

Технологии программирования для мобильных приложений

ЛАБОРАТОРНАЯ РАБОТА № 3. Консольные приложения и их сборка

Содержание

Учебные материалы и руководства.....	3
Стандарты кодирования для оформления кода.....	3
1. Правила именования файлов.....	3
2. Организация программы согласно модели КИС.....	3
3. Организация файлов.....	4
3.1 Заголовочные файлы.....	4
3.2 Файлы с исходным кодом.....	5
4. Табуляция.....	6
5. Комментарии.....	6
Дополнительные материалы по стандартам кодирования.....	6
Сборка приложений.....	7
Структура проекта согласно модели КИС и правил сборки.....	8
Сборка проекта с Github Action.....	9
Дополнительные материалы по сборке.....	11
ЗАДАНИЯ.....	11
Требования к репозиториям.....	11
Требования к отчёту.....	11
Критерии оценивания.....	13
ЗАДАНИЕ 1. КОНСОЛЬНЫЕ ПРИЛОЖЕНИЯ В REPL.IT.....	13
ЗАДАНИЕ 2. ИЗУЧАЕМ КУРС GitHub Actions: Hello World.....	14
ЗАДАНИЕ 3. ИСПОЛЬЗУЕМ СТРУКТУРЫ ДЛЯ ПРЕДСТАВЛЕНИЯ ДАННЫХ.....	15
ЗАДАНИЕ 4. ЗАПИСЬ И ЧТЕНИЕ ТЕКСТОВЫХ ФАЙЛОВ.....	15
Варианты.....	16
Вариант 1.....	16
Вариант 2.....	16
Вариант 3.....	17
Вариант 4.....	17
Вариант 5.....	18
Вариант 6.....	18
Вариант 7.....	18
Вариант 8.....	19
Вариант 9.....	19
Вариант 10.....	20
Вариант 11.....	20
Вариант 12.....	21
Вариант 13.....	21
Вариант 14.....	22
Вариант 15.....	22
Вариант 16.....	23
Вариант 17.....	23
Вариант 18.....	24

Вариант 19.....	24
Вариант 20.....	25
Вариант 21.....	25
Вариант 22.....	26
Вариант 23.....	26
Вариант 24.....	27
Вариант 25.....	27
Вариант 26.....	28
Вариант 27.....	28
Вариант 28.....	28
Вариант 29.....	29
Вариант 30.....	29
Вариант 31.....	30
Вариант 32.....	30
Вариант 33.....	31
Вариант 34.....	31
Вариант 35.....	32
Вариант 36.....	32
Вариант 37.....	33
Вариант 38.....	33
Вариант 39.....	34
Вариант 40.....	34
КОНТРОЛЬНЫЕ ВОПРОСЫ.....	34
Материалы к занятию.....	36
Как быстро ознакомиться с отличительными характеристиками языка программирования?.....	36
Интегрированные веб-среды разработки и обучения.....	36
Команды терминала в Mac OS X. Редактор текстов nano.....	36
Полный синтаксис команды запуска:.....	37
Чтение и запись файлов.....	38

Цель работы — овладеть навыками разработки консольных приложений на языке C, стандартами кодирования, сборки приложений и публикации изменений в репозиторий.

Продемонстрировать владение облачной IDE <https://repl.it> и редактор nano (vi) в macOS, для компиляции в macOS — gcc, для сборки — make, для публикации изменений в репозиторий — git.

Для достижения цели необходимо выполнить следующие задачи:

- изучить учебный материал по альтернативным IDE и тестовым редакторам для работы с кодом;
- изучить учебный материал по стандартам кодирования для языка C;
- изучить учебный материал по сборке приложений и утилите make;
- познакомиться с альтернативными IDE;
- компилировать приложение в консоли macOS;
- выполнить сборку приложений с помощью утилиты make;
- использовать модель управления репозиторием [A simple git branching model](#);
- опубликовать изменения в репозиторий.

