МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное

образовательное учреждение высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт компьютерных технологий и информационной безопасности

Кафедра информационно-аналитических систем безопасности

имени Л.С. Берштейна

**ОТЧЕТ**

**По индивидуальному заданию**

**по дисциплине**

**Основы алгоритмизации и программирования**

**Вариант - 18**

Выполнил:

студент гр. КТбо1-8

Кравцов В.Д.

Проверил:

Профессор кафедры ИАСБ

Беляков С.Л.

«\_\_\_» 2017 г.

Таганрог – 2017

Оглавление

[Цели работы: 2](#_Toc497161631)

[Выполнение работы: 3](#_Toc497161632)

[Программный код. 4](#_Toc497161633)

[Демонстрация работы программы 5](#_Toc497161634)

[Заключение: 6](#_Toc497161635)

# Цели работы

**Вариант 18. Задание :** Получить общую длительность телефонных разговоров указанного вида для клиентов, заключивших договоры в текущем месяце.

**Входные данные :** Коды двух услуг по разговорам.

# Программный код

#include "stdafx.h"

#include <stdio.h>

#include <string>

#include <time.h>

struct Aminities\_st

{

char name[1000];

char code[1000];

char price[1000];

char points[1000];

};

struct Used\_aminities\_st

{

char t\_number[11];

char code[1000];

char date[1000];

char time[1000];

char time\_of\_using[1000];

int converted\_time\_of\_using;

int count = 0;

};

struct Clients\_st

{

char name[1000];

char t\_number[11];

char date\_s[11];

char date\_e[11];

char bill[1000];

char credit[1000];

int count;

};

struct Settings

{

char code\_1[1000];

char code\_2[1000];

};

struct Date

{

char day[3];

char month[3];

char year[5];

};

int fileNotExist(FILE\*, FILE\*, FILE\*, FILE\*);

void fillData(Clients\_st\*, Aminities\_st\*, Used\_aminities\_st\*, FILE\*, FILE\*, FILE\*);

void fillClients(Clients\_st\*, FILE\*);

void fillAminities(Aminities\_st\*, FILE\*);

void fillUsedAminities(Used\_aminities\_st\*, FILE\*);

void convertTimeOfUsing(Used\_aminities\_st\*);

int CharToInt32(char\*);

int Compare(char\*, char\*);

int compareDate(int, int, char\*);

void Report( char\*, char\*,Clients\_st\*, Aminities\_st\*, Used\_aminities\_st\*, FILE\*);

int main()

{

setlocale(LC\_ALL, "Russian");

FILE \*clients = NULL;

FILE \*aminities = NULL;

FILE \*used\_aminities = NULL;

FILE \*report = NULL;

clients = fopen("Clients.txt", "r");

aminities = fopen("Aminities.txt", "r");

used\_aminities = fopen("Used\_aminities.txt", "r");

report = fopen("Report.txt", "w");

if (fileNotExist(clients, aminities, used\_aminities, report))

{

return 0;

}

struct Clients\_st people[10];

struct Aminities\_st amin[50];

struct Used\_aminities\_st used\_amin[50];

struct Settings settings = {"", ""};

struct Date dt;

fillData(people, amin, used\_amin, clients, aminities, used\_aminities); // Filling structures with data.

fclose(clients);

fclose(aminities);

fclose(used\_aminities);

convertTimeOfUsing(used\_amin);//Converting data from string to int32

printf("Type first rate code: ");

scanf("%s", settings.code\_1);

printf("\nType second rate code: ");

scanf("%s", settings.code\_2);

Report(settings.code\_1, settings.code\_2, people, amin, used\_amin, report);

fclose(report);

return 0;

}

int fileNotExist(FILE \*clients, FILE \*aminities, FILE \*used\_aminities, FILE \*report)

{

if (clients == NULL || aminities == NULL || used\_aminities == NULL || report == NULL)

{

printf("One or more files does not exist! Check your data!\n\n");

return true;

}

{

printf("All files were succesfully found!\n\n");

return false;

}

}

void fillClients(Clients\_st \*people, FILE \*clients)

{

char chr[1000];

int k = 0, s = 0, sch = 0;

while (!feof(clients)) {

fgets(chr, 1000, clients);

if (!feof(clients))

