

Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Кафедра программных систем

**Практическая работа 2**

**Реализация алгоритма решения задачи согласно процедурной парадигме**

Выполнил: Мордвинцев Максим Витальевич

Группа № К3120

Проверила: Казанова П.П.

Цель работы:

Уяснить принципы разработки программ согласно процедурной парадигме и разработать программу, реализующую функциональные требования.

Представление реализации функциональной части программного продукта:

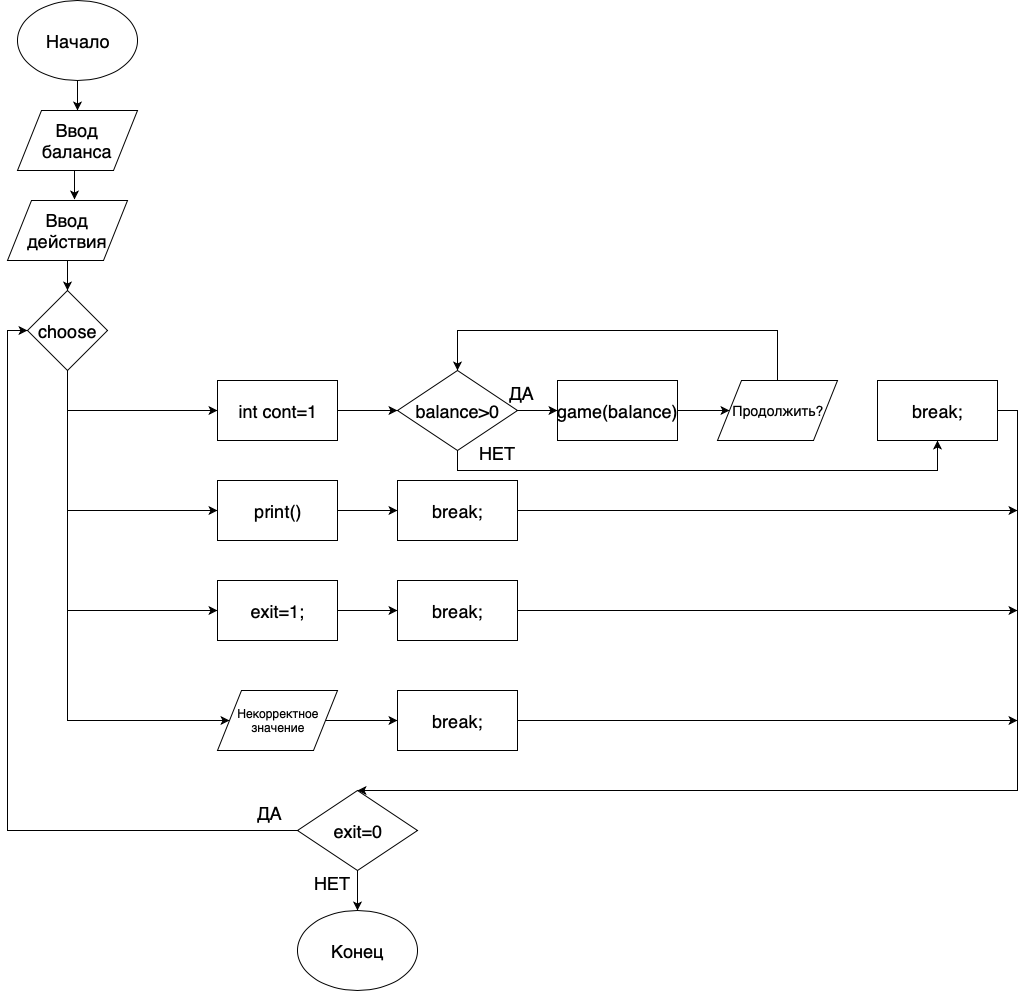
Для реализации программного продукта необходимо помимо основной функции *Main* обращаться также к функциям *Game* и *Print*

*Game* – получает на вход значение переменной *balance*, отвечающей за баланс игрока. Данная функция реализует сам процесс установления игроком ставки, случайного факта победы/проигрыша игрока и изменение значения переменной *balance*. Возвращает значение данной переменной.

*Print* – выводит значение баланса на экран.

1. Блок-схема функции Main()

Блок-схема функции ***Main()*** представлена на рисунке 1.