Учебные материалы и руководства

Стандарты кодирования для оформления кода

1. Правила именования файлов

В операционной системе Unix используются определённые расширения файлов, обрабатываемых компилятором языка C.

Используйте следующие расширения файлов:

- для файлов с исходным кодом на языке C — расширение .c;
- для файлов с исходным кодом на языке assembler — расширение .s;
- для перемещаемых объектных файлов — расширение .o;
- для заголовочных файлов — расширение .h;
- для архивных библиотечных файлов — расширение .a;
- для файлов с исходным кодом на скриптовом языке C shell — расширение .csh;
- для файлов с исходным кодом командной оболочки Unix (Bourne shell) — расширение .sh;
- для файлов с кодом генератора синтаксических анализаторов yacc — расширение .y;
- для файлов с кодом генератора лексических анализаторов lex — расширение .l.

2. Организация программы согласно модели КИС

Любая программа имеет свой *репозиторий* - рабочий каталог, в котором находятся исходники, сценарии сборки (Makefile) и прочие файлы, относящиеся к проекту.

Разделяйте программы на модули. Модульный код с чётким разделением меньше подвержен программным ошибкам, проще сопровождается и легче расширяется.

Модуль – это группа процедур и данных, совместно реализующих некую абстракцию, например: таблицу символов или задачу управления памятью и т.д.

Модуль состоит из двух файлов: заголовочного (интерфейсного) (файл с расширением .h) и файла с исходным кодом на языке C (файлы с расширением .c).

Модель КИС (Клиент-Интерфейс-Сервер) — это элегантная концепция распределения исходного кода в репозитории, в рамках которой все исходники можно поделить на *клиенты*, *интерфейсы* и *серверы*.

Сервер предоставляет услуги. В нашем случае это могут быть функции, структуры, перечисления, константы, глобальные переменные и проч. В языке C++ это чаще всего классы или иерархии классов. Любой желающий (клиент) может воспользоваться предоставленными услугами, то есть вызвать функцию со своими фактическими параметрами, создать экземпляр структуры, воспользоваться константой и т. п. В C++, как правило, клиент использует класс как тип данных и использует его

члены.

Часто бывает, что клиент сам становится сервером, точнее начинает играть роль промежуточного сервера. Например, в примере Hello World можем реализовать функцию `print_hello()` (клиент), которая будет пользоваться услугами стандартной библиотеки языка C (сервер), вызывая функцию `printf()`. Однако в дальнейшем функция `print_hello()` сама становится сервером, предоставляя свои услуги функции `main()`.

Клиент с сервером должны "понимать" друг друга, иначе взаимодействие невозможно. **Интерфейс** (протокол) — это условный набор правил, согласно которым взаимодействуют клиент и сервер. Обычно для организации интерфейсов используются *объявления* (прототипы), которые помещаются чаще всего в *заголовочные файлы*. Таким образом интерфейс (набор объявлений, в данном случае - в заголовочном файле) - это публичная оферта сервера клиенту. Включение заголовочного файла директивой `#include` - это акцепт или подпись.

Пример модели КИС для проекта Hello World. Создадим файл `hello.h`:

```
/* hello.h */
void print_hello (void);
```

Теперь включим этот файл в `main.c`:

```
/* main.c */
#include "hello.h"

int main (void)
{
    print_hello ();
}
```

3. Организация файлов.

Файл должен разделяться на секции. Секции в файле должны отделяться пустыми строками.

Хоть формально максимальные требования к размеру файла не предъявляются, не рекомендуется использовать более 3000 строк кода в одном файле, так как код будет трудным для восприятия.

Так же следует избегать слишком длинных строк. Не все терминалы и текстовые редакторы хорошо работают с ними.

3.1 Заголовочные файлы.

Порядок разделов для `.h` файлов выглядит следующим образом:

1. **Идентификатор автора:** файл должен начинаться с указания автора кода.
2. **#ifndef:** Заголовочный файл часто может подключаться в нескольких файлах. Чтобы избежать многократного включения заголовочного файла в рамках одного

проекта используется директива `#ifndef`. Директива `#ifndef` должна следовать сразу после указания автора.

