**РУСЕНСКИ УНИВЕРСИТЕТ „АНГЕЛ КЪНЧЕВ“**

Катедра: „Компютърни системи и технологии“

**КУРСОВА ЗАДАЧА**

**по дисциплината**

**„Синтез и анализ на алгоритми“**

**Изготвил:** Диан Лъчезаров Лазаров

**Фак. №:** 233078

**Група:** 25Б

**Специалност:** КСТ

**Курс:** 2

**Дата:** 14.12.2024 г. **Приел:** ................................... **/**доц. д-р Е. Големанова**/**

**Задание №7**

Авиокомпания съхранява справочната си информация в списък, всеки елемент на който съдържа следните данни: номер на полет, брой свободни места, брой заети места. Да се напишат функции за:

* създаване на списъка;
* запазване на място по зададен номер на полет;
* извеждане на информацията за всички полети със свободни места.

1. **Графично изображение на динамичния списък с конкретни примерни стойности**

Pwaasa

Last

next

NULL

C409

498

2

213

186 (occupiedSeats)

14 (freeSeats)

SU6086 (flightNumber)

37

QF103

1. **Описание на алгоритмите, съответстващи на отделните подточки от заданието**

* **Създаване на списъка**

Създаването на едносвързан списък от полети става чрез процедурата **CreateFlight**(**point &Head**), която има „главата“ на списъка, зададена като параметър. Използват се указатели към първия и последния елемент (**point P** и **point Last**). Създаването и добавянето на елементи в списъка е реализирано посредством **while** цикъл, в който потребителят трябва да въведе информация за полетите (номер на полет, брой свободни места и брой заети места). Ако списъкът е празен (**Head == NULL**), новият елемент става първи („глава“ на списъка). В противен случай, новият елемент се добавя в края на списъка.

* **Запазване на място по зададен номер на полет**

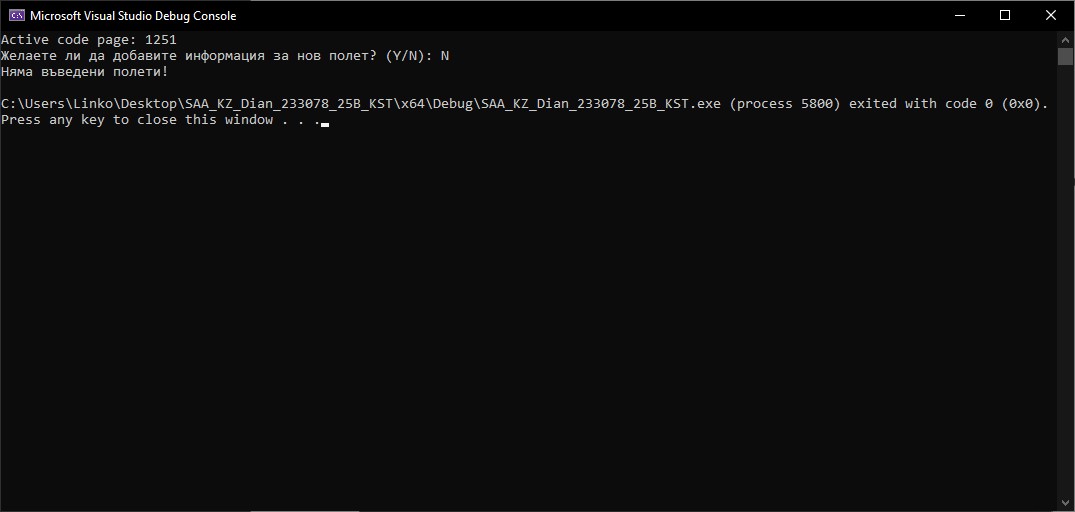
За целта е разработена процедурата **ReserveSeat(point P, string flightNumber)** с параметри указател към първия елемент и номер на полет. Използвана е булева променлива (**bool FlightFoundCheck**), чрез която се проверява дали въведеният от потребителя полет съществува в списъка. Ако това не е така, се извежда уместно съобщение и програмата приключва. Докато списъкът се намира в непразно състояние **(P != NULL)**, процедурата проверява дали текущият номер съвпада с въведения и дали броят на свободните места е по-голям от нула (т.е. дали има свободни места). Ако тези две условия са налице, броят на свободните места се намалява с единица (**P->freeSeats--**), а броят на заетите места се увеличава (**P->occupiedSeats++**). След това се извежда съобщение, указващо, че мястото за съответния полет е запазено успешно. Ако условията не са изпълнени, отново е предвидено да се изведе подходящо съобщение. След приключване на обработката указателят на текущия елемент се пренасочва към следващия (**P = P->next**).

