Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное  
учреждение высшего образования  
ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт Информационных технологий и анализа данных  
  
Отделение прикладной математики и информатики

ИСПОЛЬЗОВАНИЕ ФУНКЦИЙ.

ОТЧЕТ  
по лабораторной работе № 4  
Вариант №10

по дисциплине «программирование»

Выполнил

Студент группы ЭВМб-23-3 \_\_\_\_\_\_\_\_\_\_\_\_Г. М. Распутин  
  
Принял

Доцент \_\_\_\_\_\_\_\_\_\_\_\_\_ А. Б. Столбов

Иркутск 2023

# Использование функций. Задание А.

## Условие задачи.

*10в:* Описать функцию CircleSR вещественного типа, находящую площадь круга радиуса R (R — вещественное). С помощью этой функции найти площади трех кругов с данными радиусами. Площадь круга радиуса R вычисляется по формуле S = р·R2.

1. Таблица внешних спецификаций всей программы

Таблица 1.1 – Внешние спецификации всей программы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Имя | Назначение | Тип | Вх/Вых. | Диапазон |
| 1 | R1 | Радиус первого круга | Вещественное | Вход | От 0 до 100000 |
| 2 | R2 | Радиус второго круга | Вещественное | Вход | От 0 до 100000 |
| 3 | R3 | Радиус третьего круга | Вещественное | Вход | От 0 до 100000 |
| 4 | С1 | Площадь первого круга | Вещественное | Выход | От 0.001 до 3.14159e+10 |
| 5 | С2 | Площадь второго круга | Вещественное | Выход | От 0.001 до 3.14159e+10 |
| 6 | С3 | Площадь третьего круга | Вещественное | Выход | От 0.001 до 3.14159e+10 |

## **3. Схема иерархии функций**

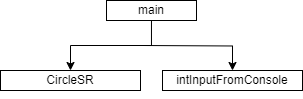


Рисунок 1 - Схема иерархии функций.

## **Таблица спецификаций функций**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Имя функции** | **Назначение** | **Тип результата** | **Параметры** |
| 1 | CircleSR | Нахождение площади круга по заданному радиусу | *­Вещественный* | *pi, вещественное, по значению, число пи*  *R, вещественное, по значению, радиус круга* |
| 2 | intInputFromConsole | Проверка на правильность ввода данных. При вводе невещественного значения – сообщение об ошибки и повторный ввод данных | *Вещественный* | *mess, строка, по значению, при необходимости выводимое сообщение*  *err, строка, по значению, при необходимости выводимая ошибка*  *min, вещественное, по значению, минимальное число, которое может ввести пользователь*  *max, вещественное, по значению, максимальное число, которое может ввести пользователь* |

1. Таблица внешних спецификаций для каждой функции

Таблица 1.2 – Внешние спецификации функции CircleSR

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Имя | Назначение | Тип | Вх/Вых. | Диапазон |
| 1 | R | Радиус круга | Вещественное | Вход | От 0 до 100000 |
| 2 | *pi* | Число Пи | Вещественное | Вход | 3.14 |
| 3 | *S* | Площадь круга | Вещественное | Выход | От 0.001 до 3.14159e+10 |

Таблица 1.3 – Внешние спецификации функции intInputFromConsole

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Имя | Назначение | Тип | Вх/Вых. | Диапазон |
| 1 | *mess* | Сообщение, которое выводится при каждом вызове функции | Текстовое | Вход | Символы ASCII кодировки |
| 2 | *err* | Сообщение, выводящееся при ошибке | Текстовое | Вход | Символы ASCII кодировки |
| 3 | *min* | Минимальное число, которое может ввести пользователь | Вещественное | Вход | От 0 до 100000 |
| 4 | *max* | Максимальное число, которое может ввести пользователь | Вещественное | Вход | От 0 до 100000 |
| 5 | *curNumber* | Итоговое число | Вещественное | Выход | От 0 до 100000 |

1. Алгоритм главной программы:
2. Вызов функции intInputFromConsole для ввода R1
3. Если R1 > max или R1 <= min, то п.1
4. Вызов функции intInputFromConsole для ввода R2
5. Если R2 > max или R2 <= min, то п.3
6. Вызов функции intInputFromConsole для ввода R3
7. Если R2 > max или R2 <= min, то п.5
8. Вызов функции CircleSR(pi, R1)
9. Вызов функции CircleSR(pi, R2)
10. Вызов функции CircleSR(pi, R3)
11. Вывод S1, S2, S3, где S – площадь заданного круга
12. Алгоритмы для каждой функций:

**Функция** intInputFromConsole

Алгоритм:

1. curNumber = 0.