3. **Блок комментариев:** после `#ifndef` должен идти блок комментариев, описывающий назначение объектов в файле (функций, глобальных переменных и т.д.). Описание должно быть коротким и по делу.
4. **`#includes`:** После блока комментариев подключаются дополнительные `.h` файлы.
5. **`#defines`:** далее следуют определения с помощью директивы `#defines`, относящиеся к модулю в целом, но не к отдельным элементам структур.
6. **`typedefs`:** после определения идентификаторов с помощью `#define`, идут определения новых типов данных с помощью оператора `typedef`.
7. **Декларация структур:** если определение директивы `#define`, относится к глобальным данным (например, флаги слова), то определение должно следовать сразу же после объявления этих данных.
8. **Декларация функций:** объявление функций должно идти после декларации структур. Все внешние функции должны быть объявлены, даже те, которые возвращают `int` или ничего не возвращают (объявляются как возвращающие `void`).
9. **Объявление внешних переменных:** далее идет декларация внешних переменных.
10. Не определяйте статические и глобальные переменные в заголовочных файлах.

3.2 Файлы с исходным кодом

Порядок разделов для `.c` файлов приведен ниже:

1. **Идентификатор автора:** файл должен начинаться с указания автора кода.
2. **Блок комментариев:** сразу после имени автора должен идти блок комментариев, описывающий содержимое файла.
3. **`#includes`:** далее должно идти подключение заголовочных файлов.
4. **`typedefs` and `#defines`:** после подключения заголовочных файлов должны следовать определения с помощью операторов `typedef` и `#define`.
5. **Определение структур:** далее идет определение структур.
6. **Глобальные переменные:** они должны располагаться после определения структур.
7. **Декларация функций:** далее должна следовать декларация функций. Декларироваться должны только приватные функции. Публичные функции уже объявлены в заголовочных файлах, которые подключены выше. Хотя C-компилятор и не требует объявления функций возвращающих `int`, декларировать такие функции считается хорошим тоном.
8. **Определения:** Каждой функции должен предшествовать блок комментариев.

Функции должны следовать в определенном порядке. Лучше если они будут упорядочены сверху вниз, чем снизу вверх. Функции схожего уровня «абстракции» лучше располагать рядом. Функции желательно размещать как можно ближе к месту их вызова. При наличии большого числа функций практически не имеющих взаимосвязи можно располагать их в алфавитном порядке.

4. Табуляция.

- а) Каждый новый уровень должен отделяться табуляцией (8 пробелов).
- б) Каждая строка должна быть ограничена размером страницы (код не должен переноситься на новую строку).
- в) Вставляйте табуляцию при объявлении переменных для выравнивания имен переменных.
- г) Имена и параметры функций должны идти в колонку по одному на строке.

5. Комментарии.

Комментарии должны содержать информацию, относящуюся только к коду программы. Комментарии, вводящие в заблуждение, хуже, чем отсутствие комментариев. В комментариях должны быть отражены:

1. Должны быть хорошо описаны нетривиальные решения и неординарные ситуаций. Следует избегать дублирования информации, возникающее при комментировании очевидного кода.
2. В комментариях должны быть отражены все изменения в функциях, внешних переменных и указателях.
3. При использовании оператора case следует вводить соответствующие комментарии.
4. Все декларации должны комментироваться, за исключением декларации одноразовых счетчиков (например, для организации цикла) и очевидных имен констант. Комментарии при декларации переменных должны быть выровнены в один столбец.
5. В комментарии не должны включаться данные о том, как создан файл или в какой директории он должен располагаться.
6. Комментарии не должны содержать списки авторов и историю изменений кода.

Существует три стиля комментариев: блочный, однострочный, завершающий строку с кодом.

Дополнительные материалы по стандартам кодирования

- Стандарт кодирования GNU — http://www.opennet.ru/docs/RUS/coding_standard/ — изучить разделы 8-11 для оформления кода.