*Рисунок 1 – блок-схема функции* ***Main()***

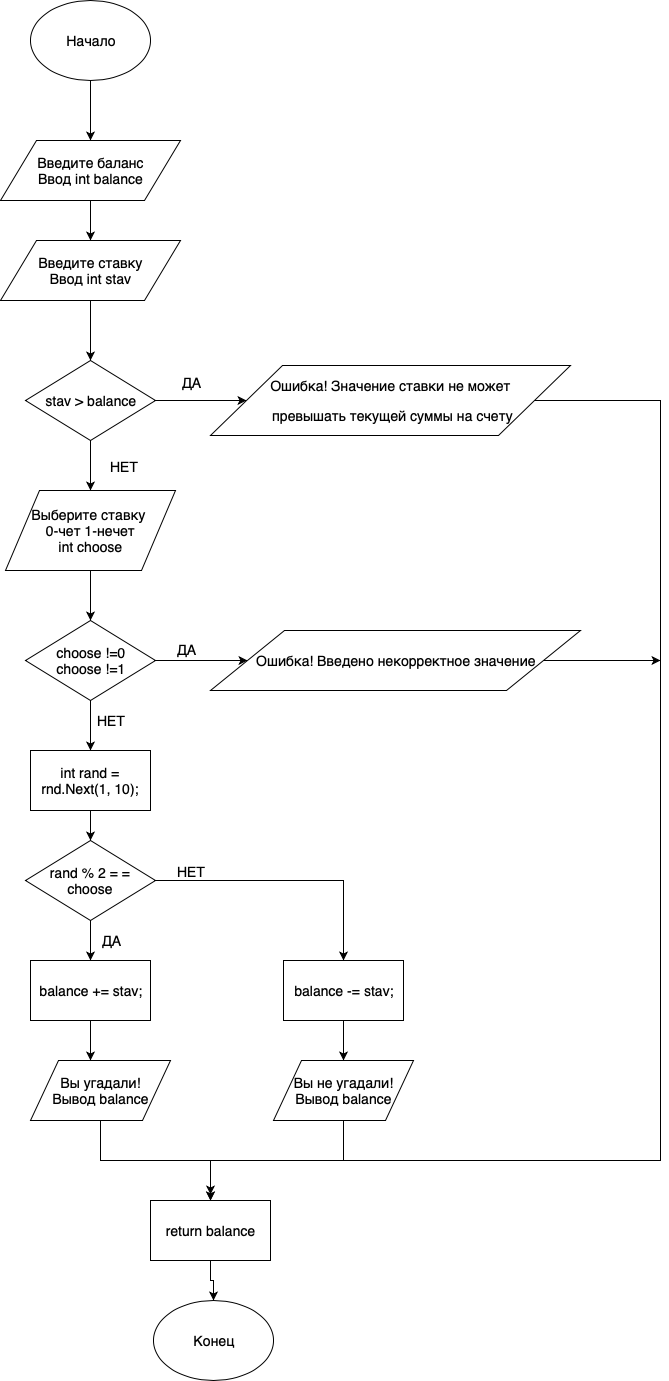
Код функции Main()

{  
                  Console.Write("\n@@@Введите начальное значение баланса:\n");  
                  int balance = int.Parse(Console.ReadLine());  
                  int exit = 0;  
  
             
            do  
            {  
                Console.Write("\n@@@Главное меню@@@\nВарианты действий: \n0 - играть\n1 - запрос баланса \n2 - выйти\n");  
                Console.Write("\nВыберите действие: ");  
  
                int choose = int.Parse(Console.ReadLine());  
  
                switch (choose)  
  
                {  
  
                    case 0:  
  
                        int cont = 1;  
  
                        while ((balance > 0) & (cont == 1))  
  
                        {  
  
                            balance = Game( balance );  
  
                            Console.Write("\nЖелаете продолжить игру? \n-0 - выйти в меню, \n-1 - продолжить, \n-2 - закончить игру\n");  
  
                            cont = int.Parse(Console.ReadLine());  
                            exit = cont - 1;  
  
  
                            if ((cont != 0) & (cont != 1) & (cont !=2))  
  
                            {  
  
                                Console.Write("\n---\nВведено неверное значение\n---\n");  
  
                                break;  
  
                            }  
  
                        }  
  
                        if (balance == 0) Console.Write("\nУ Вас больше нет средств!\n");  
                        exit = 1;  
                        break;  
  
                    case 1:  
  
                        Print(balance);  
  
                        Console.Write("\n-0 - выход в главное меню, \n-1 - закончить игру\n");  
  
                        exit = int.Parse(Console.ReadLine());  
                        Console.Write("\nВарианты действий: \n0 - играть\n1 - запрос баланса \n2 - выйти\n");  
  
                        break;  
  
                    case 2:  
  
                        exit = 1;  
  
                        break;  
                          
                    default:  
  
                        Console.Write("\nВведите корректное значение\n");  
                        break;  
  
                }  
  
            }  
  
            while (exit != 1);  
        }

}

1. Блок – схема функции Game()

Блок-схема функции *Game()* представлена на рисунке 2.