2. создать переменную line.

3. Вывести сообщение mess на консоль.

4. while, пока getline(cin, line).

5. переедать в stringstream ss line.

6. Проверить, можно ли извлечь число из ss с помощью оператора >> и сохранить его в curNumber.

7. Если удалось извлечь число, проверить, закончился ли поток ss с помощью функции eof().

8. Если поток ss закончился, проверить, находится ли curNumber в диапазоне от min до max.

9. Если curNumber находится в диапазоне от min до max, вернуть его.

10. Если число не удовлетворяет условиям, вывести сообщение об ошибке err на консоль.

11. Вернуть 0, если не удалось получить корректное число из ввода пользователя.

**Функция** CircleSR

Алгоритм:

1. S = pi \* R \* R
2. Вернуть S
3. Таблица тестов

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Назначение теста | Входные данные | Выходные данные |
| 1 | Простой тест с целыми значениями | R1 = 1 R2 = 2  R3 = 3 | S1 = 3.14159  S2 = 12.5664  S3 = 28.2743 |
| 2 | Простой тест с вещественными значениями | R1 = 5.6  R2 = 7.8  R3 = 9.1 | S1 = 78.5398  S2 = 153.938  S3 = 254.469 |

1. Программа на языке C++

<https://replit.com/@GieorghiiRasput/LABPROG4AAA#main.cpp>

//примечание, в этом задании я разбил код на несколько файлов, поэтому будет удобнее, если он будет загружен на replit

## 

# Использование функций. Задание Б.

## Условие задачи.

*10в:* Написать программу, которая проверяет, является ли заданная последовательность литер, заканчивающаяся пробелом, изображением целого числа без знака, делящегося на 3.

1. Таблица внешних спецификаций

Таблица 2.1 – Внешние спецификаций

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Имя | Назначение | Тип | Вх/Вых. | Диапазон |
| 1 | line | Последовательность литер, которая должна заканчиваться пробелом | Вещественное | Вход | От 0 до 100000 |
| 2 | С1 | Сообщение о соответствии литеральной последовательности условию и о делимости на три | Текстовое | Выход | Три вида |

С1:

1. Введенная последовательность не соответствует условиям
2. Заданная последовательность литер делится на 3 и заканчивается на пробел
3. заданная последовательность литер не делится на 3, но заканчивается на пробел

## **Схема иерархии функций**

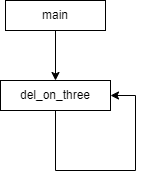


Рисунок 2 - Схема иерархии функций.

## **Таблица спецификаций функций**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Имя функции** | **Назначение** | **Тип результата** | **Параметры** |
| 1 | del\_on\_three | Проверка на соответствие условию задачи и вычисление суммы чисел заданной последовательности литералов | *Целый* | *line, строковые, по значению, последовательность литералов, вводимая пользователем*  *num\_of\_numbers\_i\_dont\_know, целое, по значению, номер элемента строки*  *sum, целое, по значению, сумма чисел литералов, которая считается рекурсивно*  *err, текстовые, по значению, вывод сообщения об ошибке первого типа,*  *err2, текстовые, по значению, вывод сообщения об ошибке второго типа* |

Таблица 2.3 – Внешние спецификации функции del\_on\_three

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Имя | Назначение | Тип | Вх/Вых. | Диапазон |
| 1 | *line* | *последовательность литералов, вводимая пользователем* | Текстовые | Вход | Символы ASCII кодировки |
| 2 | *sum* | *сумма чисел литералов, которая считается рекурсивно* | *Целые* | Вход/выход | От -1 до 999999 |
| 3 | *num\_of\_numbers \_i\_dont\_know* | *Номер элемента строки, который увеличивается с каждым вызовом функции* del\_on\_three | *Целые* | Вход | От 0 до 10 |
| 4 | *err* | *вывод сообщения об ошибке первого типа* | *Текстовые* | Вход/выход | "Вы ввели не число. " |
| 5 | *err2* | *вывод сообщения об ошибке второго типа* | *Текстовые* | Вход/выход | "В конце литеральной последовательности не пробел" |