* **Извеждане на информацията за всички полети със свободни места**

За извеждане на информацията за всички полети със свободни места е създадена процедурата **PrintFreeSeats(point P)** с параметър указател към първия елемент. Подобно на предходната процедура, тук отново се използва булева променлива (**bool FreeSeatsCheck**) за проверка за полети със свободни места в списъка. Докато списъкът не е празен **(P != NULL)**, се проверява дали броят на свободните места е по-голям от нула и ако това е така, се извежда информацията за всички полети със свободни места (номер на полет, брой свободни места, брой заети места). След приключване на обработката указателят на текущия елемент се пренасочва към следващия (**P = P->next**). Ако **FreeSeatsCheck == false** (т.е. ако няма полети със свободни места), се извежда подходящо съобщение и програмата завършва.

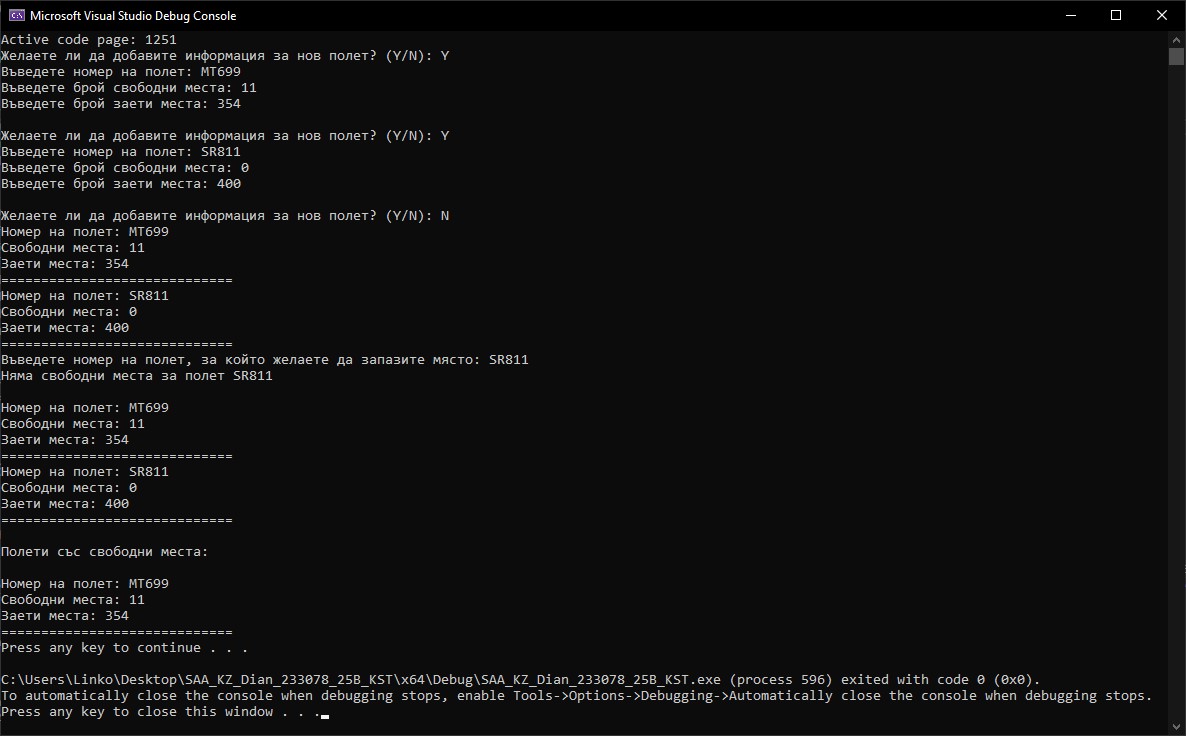
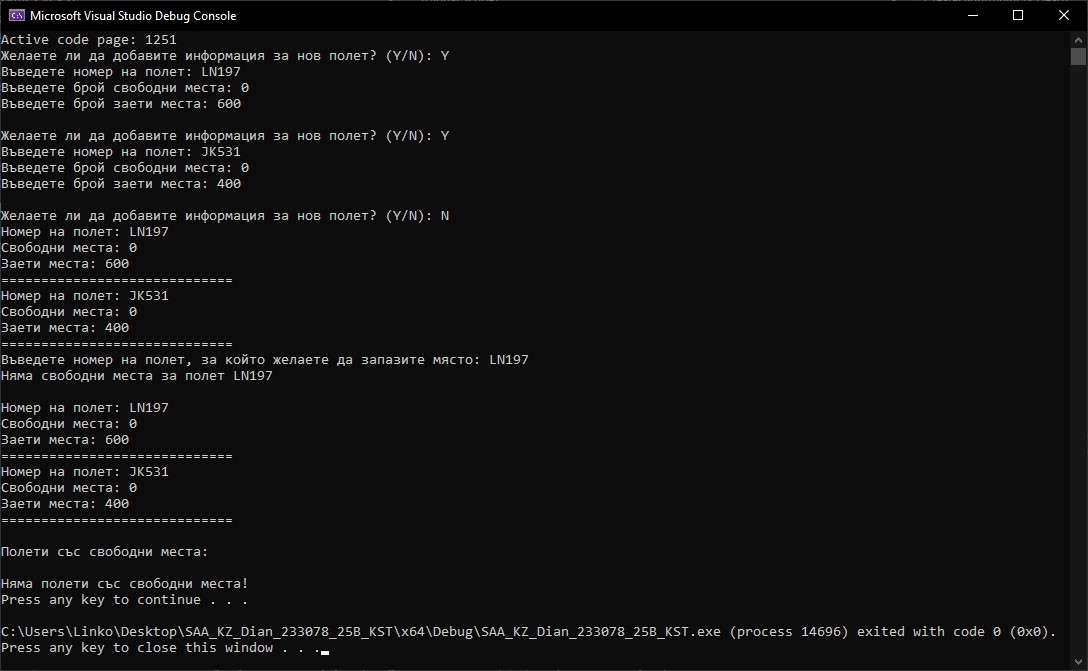
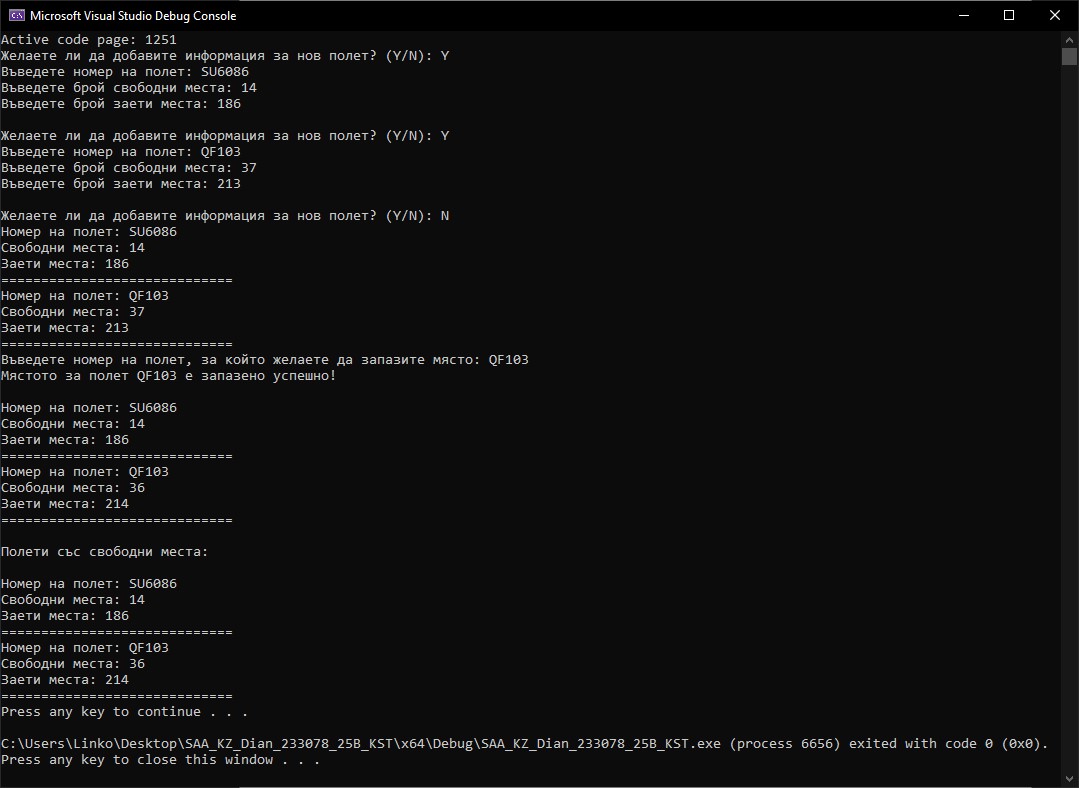
1. **Описание на използваните функции/процедури**