{

chr[strlen(chr) - 1] = ',';

}

//Getting name

while (chr[k] != ',')

{

people[sch].name[k] = chr[k];

k++;

}

people[sch].name[k] = '\0';

k++;

if (chr[k] == ' ')

{

k++;

}

//Getting telephone number

while (chr[k] != ',')

{

people[sch].t\_number[s] = chr[k];

k++;

s++;

}

people[sch].t\_number[s] = '\0';

k++;

s = 0;

//Getting date of starting deal

while (chr[k] != ',')

{

people[sch].date\_s[s] = chr[k];

s++;

k++;

}

people[sch].date\_s[s] = '\0';

k++;

s = 0;

if (chr[k] == ' ')

{

k++;

}

//Getting date of ending deal

while (chr[k] != ',')

{

people[sch].date\_e[s] = chr[k];

s++;

k++;

}

people[sch].date\_e[s] = '\0';

k++;

s = 0;

if (chr[k] == ' ')

{

k++;

}

//Getting bill

while (chr[k] != ',')

{

people[sch].bill[s] = chr[k];

s++;

k++;

}

people[sch].bill[s] = '\0';

k++;

s = 0;

if (chr[k] == ' ')

{

k++;

}

//Getting credit

while (chr[k] != ',' && k != strlen(chr))

{

people[sch].credit[s] = chr[k];

s++;

k++;

}

people[sch].credit[s] = '\0';

k++;

s = 0;

//printf("%s %s %s %s %s %s\n", people[sch].name, people[sch].t\_number, people[sch].date\_s, people[sch].date\_e, people[sch].bill, people[sch].credit);

sch++;

k = 0;

}

people[1].count = sch;

}

void fillAminities(Aminities\_st \*amin, FILE \*aminities)

{

char rd[1000], chr[1001];

int k = 0, s = 0, sch = 0;

while (!feof(aminities)) {

fgets(rd, 1000, aminities);

strcpy(chr, rd);

if (!feof(aminities))

{

chr[strlen(rd) - 1] = ',';

}

//Getting name

while (chr[k] != ',')

{

amin[sch].name[k] = chr[k];

k++;

}

amin[sch].name[k] = '\0';

k++;

if (chr[k] == ' ')

{

k++;

}

//Getting code

while (chr[k] != ',')

{

amin[sch].code[s] = chr[k];

k++;

s++;

}

amin[sch].code[s] = '\0';

k++;

if (chr[k] == ' ')

{

k++;

}

s = 0;

//Getting price

while (chr[k] != ',')

{

amin[sch].price[s] = chr[k];

k++;

s++;

}

amin[sch].price[s] = '\0';

k++;

if (chr[k] == ' ')

{

k++;

}

s = 0;

//Getting points

while (chr[k] != ',')

{

amin[sch].points[s] = chr[k];

k++;

s++;

}

amin[sch].points[s] = '\0';

s = 0;

//printf("%s %s %s %s\n", amin[sch].name, amin[sch].code, amin[sch].price, amin[sch].points);

sch++;

k = 0;

}

}

void fillUsedAminities(Used\_aminities\_st \*used\_amin, FILE \*used\_aminities)

{

char rd[1000], chr[1001];

int k = 0, s = 0, sch = 0;

while (!feof(used\_aminities)) {

fgets(rd, 1000, used\_aminities);

strcpy(chr, rd);

if (!feof(used\_aminities))

{

chr[strlen(rd) - 1] = ',';

}

//Getting telephone number

while (chr[k] != ',')

{

used\_amin[sch].t\_number[k] = chr[k];

k++;

}

used\_amin[sch].t\_number[k] = '\0';

k++;

if (chr[k] == ' ')

{

k++;

}

//Getting code

while (chr[k] != ',')

{

used\_amin[sch].code[s] = chr[k];

k++;

s++;

}

used\_amin[sch].code[s] = '\0';

k++;

s = 0;

if (chr[k] == ' ')

{

k++;

}

//Getting date

while (chr[k] != ',')

{

used\_amin[sch].date[s] = chr[k];

k++;

s++;

}

used\_amin[sch].date[s] = '\0';

s = 0;

k++;

if (chr[k] == ' ')