- Стиль C и стандарты программирования — <http://all-ht.ru/inf/prog/c/style.html>
- Стандарт кодирования GNU — http://www.opennet.ru/docs/RUS/coding_standard/ — изучить раздел 5 для ознакомления с правилами оформления Makefile-ов.

Сборка приложений

Утилита make работает со своими собственными сценариями. Сценарий записывается в файле с именем Makefile и помещается в репозиторий (рабочий каталог) проекта. Сценарии утилиты make просты и многофункциональны, а формат Makefile используется повсеместно (и не только на Unix-системах). Дошло до того, что стали создавать программы, генерирующие Makefile'ы. Самый яркий пример - набор утилит GNU Autotools. Самое главное преимущество make - это "интеллектуальный" способ рекомпиляции: в процессе отладки make компилирует только измененные файлы.

То, что выполняет утилита make, называется сборкой проекта, а сама утилита make относится к разряду сборщиков.

Любой Makefile состоит из трех элементов: *комментарии*, *макроопределения* и *целевые связи* (или просто *связки*). В свою очередь связи состоят тоже из трех элементов: *цель*, *зависимости* и *правила*.

Сценарии make используют однострочные комментарии, начинающиеся с литеры # (решетка).

Макроопределения позволяют назначить имя практически любой строке, а затем подставлять это имя в любое место сценария, где должна использоваться данная строка. Макросы Makefile схожи с макроконстантами языка C.

Связки определяют: 1) что нужно сделать (*цель*); 2) что для этого нужно (*зависимости*); 3) как это сделать (*правила*). В качестве цели выступает имя или макроконстанта. **Зависимости** — это список файлов и целей, разделенных пробелом. **Правила** — это команды передаваемые оболочке.

Теперь рассмотрим пример сценария сборки Makefile для мультифайлового проекта Hello World:

```
# Makefile for Hello World project

hello: main.o hello.o
    gcc -o hello main.o hello.o

main.o: main.c
    gcc -c main.c

hello.o: hello.c
    gcc -c hello.c

clean:
    rm -f *.o hello
```

Первая строка — комментарий. Здесь можно писать все, что угодно. Комментарий начинается с символа # (решетка) и заканчивается символом новой строки. Далее по порядку следуют четыре связи: 1) связь для компоновки объектных файлов main.o и hello.o; 2) связь для компиляции main.c; 3) связь для компиляции hello.c; 4) связь для очистки проекта.

Первая связь имеет цель hello. Цель отделяется от списка зависимостей

двоеточием. Список зависимостей отделяется от правил символом новой строки. А каждое правило начинается на новой строке с символа табуляции. В нашем случае каждая связка содержит по одному правилу. В списке зависимостей перечисляются через пробел вещи, необходимые для выполнения правила. В первом случае, чтобы скомпоновать бинарник, нужно иметь два объектных файла, поэтому они оказываются в списке зависимостей. Изначально объектные файлы отсутствуют, поэтому требуется создать целевые связки для их получения. Итак, чтобы получить main.o, нужно откомпилировать main.c. Таким образом файл main.c появляется в списке зависимостей (он там единственный). Аналогичная ситуация с hello.o. Файлы main.c и hello.c изначально существуют (мы их сами создали), поэтому никаких связей для их создания не требуется.

Особую роль играет целевая связка clean с пустым списком зависимостей. Эта связка очищает проект от всех автоматически созданных файлов. В нашем случае удаляются файлы main.o, hello.o и hello. Очистка проекта бывает нужна в нескольких случаях: 1) для очистки готового проекта от всего лишнего; 2) для пересборки проекта (когда в проект добавляются новые файлы или когда изменяется сам Makefile; 3) в любых других случаях, когда требуется полная пересборка (например, для измерения времени полной сборки).

