Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Навчально-науковий комплекс  
«Інститут прикладного системного аналізу»

Відділення другої вищої та післядипломної освіти

**ЛАБОРАТОРНА РОБОТА №3**

(варіант 19)

з курсу «Програмування»

на тему «Динамічні структури данних»

Виконав: студент 3-го курсу

групи ІС-зп71

Бутузов О.В.

Прийняв: викл. Древаль М.М.

Захищено з оцінкою

« » 2017 р.

Київ – 2017

# Умова завдання

Створити однозв’язний список працівників, який містить персональні дані працівника та інформацію про заробітну плату за останні місяці (кількість місяців вводиться з клавіатури). Персональні дані включають таку інформацію: прізвище, посада, дата народження (з окремими полями для дня, місяця та року). Інформація про заробітну плату — масив записів: назва місяця, нарахована сума.

1) Реалізувати функції перегляду записаних даних, додавання нового елемента на задану позицію (INSERT), видалення елемента з заданої позиції (DELETE). Початковий вміст списку заповнити з клавіатури.

2) Побудувати двійкове дерево працівників, яким за визначений місяць було нараховано суму, більшу за вказану (назва місяця та сума вводяться з клавіатури). Вивести його вміст у порядку зростання віку працівників. Оформити як окрему функцію.

Примітка: за погодження викладача введення інформації було реалізовано на читання з файлу.

# Розвязання задачі

Задача була реалізована наступним чином:

1. Було реалізовано 2 контейнера для структур данних що ми мали викорастати за умовою задачі – linked list та binary tree. Контейнери реалізовані через примітивні класи.
2. В обох стуктурах було передбачено використання довільних данних (через реалізацію template’ів). З збереженням у данних у спец структурах.
3. Тестові данні знаходяться в файлі employees.csv і можуть бути введені в программу насутпним чином ‘./salary-report employees.csv’.
4. Було імплементовано (мінімальне) інтерактивне середовище для генерації рапортів (відповідно поставленій задачі).
5. Окрім данимічних структур які ми імплементували в ході виконання також використовувались: строки (std::string), вектора (std::vector), мапи (std::map) та туплі (std::tuple).

# Лістинг програм

// clang++ employees.cpp -o test -std=c++14

#include <iostream> /\* cout , err \*/

#include <stdio.h> /\* printf, scanf, NULL \*/

#include <stdlib.h> /\* malloc, delete, atoi \*/

#include <sstream> /\* string stream \*/

#include <string> /\* std::string \*/

#include <vector> /\* std::vector \*/

#include "list.cpp" /\* linkedlist implementation \*/

#include "app.cpp" /\* app specific code \*/

#include "binary-tree.cpp" /\* binary tree implementation \*/

// спец дебагер

void debug(std::string message){ std::cout << message <<std::endl; }

// кастом вюшка для лінкованого списку.

template<>

std::string LinkedList<Employee>::view(Employee item){

return item.Name;

}

// custom view for workage structure

template <>

std::string BinaryTree<WorkerAge>::view(WorkerAge item){

return

+ "("

+ std::to\_string( std::get<1>(item) )

+ "-"

+ std::to\_string( std::get<2>(item) )

+ "-"

+ std::to\_string( std::get<3>(item) )

+ ") "

+ std::get<0>(item)

+ " з зарплатнею "

+ std::to\_string( std::get<4>(item) );

}

int main(int argc, char\* argv[]){

// Declaring Linked List for Employees

LinkedList<Employee> employees;

// in case if you don't want to have input data...

bool debug(false);

// Load Data from csv to speed up development.