1. Алгоритм главной программы:
2. Ввод line
3. Вызов del\_on\_three(*line, num\_of\_numbers\_i\_dont\_know, sum, err, err2* )
4. С каждым вызовом del\_on\_three увеличивать sum на
5. Вызывать del\_on\_three до тех пор, пока *num\_of\_numbers\_i\_dont\_know != line.size() – 1 или до того, как алгоритм найдет ошибку в последовательности литералов(при ошибке возврат sum = -1)*
6. Сообщение о соответствии литеральной последовательности условию и о делимости на три, при значении sum == -1, сообщение об ошибке
7. Алгоритм рекурсивной функции

Алгоритм del\_on\_three:

1. Присвоить переменной that\_num символ из строки line по индексу num\_of\_numbers\_i\_dont\_know.

2. Установить флаг check\_num в значение false.

3. Начать цикл от 0 до 9.

4. Проверить, равно ли целое значение символа that\_num - '0' текущему значению цикла.

5. Если равно, установить check\_num в true и выйти из цикла.

6. Если check\_num по-прежнему false и that\_num не равен пробелу, вывести сообщение об ошибке err и вернуть -1.

7. Если that\_num равен пробелу и num\_of\_numbers\_i\_dont\_know не равно размеру строки минус 1, вывести сообщение об ошибке err и вернуть -1.

8. Если num\_of\_numbers\_i\_dont\_know равно размеру строки минус 1 и that\_num не равен пробелу, вывести сообщение об ошибке err2 и вернуть -1.

9. Если num\_of\_numbers\_i\_dont\_know равно размеру строки минус 1 и that\_num равен пробелу, вернуть sum.

10. Если that\_num не равен пробелу, вызвать функцию del\_on\_three с увеличенным на 1 значением num\_of\_numbers\_i\_dont\_know, увеличенным на значение that\_num - '0' значением sum, сообщением об ошибке "Вы ввели не число.\\n" и сообщением об ошибке "В конце литеральной последовательности не пробел\\n".

1. Таблица тестов

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Назначение теста | Входные данные | Выходные данные |
| 1 | Тест с правильно заданной последовательностью, делящейся на 3 | “123 “ | Заданная последовательность литер делится на 3 и заканчивается на пробел |
| 2 | Тест с правильно заданной последовательностью, не делящейся на 3 | “112 “ | заданная последовательность литер не делится на 3, но заканчивается на пробел |
| 3 | Тест с неправильно введённой последовательностью(без пробела на конце) | “111” | В конце литеральной последовательности не пробел  Введенная последовательность не соответствует условиям |
| 4 | Тест с неправильно введённой последовательностью(Ввод не числа или пробела на конце) | “Aslkfdjsiofhidshiofsaloihcdwapoqhdopqhpodhpodhposahoiahdpoajdhasidjpoqduqihopwdjpq[wdkd” | Вы ввели не число.  Введенная последовательность не соответствует условиям |

1. Программа на языке C++

<https://replit.com/@GieorghiiRasput/LABPROG4BBB#main.cpp>

//На 3 делятся только те числа, у которых сумма цифр делится на 3.

#include <iostream>

#include <sstream>

#include <string>

using namespace std;

int del\_on\_three(string line, int num\_of\_numbers\_i\_dont\_know, int sum,

string err, string err2) {

const char that\_num = line[num\_of\_numbers\_i\_dont\_know];

bool check\_num = false;

for (int i = 0; i < 10; i++) {

if (int(that\_num - '0') == i) {

check\_num = true;

break;

}

}

if (check\_num == false && that\_num != ' ') {

cout << err << "\n";

return (-1);

}

if (that\_num == ' ' && num\_of\_numbers\_i\_dont\_know != line.size() - 1) {

cout << err << "\n";

return (-1);

}

if (num\_of\_numbers\_i\_dont\_know == line.size() - 1 && that\_num != ' ') {

cout << err2 << "\n";

return (-1);

}

if (num\_of\_numbers\_i\_dont\_know == line.size() - 1 && that\_num == ' ') {

return sum;