{

k++;

}

//Getting time

while (chr[k] != ',')

{

used\_amin[sch].time[s] = chr[k];

k++;

s++;

}

used\_amin[sch].time[s] = '\0';

s = 0;

k++;

if (chr[k] == ' ')

{

k++;

}

//Getting time of using

while (chr[k] != ',')

{

used\_amin[sch].time\_of\_using[s] = chr[k];

s++;

k++;

}

used\_amin[sch].time\_of\_using[s] = '\0';

s = 0;

k++;

if (chr[k] == ' ')

{

k++;

}

//printf("{%s} {%s} {%s} {%s} {%s} {%d}\n", used\_amin[sch].t\_number, used\_amin[sch].code, used\_amin[sch].date, used\_amin[sch].time, used\_amin[sch].time\_of\_using, sch);

k = 0;

sch++;

used\_amin[1].count++;

}

}

void fillData(Clients\_st \*people, Aminities\_st \*amin, Used\_aminities\_st \*used\_amin, FILE \*clients, FILE \*aminities, FILE \*used\_aminities)

{

fillClients(people, clients);

fillAminities(amin, aminities);

fillUsedAminities(used\_amin, used\_aminities);

printf("Data was loaded succesfully\n\n");

}

void convertTimeOfUsing(Used\_aminities\_st \*used\_amin)

{

for (int i = 0; i < used\_amin[1].count; i++)

{

used\_amin[i].converted\_time\_of\_using = CharToInt32(used\_amin[i].time\_of\_using);

}

}

int CharToInt32(char \*x)

{

if (x != "#")

{

int len = strlen(x), razr = 1, res = 0;

for (int i = 0; i < len - 1; i++)

{

razr \*= 10;

}

for (int i = 0; i < len; i++)

{

switch (x[i])

{

case '1': {res += razr; razr /= 10; break; }

case '2': {res += 2 \* razr; razr /= 10; break; }

case '3': {res += 3 \* razr; razr /= 10; break; }

case '4': {res += 4 \* razr; razr /= 10; break; }

case '5': {res += 5 \* razr; razr /= 10; break; }

case '6': {res += 6 \* razr; razr /= 10; break; }

case '7': {res += 7 \* razr; razr /= 10; break; }

case '8': {res += 8 \* razr; razr /= 10; break; }

case '9': {res += 9 \* razr; razr /= 10; break; }

case '0': {razr /= 10; break; }

}

}

return res;

}

else

return 1;

}

int Compare(char \*a, char \*b)

{

if (strlen(a) == strlen(b))

{

for (int i = 0; i < strlen(a); i++)

{

if (a[i] != b[i])

{

return false;

}

}

return true;

}

else

return false;

}

int compareDate(int m, int y, char \*b)

{

char month[3] = { b[3], b[4], '\0' }, year[5] = { b[6], b[7], b[8], b[9], '\0' };

if (CharToInt32(month) == m && CharToInt32(year) == y)

{

return true;

}

else

return false;

}

void Report(char \*ami1, char \*ami2, Clients\_st \*clients, Aminities\_st \*aminities, Used\_aminities\_st \*used\_aminities, FILE \*report)

{

time\_t t;

struct tm \*nabor;

time(&t);

nabor = localtime(&t);

int year;

int month;

year = nabor->tm\_year += 1900;

month = nabor->tm\_mon + 1;

int summ = 0, summ\_each = 0;

for (int i = 0; i < clients[1].count; i++)

{

for (int j = 0; j < used\_aminities[1].count; j++)

{

if (Compare(used\_aminities[j].t\_number, clients[i].t\_number) && (Compare(used\_aminities[j].code, ami1) ||Compare(used\_aminities[j].code, ami2)) && compareDate(month, year, used\_aminities[j].date))

{

summ\_each += used\_aminities[j].converted\_time\_of\_using;

}

}

fprintf(report, "User %s was using this function by %d seconds\n", clients[i].name, summ\_each);

summ += summ\_each;

