Министерство образования и науки РФ

Федеральное государственное автономное образовательное учреждение высшего образования

«Омский государственный технический университет»

|  |  |
| --- | --- |
| Факультет (институт) | *Информационных технологий и компьютерных систем* |
|  |  |
| Кафедра | *Прикладная математика и фундаментальная информатика* |
|  |  |

**Расчетно–графическая работа**

|  |  |
| --- | --- |
| по дисциплине | ***Алгоритмизация и программирование*** |
|  |  |
| на тему | Программная реализация задач |

Пояснительная записка

|  |  |
| --- | --- |
| **Шифр проекта** | 020–РГР–02.03.02–№ 17 – ПЗ |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **Студента** | | Николайчук Александр Алексеевич | | | | | |
|  |  |  |  | | фамилия, имя, отчество полностью | | | | | |
|  |  |  | Курс | 1 |  | Группа | | МО–241 | | |
|  |  |  |  |  |  | |  |  | |  |
|  | | | **Направление (специальность)** | | | | | ***02.03.03*** | | |
|  | | | Математическое обеспечение и администрирование информационных систем | | | | | | | |
|  |  |  | код, наименование | | | | | | | |
|  |  |  | Руководитель | | ***ст. преподаватель*** | | | | | |
|  |  |  | ученая степень, звание | | | | | |
|  |  |  | ***Федотова И.В.*** | | | | | | | |
|  |  |  | фамилия, инициалы | | | | | | | |
|  |  |  | Выполнил | |  | | | | | |
|  |  |  | дата, подпись студента | | | | | |
|  |  |  |  | | | | | | | |
|  |  |  | **Работа защищена с количеством баллов** | | | | | |  | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | дата, подпись руководителя |  |  |  |

Омск 2024

**Оглавление**

[Введение 3](#_Toc184470813)

[Постановка задачи 1 4](#_Toc184470814)

[Ход решения задачи 1 6](#_Toc184470815)

[Постановка задачи 2 8](#_Toc184470816)

[Ход решения задачи 2 9](#_Toc184470817)

[Постановка задачи 3 11](#_Toc184470818)

[Ход решения задачи 3 12](#_Toc184470819)

[Постановка задачи 4 14](#_Toc184470820)

[Ход решения задачи 4 15](#_Toc184470821)

[Заключение 17](#_Toc184470822)

[Список литературы 18](#_Toc184470823)

Введение

В современном мире программирование занимает особое место как в профессиональной деятельности, так и в повседневной жизни. Разработка программного обеспечения является важнейшей частью цифровой трансформации, которая охватывает все сферы экономики и социальной жизни. Одним из популярных языков программирования, который активно используется как в крупных корпоративных проектах, так и в разработке мобильных приложений и веб-сервисов, является C#. Этот язык, созданный компанией Microsoft, стал неотъемлемой частью экосистемы .NET, предоставляя разработчикам мощные инструменты для создания высококачественного и производительного ПО.

C# (произносится как "си шарп") был разработан в начале 2000-х годов и быстро завоевал популярность благодаря своей простоте синтаксиса, богатым возможностям объектно-ориентированного программирования и широким библиотекам, которые облегчают разработку. Язык идеально подходит как для начинающих программистов, так и для опытных специалистов, стремящихся повысить свою продуктивность и расширить кругозор в области программирования. Актуальность C# сегодня обусловлена не только его универсальностью, но и активной поддержкой Microsoft, что гарантирует регулярные обновления, новые функции и возможности, адаптированные к современным требованиям разработки.

Постановка задачи 1

Компания ХХХ

В компании “ХХХ” каждый служащий имеет уникальный идентификационный номер и может иметь уникальное имя. Между служащими установлены отношения типа “начальник – подчиненный”.

Требуется определить всех (т.е. не только непосредственных) подчиненных для заданного начальника. Начальник может быть задан как по идентификационному номеру, так и по уникальному имени.

Примечания:

* в именах нет различий между строчными и прописными буквами;
* служащий не может иметь более одного начальника;
* отношения “начальник – подчиненный” не образуют циклов.