* **Заглавна част на функцията:** void **CreateFlight(point &Head)**
* **Действие:** процедура за създаване на едносвързан списък от полети.
* **Параметри:** указател към „главата“ на списъка (Head).
* **Използвани глобални променливи:** няма.
* **Извиквани функции:** няма.
* **Заглавна част на функцията**: **void PrintFlight(point P)**
* **Действие:** процедура за извеждане на съдържанието на списъка.
* **Параметри:** указател P към текущия елемент.
* **Използвани глобални променливи:** няма.
* **Извиквани функции:** няма.
* **Заглавна част на функцията**: **void ReserveSeat(point P, string flightNumber)**
* **Действие:** процедура за запазване на място по зададен номер на полет.
* **Параметри:** указател P към текущия елемент; символен низ flightNumber.
* **Използвани глобални променливи:** няма.
* **Извиквани функции:** няма.
* **Заглавна част на функцията**: **void PrintFreeSeats(point P)**
* **Действие:** процедура за извеждане на информация за всички полети със свободни места.
* **Параметри:** указател P към текущия елемент.
* **Използвани глобални променливи:** няма.
* **Извиквани функции:** няма.
* **Заглавна част на функцията**: **void main()**
* **Действие:** главна функция за стартиране на програмата и извикване на създадените функции/процедури.
* **Параметри:** няма.
* **Използвани глобални променливи:** няма.
* **Извиквани функции:**
* **CreateFlight(Head)**
* **PrintFlight(Head)**
* **ReserveSeat(Head, flightReserve)**
* **PrintFreeSeats(Head)**

1. **Тестови примери (снимки на екрана), доказващи коректността на програмата**

**Тест №2**

**Тест №1**

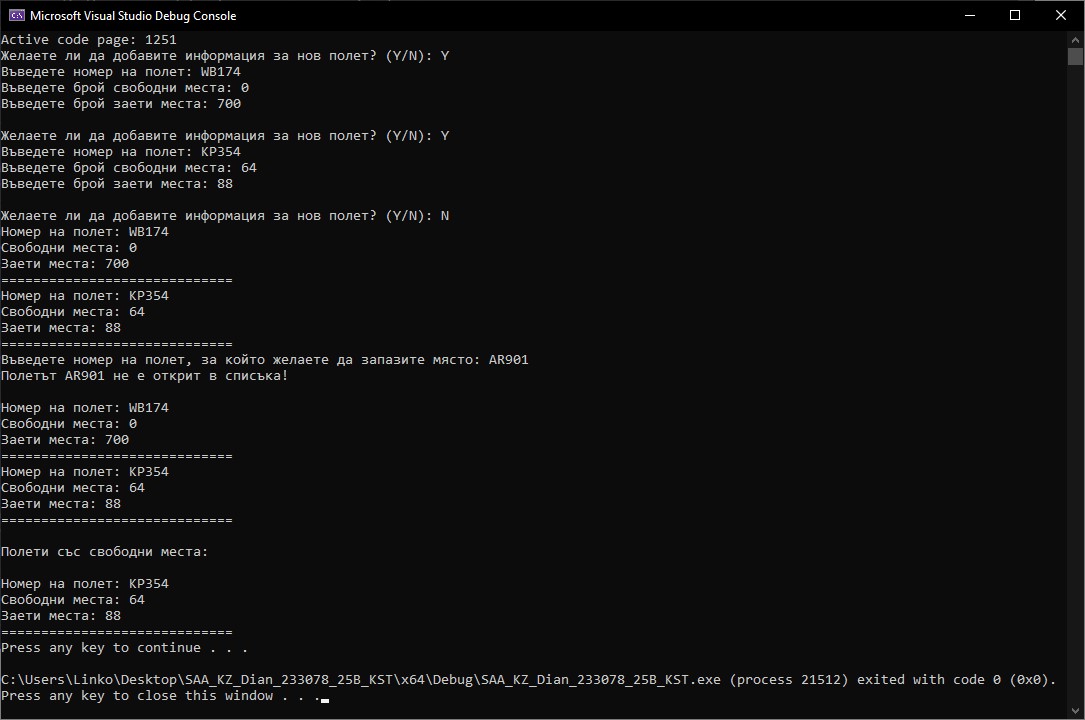
****

**Тест №2**

**Тест №4**

**Тест №3**

**Тест №4**

****

**Тест №5**

1. **Разпечатка на кода**

#include <iostream>

using namespace std;