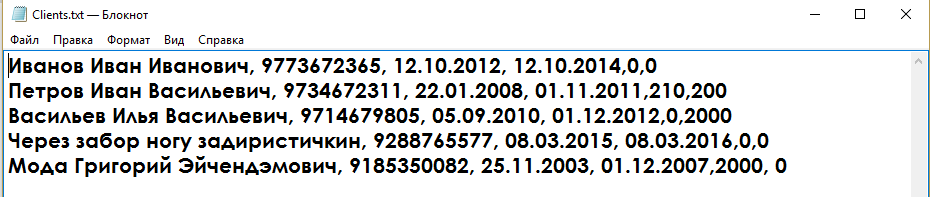
summ\_each = 0;

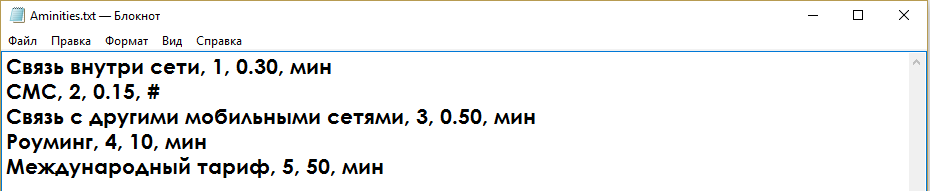
}

fprintf(report, "All users was using this function by %d seconds\n", summ);

}

# Демонстрация работы программы





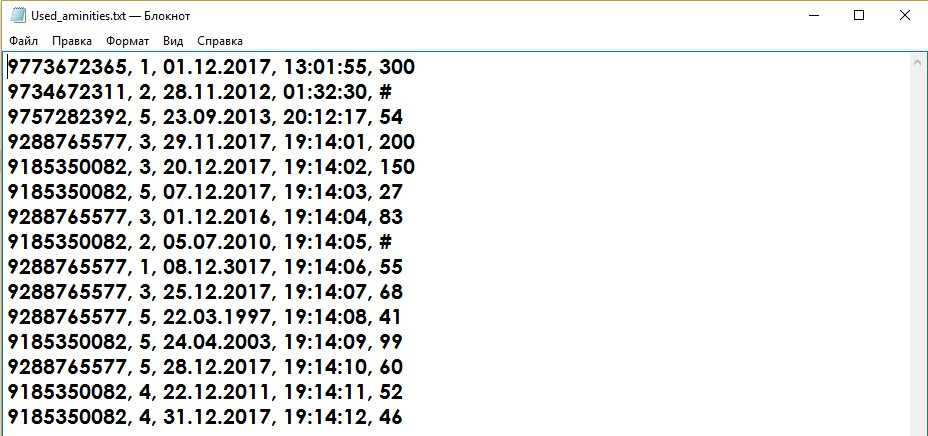


Рис. 1, 2, 3. Пример входных данных.

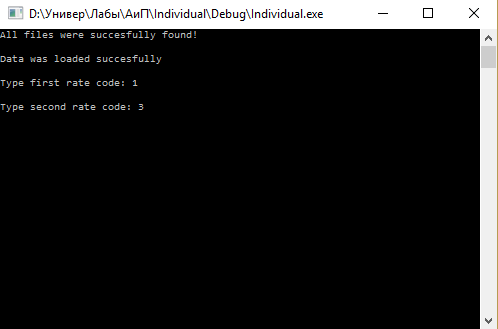


Рис. 4. Пример ввода данных с клавиатуры.

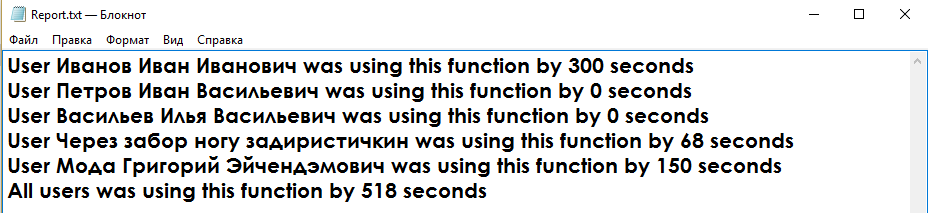


Рис. 5. Пример выходных данных.

# Заключение

В индивидуальном задании были использованы: сложные типы данных, работа с текущей датой и временем и операции над ними. Благодаря этой работе были укреплены навыки программирования.