Теперь осталось запустить сценарий. Формат запуска утилиты make следующий:

```
make [опции] [цели...]
```

Опции make нам пока не нужны. Если вызвать make без указания целей, то будет выполнена первая попавшаяся связка (со всеми зависимостями) и сборка завершится. Нам это и требуется:

```
$ make
gcc -c main.c
gcc -c hello.c
gcc -o hello main.o hello.o
$ ls
hello*  hello.c  hello.o  main.c  main.o  Makefile
$ ./hello
Hello World
$
```

В процессе сборки утилита make пишет все выполняемые правила. Проект собран, все работает.

Структура проекта согласно модели КИС и правил сборки

Структура проектов для задач 2 и 3 текущей лабораторной работы должна выглядеть следующим образом:

- имя_проекта /
 - bin /
 - doc /

- *README*
- *include /*
- *obj /*
- *src /*
- *Makefile*

В каталог *bin* помещаются результаты компиляции исполняемые файлы, в каталог *obj* — объектные файлы. Каталог *bin/* можно "безболезненно" удалить — при следующей компиляции он будет автоматически создан заново.

В директорию *doc* добавляются различные текстовые файлы — документация, замечания, список ошибок и тому подобное. Здесь же располагается и файл *README*, в котором содержится описание задачи и ссылка на отчёт.

В каталог *src* — исходные тексты программы. Внутри директории *src* имеется своя иерархия каталогов, отражающая логическую структуру программы.

В каталоге *include* — заголовочные файлы.

В корневой каталоге проекта находится *make*-файл проекта.

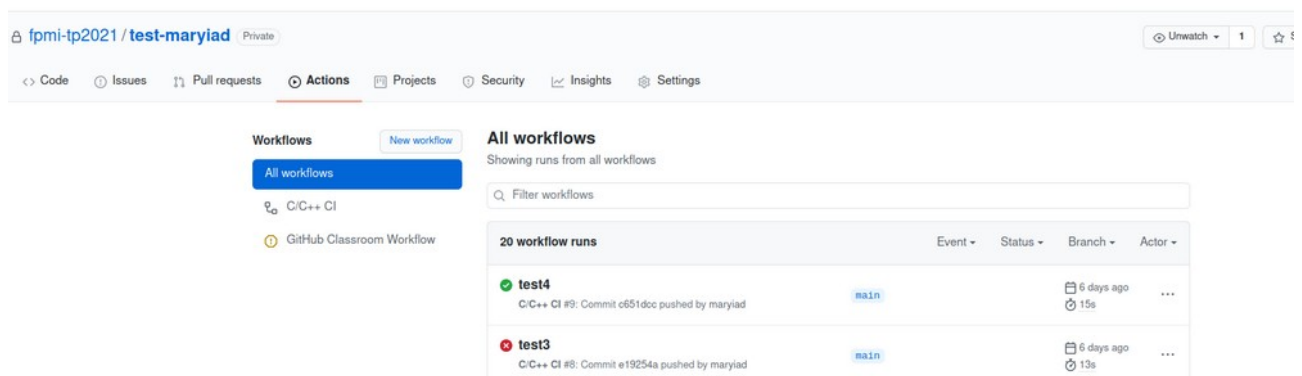
Сборка проекта с Github Action

GitHub Actions, релиз которого состоялся 13 ноября 2019 года, позволяет легко автоматизировать все рабочие процессы в области сборки и тестирования программного обеспечения.

GitHub Actions обеспечивает ряд преимуществ:

1. Предоставляет возможность реагировать на события внутри GitHub, избавляя тем самым от необходимости привлечения внешнего инструмента и пересылки данных через другой сервис.
2. Предоставляет более дешевый тариф, чем отдельный сервис для CI/CD.
3. Позволяет повторно задействовать общие рабочие процессы.

В интерфейсе репозитория на github на вкладке Actions создаются и настраиваются все процессы и этапы сборки (см. рис. ниже).