Входной файл состоит из двух блоков:

* в первом блоке перечисляются отношения “начальник – подчиненный” (0 < количество отношений  5000). Каждое отношение задается двумя строками. В первой строке содержатся данные о начальнике, во второй – о его подчиненном. Каждая строка имеет формат XXXX\_NAME, где XXXX – уникальный идентификационный номер служащего, всегда состоящий из 4 цифр; NAME – уникальное имя служащего, которое может отсутствовать в строке, даже если оно присутствует в других строках; “\_” – пробел. В случае, если имя отсутствует, пробел не ставится. Длина XXXX\_NAME не превышает 30 символов. Между любыми двумя сотрудниками может быть только одно отношение “начальник – подчиненный”.
* Блок заканчивается словом “END” (без кавычек) в очередной строке;
* второй блок содержит одну строку – уникальный номер или уникальное имя начальника, подчиненных которого необходимо вывести. Данный сотрудник обязательно присутствует хотя бы в одном отношении “начальник – подчиненный”.

Выходные данные должны содержать найденных подчиненных, выведенных в соответствии со следующими правилами:

* подчиненных выводить по одному в строке, указывая для каждого его уникальный идентификационный номер и имя, разделенные пробелом;
* если имя подчиненного неизвестно, то вместо него выводить “Unknown Name” (без кавычек);
* подчиненные должны быть выведены в порядке возрастания уникального номера;
* в случае, если служащий не имеет подчиненных, выводить “NO” (без кавычек).

Ход решения задачи 1

Ввод данных для работы программы будет через файл. Вывод через консоль. Код программы представлен на рисунке 1.



Рисунок 1 – Код программы задачи 1

Данные с файла присваиваются массиву array (строки файла становятся элементами массива). Далее программа работает с ним. Переменной nach присваивается номер начальника с последней строки. В случае, если на последней строке дано его имя, то программа ищет номер начальника. В программу добавлена рекурсивная функция search, алгоритм которой описывается в блок схеме (рисунок 2). Примеры работы программы представлены на рисунке 3.