}

if (that\_num != ' ') {

return del\_on\_three(line, num\_of\_numbers\_i\_dont\_know + 1,

sum + int(that\_num - '0'), "Вы ввели не число.\n",

"В конце литеральной последовательности не пробел\n");

}

return 0;

}

int main() {

setlocale(LC\_ALL, "ru");

string line;

cout << "Введите пследовательность литералов, оканчивающихся пробелом\n";

getline(cin, line);

int sum = del\_on\_three(line, 0, 0, "Вы ввели не число.\n",

"В конце литеральной последовательности не пробел\n");

cout << ((sum == -1)

? "Введенная последовательность не соответсувет условиям\n"

: ((sum % 3 == 0)

? "Заданная последовательность литер делится на 3 и "

"заканчивается на пробел\n"

: "заданная последовательность литер не делится на 3, но "

"заканчивается на пробел\n"));

}

# Использование функций. Задание В.

## Условие задачи.

*10в:* Создать файл, содержащий информацию об игроках футбольной команды. Структура информации: номер игрока, фамилия, имя, название клубов, за которые играл и в какие сезоны, дата рождения, количество забитых мячей. Определить список игроков команды, рассортированный по дате рождения. Ввести фамилию игрока и определить игроков этой команды, которые играли, хотя бы один сезон, против игрока фамилию которого вы ввели. Посчитать количество игроков, которые играли в разные сезоны в одном клубе.

## **Схема иерархии функций**

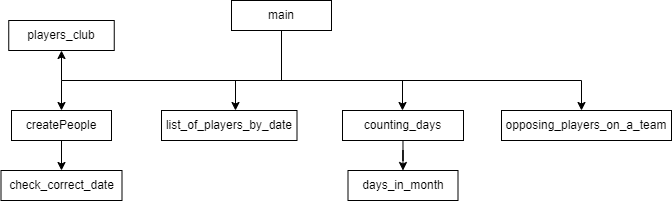


Рисунок 3 - Схема иерархии функций.

1. Таблицы внешних спецификаций для каждой функции

Таблица 3.3 – Внешние спецификации функции createPeople

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Имя | Назначение | Тип | Вх/Вых. | Диапазон |
| 1 | People | Новый элемент для заполнения массива команды игроков | Football\_chel | Выход | Переменная типа Football\_chel |
| 2 | number\_of\_player | Номер игрока | Строка | Вход | Символы ASCII кодировки |
| 3 | first\_name | Имя игрока | Строка | Вход | Символы ASCII кодировки |
| 4 | last\_name | Фамилия игрока | Строка | Вход | Символы ASCII кодировки |
| 5 | date\_of\_birth | Дата рождения игрока | Строка | Вход | Символы ASCII кодировки |
| 6 | killed\_balls | Количество забитых мячей | Строка | Вход | Символы ASCII кодировки |
| 7 | clubs\_seasons | Список сезонов и клубов игрока | Строка | Вход | Символы ASCII кодировки |

Таблица 3.4 – Внешние спецификации функции check\_correct\_date

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Имя | Назначение | Тип | Вх/Вых. | Диапазон |
| 1 | date | Дата в формате ДД.ММ.ГГГГ, вводимая пользователем | Строка | Вход/выход | Символы ASCII кодировки в формате ДД.ММ.ГГГГ |

Таблица 3.5 – Внешние спецификации функции list\_of\_players\_by\_date

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Имя | Назначение | Тип | Вх/Вых. | Диапазон |
| 1 | Player | Массив переменных типа Football\_chel | Одномерный массив Football\_chel | Вход | Переменные типа Football\_chel |
| 2 | num\_of\_players | Количество игроков | Целый | Вход | От 1 до 500 |
| 3 | С1 | Вывод списка игроков по дате их рождения(с помощью перевода даты в дни с помощью counting\_days) | Строка | Выход | Символы ASCII кодировки |

Таблица 3.6 – Внешние спецификации функции counting\_days

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Имя | Назначение | Тип | Вх/Вых. | Диапазон |
| 1 | date | Дата для перевода в целочисленное количество дней, прошедших с нулевого дня нулевого года | Строка | Вход | Строка в формате DD.MM.YYYY |
| 2 | days | Количество дней, прошедших с нулевого дня нулевого года | Целое | Выход | От 1 до 999999 |