// other wise plese ./employees employees.csv

int error; // return state.

if ( argc == 2 ) {

error = load\_csv( argv[1], &employees );

} else {

// debug settings

std::string FName = "/Users/butuzov/Dropbox/CS122-AIS/Programming\_C&C++/PSet\_03/employees.csv";

error = load\_csv( FName.c\_str(), &employees );

}

if ( error != 0 ) {

return error;

}

// Looks like this in linked list structure.

std::cout << std::endl

<< "Зв'язний список для нашої структури виглядає наступним чином"

<< std::endl;

employees.visualize();

std::cout << std::endl

<< "Створення рапортів зарплатні за місяць"

<< std::endl;

std::map<std::string, int> monthes;

monthes.insert ( std::pair<std::string, int>("січень", 1) );

monthes.insert ( std::pair<std::string, int>("лютий", 2) );

monthes.insert ( std::pair<std::string, int>("березень", 3) );

monthes.insert ( std::pair<std::string, int>("квітень", 4) );

monthes.insert ( std::pair<std::string, int>("травень", 5) );

monthes.insert ( std::pair<std::string, int>("червень", 6) );

monthes.insert ( std::pair<std::string, int>("липень", 7) );

monthes.insert ( std::pair<std::string, int>("серпень", 8) );

monthes.insert ( std::pair<std::string, int>("вересень", 9) );

monthes.insert ( std::pair<std::string, int>("жовтень", 10) );

monthes.insert ( std::pair<std::string, int>("листопад", 11) );

monthes.insert ( std::pair<std::string, int>("грудень", 12) );

while( true ) {

int month(-1);

int n = 0;

while( ! ( 0 < month || month > 11 ) ) {

std::cout << "Оберіть місяць для створення рапорту по зарплатні"

<< std::endl

<< "Доступні опції:"

<< std::endl;

for ( const auto& i : monthes) {

std::cout << "- "

<< i.first

<< std::endl;

}

std::cout << std::endl

<< "Будь-ласка введіть обраний місяць: > ";

std::string in("вересень");

if (debug == false){

std::cin >> in;

}

std::cout << std::endl

<< "Будь-ласка введіть обраний рівень зарплати: > ";

int salary\_level(3000);

if (debug == false){

std::cin >> salary\_level;

}

if ( salary\_level <= 0 ) {

// do nothing

std::cerr << std::endl

<< "Ви щось переплутали, зарплатня замала."

<< " "

<< "Cпробуйте ще раз"

<< std::endl;

continue;

}

try {

month = monthes.at(in);

BinaryTree<WorkerAge> salaryData;

std::cout << "Формуємо звіт за " << in ;

employees.front();

do {

Employee worker;

worker = employees.current();

//std::cout << worker.Name << std::endl;

if ( salary\_level < worker.salary[ (month - 1) ] ) {

salaryData.insert(

std::make\_tuple(

worker.Name,

worker.Year,

worker.Month,

worker.Day,

worker.salary[ (month - 1) ]

)

);

}

} while ( employees.next() );

std::cout << " і показуємо зарплатний звіт:"

<< std::endl

<< std::endl ;

salaryData.showtree();

std::cout << std::endl

<< std::endl ;

} catch( const std::out\_of\_range& oor ){

// do nothing

std::cerr << "Ви щось переплутали, місяць не доступний."

<< " "

<< "Cпробуйте ще раз"

<< std::endl;

continue;

}

if (++n > 10){

std::cerr << "примусове припинення программи"

<< std::endl;

return 0;

}

}

std::cout << "Бажаєте продовжити генерувати рапорти? (т/так) ";

std::string answer("ні");

if (debug == false){

std::cin >> answer;

}

//

if ( answer != std::string("т") ) {

std::cout

<< std::endl

<< "До побачення!"

<< std::endl

<< "Дякуємо за використання нашого програмного забезпечення!"

<< std::endl;

break;

}

std::cout << std::endl;

}

}

# Результати роботи програми

Результати цієї лабораторної роботи у вигляді коду та тестових даних доступні за адресою

https://github.com/butuzov/CS122-AIS/tree/master/Programming\_C&C++/PSet\_03

1. **Висновки**

Я навчився як ефективно використовувати пам'ять при роботі з динамічними структурами данних.