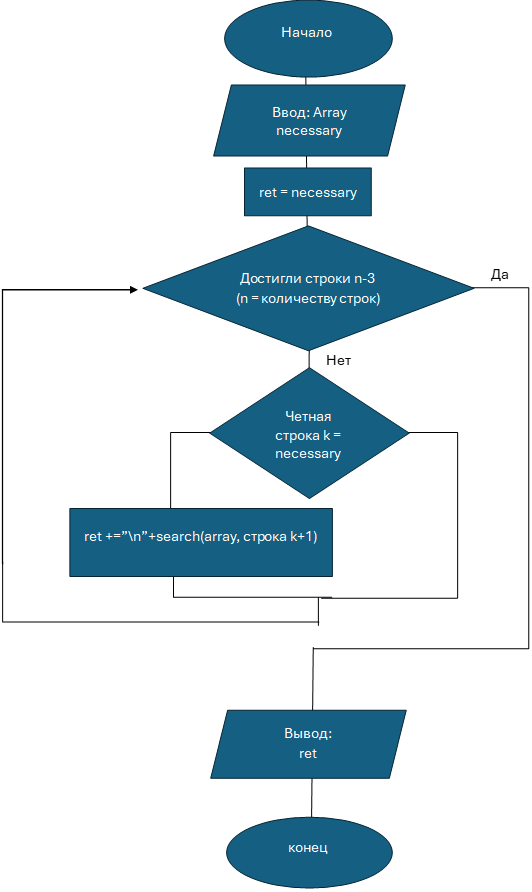


Рисунок 2 – Блок схема функции задачи 1

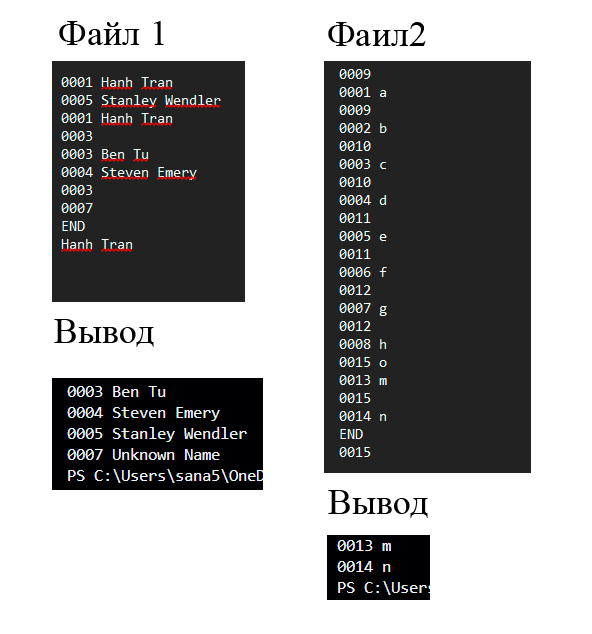


Рисунок 3 – Примеры работы программы задачи 1

Постановка задачи 2

Интенсификация производства

Перед коллективом предприятия “Ни шагу назад” была поставлена задача наращивать каждый день производство продукции на 1.

Требуется определить, какой суммарный объем продукции будет выпущен предприятием за заданный период, если в первый день периода предприятие выпускало P единиц продукции.

Примечания:

* период задается в виде двух календарных дат;
* длительность периода лежит в диапазоне от 1 до 60000;
* високосные годы учитываются по упрощенному правилу: високосным считается год, делящийся нацело на 4;
* день начала периода и день его окончания учитываются при подсчете суммарного объема продукции и длительности периода;
* все даты заданы корректно.
* Входные данные содержат:
* в первой строке – дата начала периода в формате ДД.ММ.ГГГГ;
* во второй строке – дата окончания периода в формате ДД.ММ.ГГГГ;
* в третьей строке целое число – начальный выпуск продукции P (0 ≤ P ≤ 5000).

Выходные данные должны содержать суммарный объем продукции.

Ход решения задачи 2

Ввод и вывод данных происходит через консоль. Код программы представлен на рисунке 4.

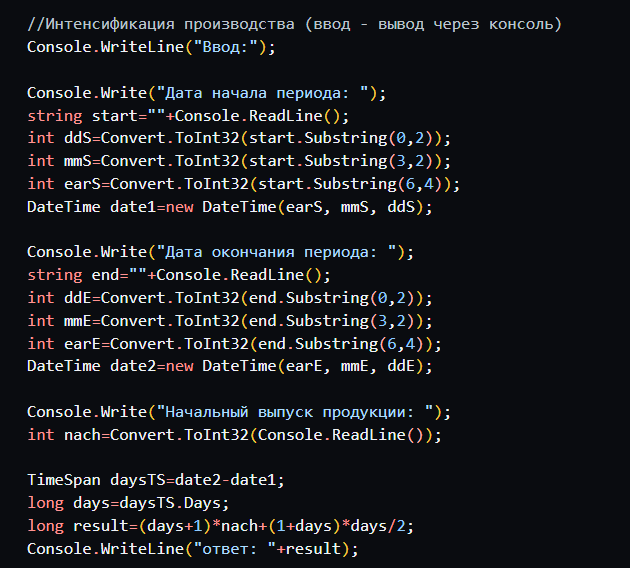


Рисунок 4 – Код программы задачи 2

В программе используются классы DateTime (представляет время, обычно выраженное как дата и время суток) и TimeSpan (Представляет интервал времени).

Программа считывает дату из первой строки и присваивает её в объект date1. Дата из второй строки присваивается объекту date2. Количество дней и часов между date1 и date2 записывается в dateTS. Разница дней (далее РД) присваивается переменной days. Третья строка (начальный выпуск продукции, далее НВП) присваивается переменной nach.

Суммарный объём продукции (далее СОП) вычисляется по формуле:

Получить её можно путем преобразования следующего равенства:

В данном равенстве НВП повторяется РД+1 раз. Сумма нарощенной продукции вычисляется по формуле алгебраической прогрессии, где начальный элемент будет равен 1, конечный – РД, количество элементов так же равно РД. Примеры работы программы представлены на рисунке 5.