Таблица 3.7 – Внешние спецификации функции days\_in\_month

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | month | Месяц для перевода в целочисленное количество дней, прошедших с 1 Января | Строка | Вход | От 1 до 12 |
| 2 | days | целочисленное количество дней, прошедших с 1 Января | Целое | Выход | От 0 до 364 |

Таблица 3.8 – Внешние спецификации функции opposing\_players\_on\_a\_team

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Имя | Назначение | Тип | Вх/Вых. | Диапазон |
| 1 | Player | Массив переменных типа Football\_chel | Одномерный массив Football\_chel | Вход | Переменные типа Football\_chel |
| 2 | num\_of\_players | Количество игроков | Целый | Вход | От 1 до 500 |
| 3 | С1 | Вывод списка противостоящих друг другу игроков в разных сезонах | Строка | Выход | Символы ASCII кодировки |

Таблица 3.9 – Внешние спецификации функции players\_club

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Имя | Назначение | Тип | Вх/Вых. | Диапазон |
| 1 | Player | Массив переменных типа Football\_chel | Одномерный массив Football\_chel | Вход | Переменные типа Football\_chel |
| 2 | num\_of\_players | Количество игроков | Целый | Вход | От 1 до 500 |
| 3 | С1 | Вывод числа игроков, которые играли в одном клубе в разные сезоны | Целый | Выход | От 0 до 500 |

## **Спецификация созданных пользовательских структур.**

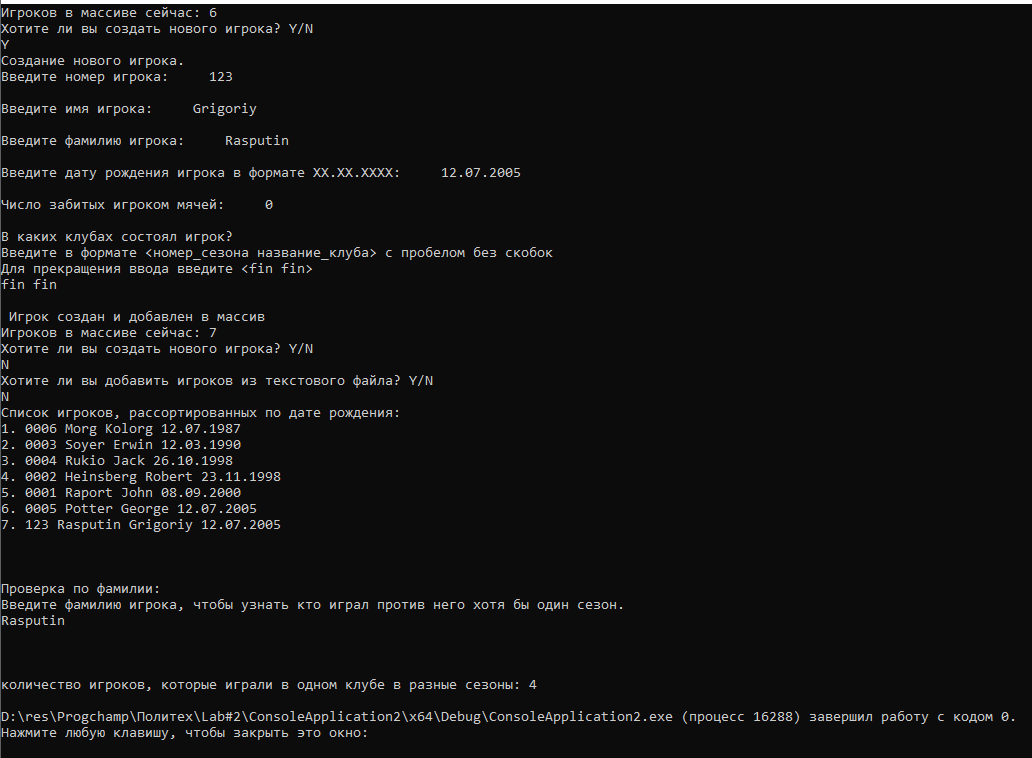
|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Назначение поля | Тип поля | Диапазон |
| number\_of\_player | Номер игрока | Строка | Символы ASCII кодировки |
| last\_name | Фамилия игрока | Строка | Символы ASCII кодировки |
| first\_name | Имя игрока | Строка | Символы ASCII кодировки |
| clubs\_seasons | Список клубов и сезонов, в которых участвовал игрок | Двумерный массив, содержащий данные | Символы ASCII кодировки |
| date\_of\_birth | Дата рождения игрока | Строка | Символы ASCII кодировки в формате “XX.YY.ZZZZ”, где XX – день, YY – месяц, ZZZZ – год рождения игрока |
| killed\_balls | Количество забитых мячей | Целое | От 0 до 999 |