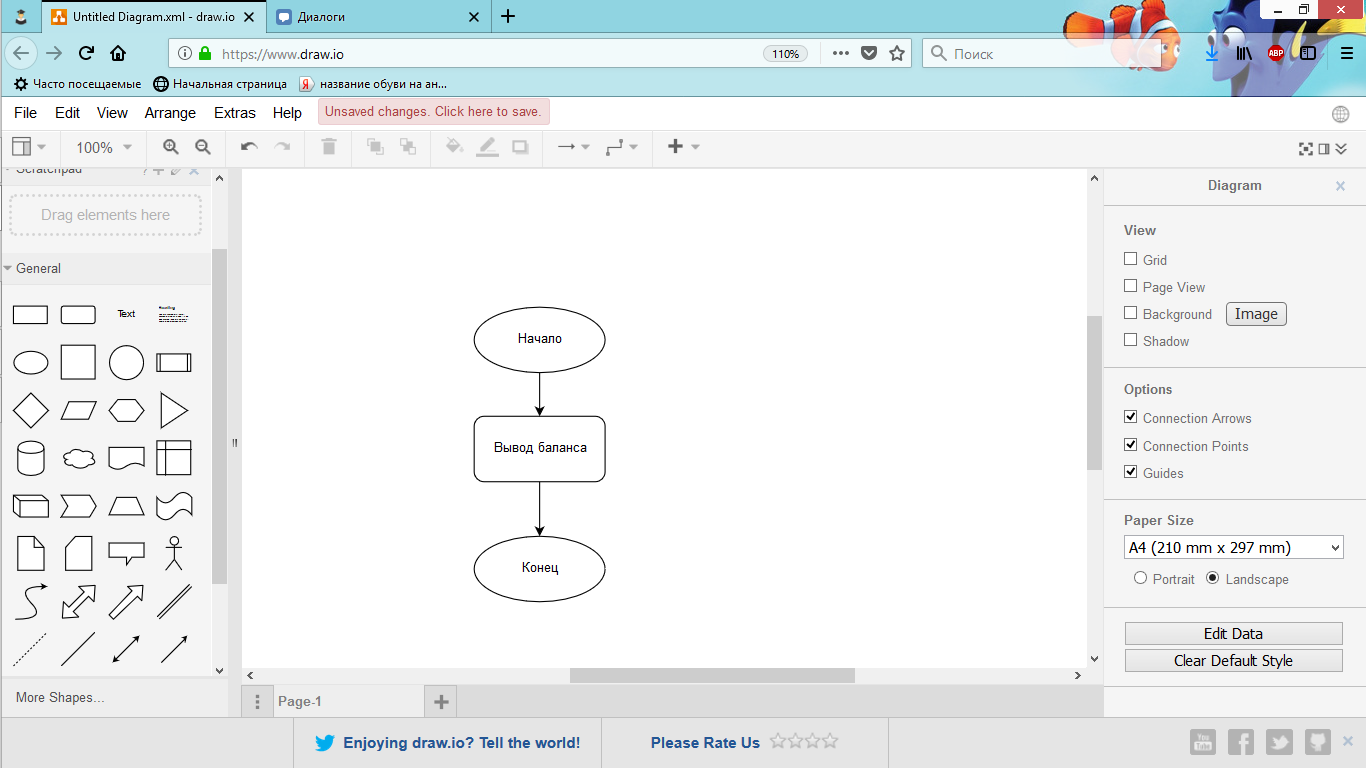
*Рисунок 2 – блок-схема функции* ***Game()***

Код функции Game()

{  
            Console.Write("\n@@@\nБаланс: {0}\n", balance);  
  
            Console.Write("Ваша ставка: ");  
  
            int stav = int.Parse(Console.ReadLine());  
  
            if (stav > balance)  
  
            {  
  
                Console.Write("Ошибка! Значение ставки не может превышать текущей суммы на счету\n");  
  
               return balance;  
  
            }  
  
            else  
            {  
  
                Console.Write("\nСделайте выбор: \n0 - чет\n1 - нечет\n");  
  
                int choose = int.Parse(Console.ReadLine());  
  
                if ((choose != 0) & (choose != 1))  
  
                {  
  
                    Console.Write("Введено неверное значение\n");  
  
                    return balance;  
  
                }  
  
                Random rnd = new Random();  
  
                int rand = rnd.Next(1, 10);  
  
                if (rand % 2 == choose)  
                {  
  
                    balance += stav;  
  
                    Console.Write("\n---\nВы угадали!\nБаланс:{0}\n", balance);  
                    Console.Write("Число:{0}\n---\n", rand);  
  
                }  
  
                else  
                {  
  
                    balance -= stav;  
  
                    Console.Write("\n---\nВы не угадали!\nБаланс:{0}\n", balance);  
                    Console.Write("Число:{0}\n---\n", rand);  
  
                }  
  
            }  
            return balance;  
        }

1. Блок – схема функции Print()

Блок-схема функции ***Print()*** представлена на рисунке 3.



*Рисунок 3 – блок-схема функции Print()*

Код функции Print()

{  
  
            Console.Write("\n---\nВаш баланс = {0}\n---\n", balance);  
        }

Вывод:

В ходе выполнения данной практической работы были составлены блок-схемы, отражающие функции для работы программного продукта «чёт-нечет», в соответствии с которыми был реализован функционал данного программного продукта, описанный в первой практической работе.