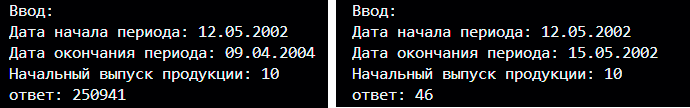


Рисунок 5 – Примеры работы программы задачи 2

Постановка задачи 3

Отгадай число.

Известен следующий фокус. Фокусник предлагает выполнить действия следующего характера: задумайте число, прибавьте 2, умножьте на 3, отнимите 5, отнимите задуманное число и т.д. После этого по названному полученному результату фокусник определяет задуманное число.

Пусть задумано некоторое целое число X. Требуется после выполнения ряда действий по известному результату R определить это число.

Примечание:

* гарантируется, что имеется только один ответ;
* гарантируется, что во время выполнения действий какие-либо промежуточные результаты не превышают по модулю 2 000 000 000.

Входной файл:

* Первая строка содержит количество действий N (0 £ N £ 100).
* Следующие N строк содержат описания действий в последовательности их выполнения, причем в каждой строке указывается одно действие в формате S V, где:
* S - тип действия, состоящий из одного символа: "\*" - умножить; "-" - отнять; "+" - прибавить;
* V - аргумент действия. Может быть целым числом (|V| £ 100) либо символом "x". Символ "x" может применяться только в действиях "-" и "+" и обозначает, что нужно отнять или прибавить задуманное число, соответственно.
* Последняя строка содержит результат R (|R| £ 2 000 000 000).
* Выходные данные должны содержать одно целое число - задуманное число X.

Ход решения задачи 3

Ввод данных через файл. Вывод через консоль. Код программы представлен на рисунке 6.



Рисунок 6 – Код программы задачи 3

Программа присваивает данные строк элементам массива array. Алгоритм считает количество x (далее A) и количество единиц (далее B), где x – задуманное число.

Имеем формулу: .

Из неё следует:

Количество х увеличивается (уменьшается) при прибавлении (вычитании) х. Количество единиц увеличивается (уменьшается) при прибавлении (вычитании) данного целого числа. Если действие – умножение, то оба числа умножаются на данное целое число.

Далее программа выводит х. Примеры работы представлены на рисунке 7.

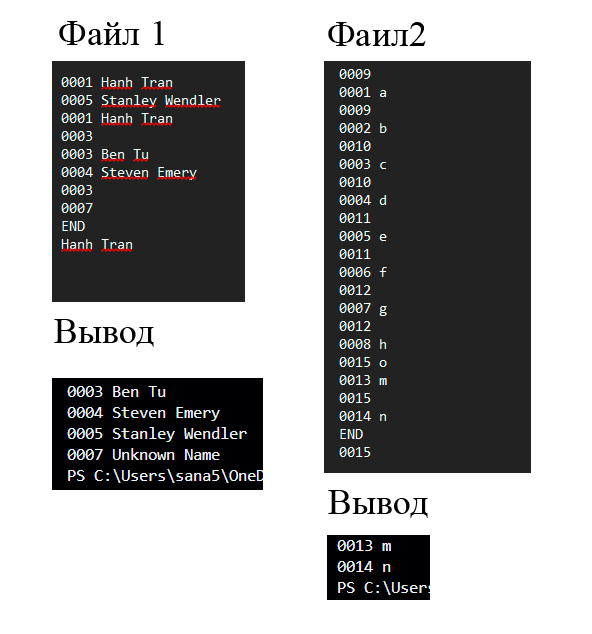


Рисунок 7 – Примеры работы программы задачи 3

Постановка задачи 4

Крестьянин и черт

Идет крестьянин и плачется: "Эхма! Жизнь моя горькая! Заела нужда совсем! Вот в кармане только несколько монет, да и те сейчас нужно отдать. И как это у других бывает, что на всякие свои деньги они еще деньги получают? Хоть бы кто помочь мне захотел".

Только успел это сказать, как глядь, а перед ним черт стоит и говорит: "Вот видишь этот мост через реку. Стоит тебе перейти через мост, и у тебя будет вдвое больше денег, чем есть. Перейдешь опять, и снова станет вдвое больше. Но за то, что я у тебя деньги удваиваю, после каждого перехода ты мне должен отдавать по K монет".

