
 print '|{:^30}|{:^30}|{:^30}|{:^30}|'.format(row[columns[1]], row[columns[3]], row[columns[0]], row[columns[2]])  
 print '-' \* 125  
 def show\_table\_athlete(self, table\_name, table):  
 print '{:^10}'.format(table\_name + ' table')  
 if not table:  
 print '{:^10}'.format('empty')  
 else:  
 columns = table[0].keys()  
 print '|{:^30}|{:^30}|{:^30}|{:^30}|{:^30}|'.format(columns[2], columns[4], columns[3], columns[0],columns[1])  
 print '-' \* 156  
 for row in table:  
 print '|{:^30}|{:^30}|{:^30}|{:^30}|{:^30}|'.format(row[columns[2]], row[columns[4]], row[columns[3]], row[columns[0]], row[columns[1]])  
 print '-' \* 156  
  
 def delete\_athlete\_info(self):  
 row = list()  
 print '\nDeleting athlete'  
 athlete\_id = int(raw\_input("Enter athlete\_id: "))  
 row.append(athlete\_id)  
 return row  
  
 def delete\_team\_info(self):  
 row = list()  
 print '\nDeleting team'  
 try:  
 row.append(raw\_input("Enter team\_name: "))  
 row.append(raw\_input("Enter kind\_of\_sport: "))  
 return row  
 except ValueError:  
 self.error('Invalid input')  
 return None  
  
 def insert\_athlete\_info(self):  
 row = list()  
 print '\nInserting athlete'  
 try:  
 row.append(raw\_input("Enter athlete\_name: "))  
 row.append(raw\_input('Enter age: '))  
 row.append(raw\_input('Enter medal: '))  
 row.append(raw\_input('Enter team: '))  
 return row  
 except ValueError:  
 self.error('Invalid input')  
 return None  
  
 def insert\_team\_info(self):  
 row = list()  
 print '\nInserting team'  
 try:  
 row.append(raw\_input("Enter team\_name: "))  
 row.append(raw\_input("Enter kind of sport: "))  
 date\_str = raw\_input("Enter found date (dd/mm/yy): ")  
 if not date\_str:  
 raise ValueError  
 row.append(datetime.datetime.strptime(date\_str, "%d/%m/%y").date())  
 return row  
 except ValueError:  
 self.error('Invalid input')  
 return None  
  
 def update\_\_info\_athlete(self):  
 row = list()  
 print '\nUpdating athlete'  
 row.append(int(raw\_input("Enter athlete\_id to update: ")))  
 return row  
  
 def update\_\_info\_team(self):  
 row = list()  
 print '\nUpdating team'  
 try:  
 row.append(raw\_input("Enter team\_name to update: "))  
 row.append(raw\_input("Enter kind of sport to update: "))  
 return row  
 except ValueError:  
 self.error('Invalid input')  
 return None  
  
 def update\_new\_info\_athlete(self):  
 row = list()  
 try:  
 row.append(raw\_input("Enter new athlete\_name (press Enter if you don't want to update this attribute): "))  
 row.append(int(raw\_input("Enter new age (press '0' if you don't want to update this attribute): ")))  
 row.append(int(raw\_input("Enter new medal (press '0' if you don't want to update this attribute): ")))  
 row.append(raw\_input("Enter new team (press Enter if you don't want to update this attribute): "))  
 return row  
 except ValueError:  
 self.error('Invalid input')  
 return None  
  
 def update\_new\_info\_team(self):  
 row = list()  
 try:  
 row.append(raw\_input("Enter new team\_name (press Enter if you don't want to update this attribute): "))  
 row.append(raw\_input("Enter new kind\_of\_sport (press Enter if you don't want to update this attribute): "))  
 date\_str = raw\_input(  
 "Enter new found date (dd/mm/yy) (press Enter if you don't want to update this attribute): ")  
 if not date\_str:  
 row.append(date\_str)  
 else:  
 row.append(datetime.datetime.strptime(date\_str, "%d/%m/%y").date())  
 return row  
 except ValueError:  
 self.error('Invalid input')  
 return None  
  
 def is\_successful(self, error\_message):  
 if not error\_message:  
 print '\nSuccess'  
 else:  
 self.error(error\_message)  
  
 def what\_table(self):  
 print '\nChoose the table: '  
 print '[1] athlete'  
 print '[2] team'  
 print '[3] Back to menu'  
 try:  
 selection = int(raw\_input('Choose an option: '))  
 if not 1 <= selection <= 3:  
 raise ValueError  
 return selection  
 except ValueError:  
 self.error('Invalid input')  
 return None  
  
 def error(self, message):  
 print '\n'+message

**Скріншоти:**

****

****

****