// описание на едносвързания списък (структурата)

struct Flight {

string flightNumber;

int freeSeats;

int occupiedSeats;

Flight\* next;

};

typedef Flight\* point;

point Head;

// опашка (FIFO) – процедура за създаване на списъка

void CreateFlight(point& Head) {

point P; // текущ елемент

point Last = NULL; // последен елемент

char input;

cout << "Желаете ли да добавите информация за нов полет? (Y/N): "; cin >> input;

while (input == 'Y' || input == 'y') {

// създаване и добавяне на нов елемент с указател P в списъка

P = new Flight;

cout << "Въведете номер на полет: "; cin >> P->flightNumber;

cout << "Въведете брой свободни места: "; cin >> P->freeSeats;

cout << "Въведете брой заети места: "; cin >> P->occupiedSeats;

cout << endl;

P->next = NULL; // край на списъка

if (Head == NULL) {

Head = P; // ако списъкът е празен, новият елемент става първи („глава“ на списъка)

}

else {

Last->next = P; // ако списъкът не е празен, новият елемент се свързва към досегашния последен

}

Last = P; // новият елемент става последен

cout << "Желаете ли да добавите информация за нов полет? (Y/N): "; cin >> input;

}

}

// процедура за извеждане на информация за въведените полети

void PrintFlight(point P) {

while (P != NULL) { // докато списъкът не е празен

cout << "Номер на полет: " << P->flightNumber << endl;

cout << "Свободни места: " << P->freeSeats << endl;

cout << "Заети места: " << P->occupiedSeats << endl;

cout << "=============================" << endl;

P = P->next; // пренасочване на указателя на текущия елемент към следващия

}

}

// процедура за запазване на място по зададен номер на полет

void ReserveSeat(point P, string flightNumber) {

bool FlightFoundCheck = false; // булева променлива, която проверява дали въведеният полет е в списъка

while (P != NULL) {

if (P->flightNumber == flightNumber) {

FlightFoundCheck = true; // true, ако въведеният полет съвпада с текущия

if (P->freeSeats > 0) { // проверка за полети със свободни места

P->freeSeats--;

P->occupiedSeats++;

cout << "Мястото за полет " << flightNumber << " е запазено успешно!" << endl << endl;

}

else {

cout << "Няма свободни места за полет " << flightNumber << endl << endl;

}

}

P = P->next;

}

if (FlightFoundCheck == false) { // false, ако въведеният полет не е открит в списъка

cout << "Полетът " << flightNumber << " не е открит в списъка!" << endl << endl;

}

}

// процедура за извеждане на информация за всички полети със свободни места

void PrintFreeSeats(point P) {

bool FreeSeatsCheck = false; // булева променлива, която проверява дали има полети със свободни места

while (P != NULL) {

if (P->freeSeats > 0) {

FreeSeatsCheck = true; // true, ако броят на свободните места е > 0

cout << "Номер на полет: " << P->flightNumber << endl;

cout << "Свободни места: " << P->freeSeats << endl;

cout << "Заети места: " << P->occupiedSeats << endl;

cout << "=============================" << endl;

}

P = P->next;

}

if (FreeSeatsCheck == false) { // false, ако няма полети със свободни места

cout << "Няма полети със свободни места!" << endl;

}

}

void main() {

system("chcp 1251");

// празен списък

Head = NULL;

// извикване на процедурата за създаване на списъка

CreateFlight(Head);

// прекратяване на програмата, ако списъкът е празен

if (Head == NULL) {

cout << "Няма въведени полети!" << endl;

return;

}

// извикване на процедурата за извеждане на информация за полетите

PrintFlight(Head);

// извикване на процедурата за запазване на място по зададен номер на полет и повторно отпечатване на обновения списък

string flightReserve;

cout << "Въведете номер на полет, за който желаете да запазите място: "; cin >> flightReserve;

ReserveSeat(Head, flightReserve);

PrintFlight(Head);

// извикване на процедурата за извеждане на информация за всички полети със свободни места

cout << "\nПолети със свободни места:\n" << endl;

PrintFreeSeats(Head);

system("pause");

}