## **Проектирование тестов.**

Таблица 4.1 – Таблица тестов для функции createPeople и check\_correct\_date

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Назначение теста | Входные данные | Выходные данные |
| 1 | Создание нового игрока | 123 Grigoriy Rasputin 12.07.2005 0 | C1 |
| 2 | допущение ошибки в дате рождения | 209 Georgiy Rasputin 123.124.1235123. 0 | C2 |

C1)



**C2)**

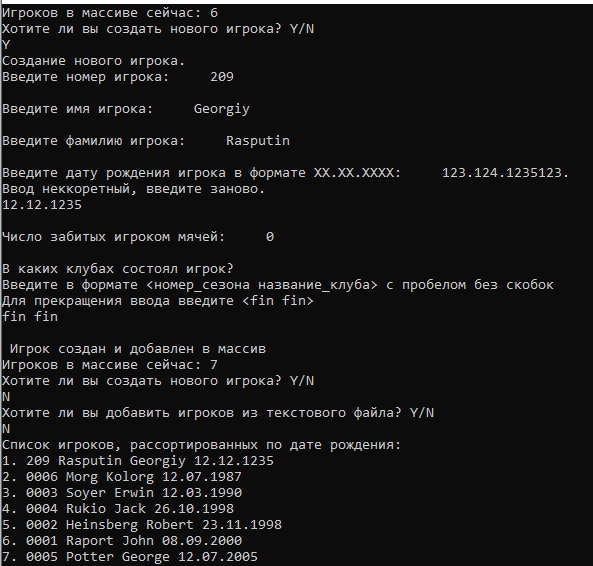
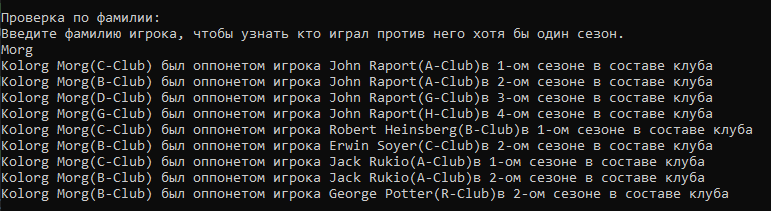


Таблица 4.2 – Таблица тестов для функции opposing\_players\_on\_a\_team

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Назначение теста | Входные данные | Выходные данные |
| 1 | Вывод соперников игрока Morg | Morg | C1 |
| 2 | Вывод соперников игрока Rukio | Rukio | C2 |

C1) 

C2)

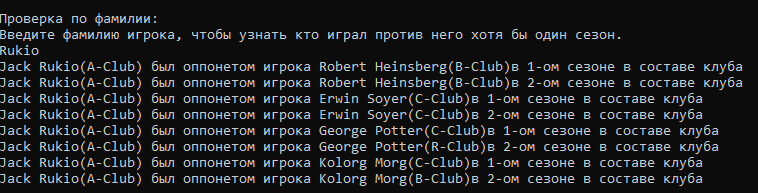
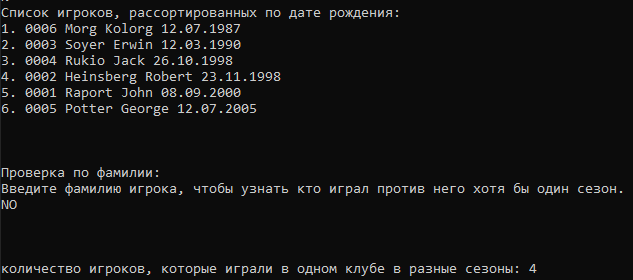


Таблица 4.3 – Таблица тестов для функции players\_club

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Назначение теста | Входные данные | Выходные данные |
| 1 | Вывод количества игроков, игравших в одном клубе без подключения игроков в файле | - | C1 |
| 2 | Вывод количества игроков, игравших в одном клубе с подключением игроков в файле | Текстовый файл "players.txt" | C2 |

С1)