"Ой ли," - сказал крестьянин -"ну-ка, попробуем". Перешел мост, и деньги у него удвоились. Отдал он черту K монет, перешел мост еще раз, и опять деньги удвоились. Снова отдал крестьянин черту K монет.

Однако после Z переходов и отдач черту по K монет оказалось, что у крестьянина не осталось ни одной монеты.

Требуется определить, сколько комбинаций условий перехода через мост может быть, если известно, что у крестьянина изначально было не более MaxN монет. Комбинацией условий перехода является тройка чисел N, K, Z, где N - начальное количество монет у крестьянина, K - количество монет, отдаваемых черту после каждого перехода, Z - количество переходов. Естественно, что для этой тройки должно выполняться условие, что после Z циклов у крестьянина не должно остаться монет.

Входные данные содержат целое число MaxN - максимальное количество, которое может быть изначально у крестьянина (1 < MaxN < 2000000000).

Выходные данные должны содержать одно целое число - количество комбинаций условий перехода через мост.

Ход решения задачи 4

Ввод и вывод данных происходит через консоль. Код программы представлен на рисунке 8.

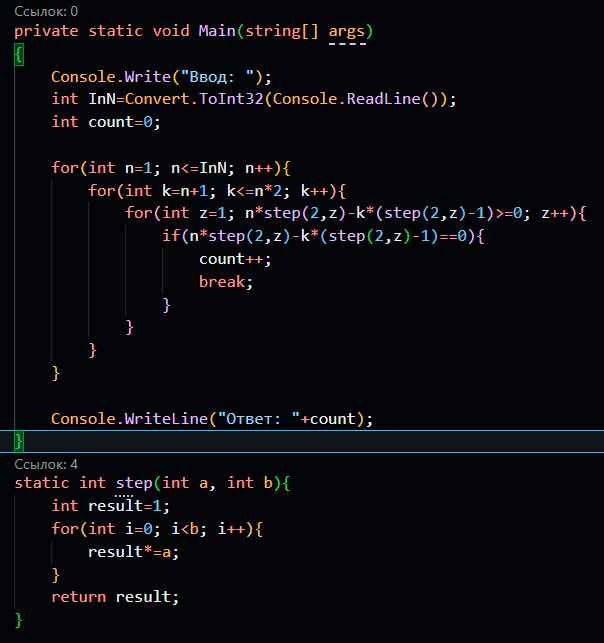


Рисунок 8 – Код программы задачи 4

Алгоритм работает по следующей формуле:

, где *R* – количество монет у крестьянина в итоге.

Итоговое количество монет у крестьянина должно быть рано 0 (по условию), поэтому программа считает только те комбинации, результатом которых у крестьянина останется 0 монет. Количество комбинаций присваивается переменой *count*. Функция *step* служит для возведение числа *a* в степень *b*. В итоге программа выводит count в консоль. Примеры работы программы представлены на рисунке 9.



Рисунок 9 – Примеры работы программы задачи 4

Заключение

В результате работы с языком C# была успешно решена серия из четырех задач, каждая из которых представляла собой уникальный вызов и требовала применения различных аспектов языка и его стандартной библиотеки.

Таким образом, выполненные задачи не только углубили практические навыки работы с языком C#, но и сформировали более глубокое понимание принципов программирования в целом. Каждая из решенных задач представила возможность применить теоретические знания на практике и улучшить логическое мышление, что, безусловно, будет полезно в дальнейшей образовательной и профессиональной деятельности. Стремление к постоянному обучению и развитию навыков программирования остается важным аспектом успешной карьеры в области разработки программного обеспечения.

Список литературы

1. Microsoft Learn [https://learn.microsoft.com/en-us 01.12.2024](https://learn.microsoft.com/en-us%2001.12.2024) (дата обращения: 01.12.2024).
2. METANIT.COM <https://metanit.com/> (дата обращения: 19.11.2024).
3. Stack Overflow <https://stackoverflow.com/> (дата обращения: 30.10.2024).
4. Habr <https://habr.com/ru/articles/> (дата обращения: 21.11.2024).
5. Википедия <https://ru.wikipedia.org> (дата обращения 22.11.2024).