C2)

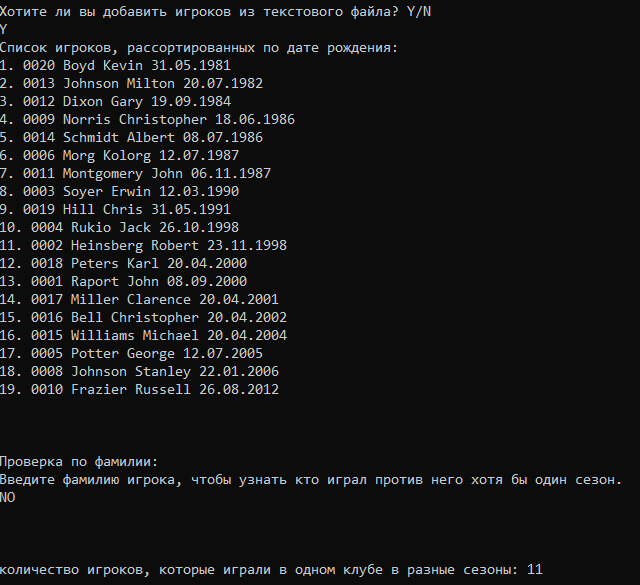
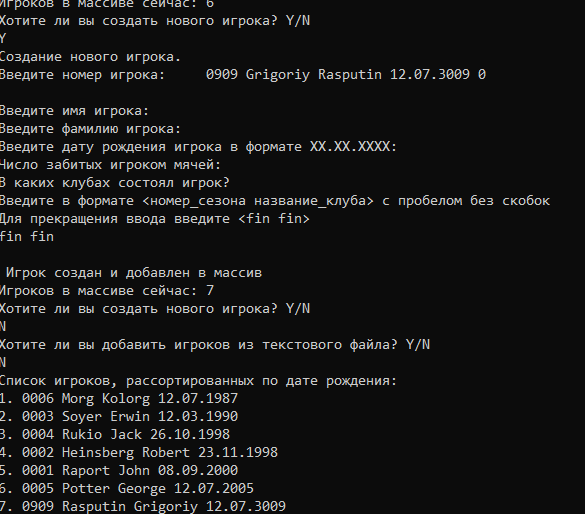


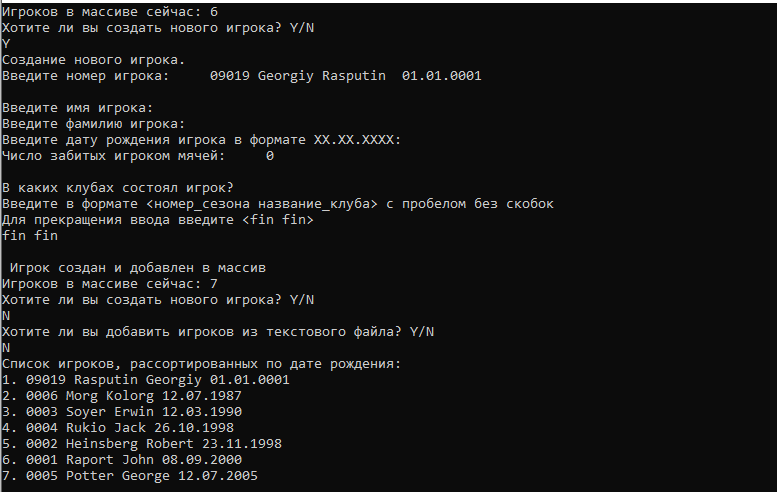
Таблица 4.4 – Таблица тестов для функции counting\_days и days\_in\_month и list\_of\_players\_by\_date

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Назначение теста | Входные данные | Выходные данные |
| 1 | Создание нового очень молодого игрока и отображение его в списке игроков по датам | 0909 Grigoriy Rasputin 12.07.3009 0 | C1 |
| 2 | Создание нового очень старого игрока и отображение его в списке игроков по датам | 09019 Georgiy Rasputin 01.01.0001 0 | C2 |

C1)



С2)



**5. Программа на языке C++**

## <https://replit.com/@GieorghiiRasput/Labprog3C#main.cpp>)

//Примечание: Ситуация не изменилась. В этом коде всё ещё 401 строка… поэтому я загружу его вместе с отчетом и файлом с игроками.