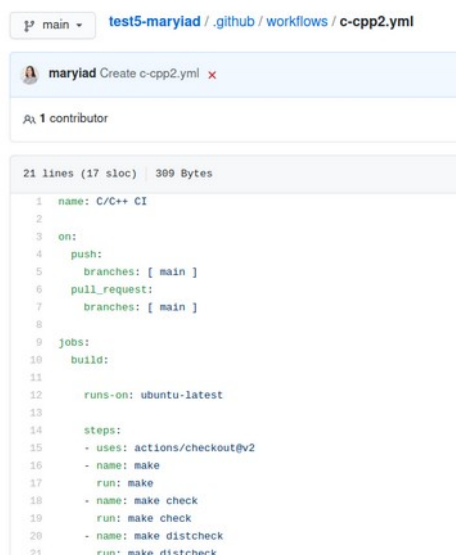


На приведенном рисунке демонстрируются две конфигурации сборки C/C++ + CI (активная) и Github Classroom Workflow (на паузе). В правой части отображаются

результаты процедуры сборки.

Для того, чтобы выполнялась сборка, в репозиторий добавлен скрытый каталог `.github`, содержащий каталог `workflows` с файлами конфигураций сборки.

Пример файла, описывающего процедуру сборки с применением github actions:



```
1 name: C/C++ CI
2
3 on:
4   push:
5     branches: [ main ]
6   pull_request:
7     branches: [ main ]
8
9 jobs:
10  build:
11
12    runs-on: ubuntu-latest
13
14    steps:
15      - uses: actions/checkout@v2
16      - name: make
17        run: make
18      - name: make check
19        run: make check
20      - name: make distcheck
21        run: make distcheck
```

Для того, чтобы сборка выполнялась корректно, Makefile проекта должен содержать следующие цели:

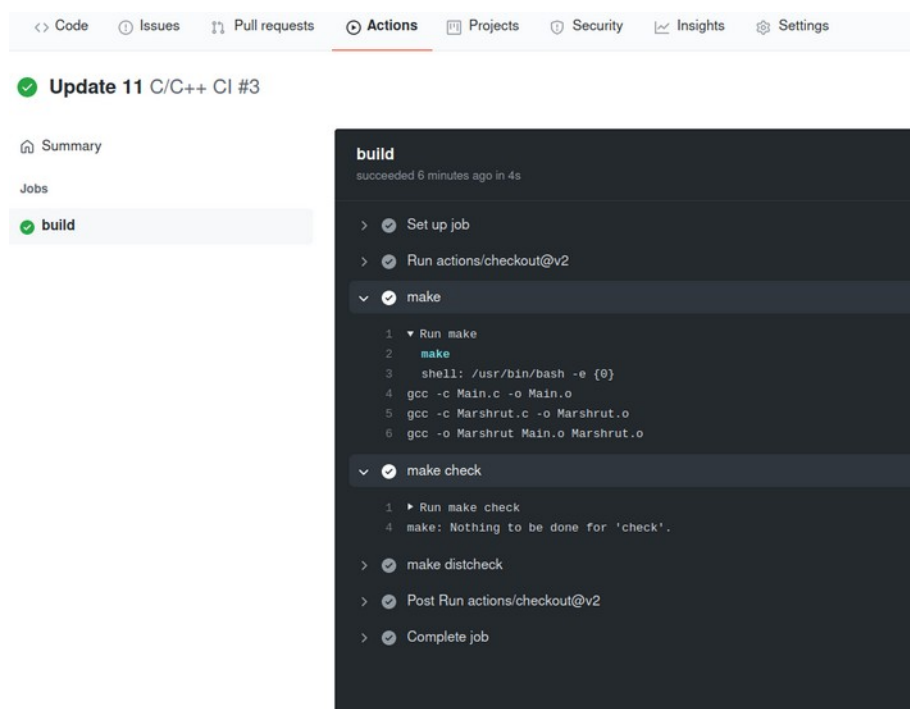
check:

```
shell: /usr/bin/bash -e {0}
```

distcheck:

```
shell: /usr/bin/bash -e {0}
```

Результат сборки представлен на рисунке ниже:



The screenshot shows the GitHub Actions interface for a workflow named 'Update 11 C/C++ CI #3'. The workflow is in a 'Completed' state. The 'Summary' tab is selected, showing a list of jobs. The 'build' job is highlighted, and its details are shown in a modal window. The job 'build' succeeded 6 minutes ago in 4s. The steps listed are: 'Set up job', 'Run actions/checkout@v2', 'make', 'make check', 'make distcheck', 'Post Run actions/checkout@v2', and 'Complete job'. The 'make' step is expanded, showing the following commands:

```
1 Run make
2 make
3 shell: /usr/bin/bash -e {0}
4 gcc -c Main.c -o Main.o
5 gcc -c Marshrut.c -o Marshrut.o
6 gcc -o Marshrut Main.o Marshrut.o
```

Подробнее в руководстве [Как пользоваться Github Actions](#).

Дополнительные материалы по сборке

Дополнительные материалы о применении утилиты make — <https://www.opennet.ru/docs/RUS/zlp/002.html>, <https://m.habrahabr.ru/post/211751/>.

ЗАДАНИЯ

Требования к репозиториям

Для текущей и последующих лабораторных работ используется модель управления репозиторием в соответствии с [A simple git branching model](#).

1. Для каждой задачи/новой функциональности лабораторной работы создаётся новая ветка.
2. После завершения работы ветка сливается в master.

Подробнее — см. [A simple git branching model](#).

Аналогичные соглашения требуется соблюдать и в последующих лабораторных работах.

Требования к отчёту

В файле Readme проекта на github должна быть ссылка на отчёт. Отчет опубликовать во внешнем хранилище, добавив ссылку на него в файл Readme репозитория. Отчёт должен результаты тестов по каждому заданию и ответы на контрольные вопросы.

Каждая лабораторная работа содержит тексты задач и контрольные вопросы, ответы на которые проверяются преподавателем при приёме работы у студента.

Выполнение студентом лабораторной работы и сдача её результатов преподавателю происходит следующим образом.

1. Студент выполняет разработку программ.

В ходе разработки студент обязан следовать указаниям к данной задаче (в случае их наличия). Исходные тексты программ следует разрабатывать в соответствии с требованиями к оформлению, приведёнными в приложении.

2. Студент выполняет самостоятельную проверку исходного текста каждой разработанной программы и правильности её работы, а также свои знания по теме лабораторной работы.

Исходные тексты программ должны соответствовать требованиям к оформлению, приведённым в приложении. Недопустимо отсутствие в тексте программы следующих важных элементов оформления: спецификации программного файла и подпрограмм, а также отступов, показывающих структурную вложенность языковых конструкций.

Для проверки правильности работы программы студенту необходимо разработать набор тестов и на их основе провести тестирование программы. Тестовый набор должен включать примеры входных и выходных данных, приведённые в тексте

задачи, а также тесты, разработанные студентом самостоятельно.

Самостоятельная проверка знаний по теме лабораторной работы выполняется с помощью контрольных вопросов и заданий, приведённых в конце текста лабораторной работы.

3. Студент защищает разработанные программы. Защита заключается в том, что студент должен ответить на вопросы преподавателя, касающиеся разработанной программы, и контрольные вопросы.

К защите необходимо представить исходные тексты программ, оформленных в соответствии с требованиями, и протоколы тестирования каждой программы, подтверждающие правильность ее работы.

Протокол тестирования включает в себя тест (описание входных данных и соответствующих им выходных данных), описание выходных данных, полученных при запуске программы на данном тесте, и отметку о прохождении теста. Тест считается пройденным, если действительные результаты работы программы совпали с ожидаемыми.

Пример оформления протокола тестирования программы на определение количества вхождений слов в строке представлен в таблице 1.

Программы, не прошедшие тестирование, к защите не принимаются. В случае неверной работы программы хотя бы на одном тесте студент обязан выполнить отладку программы для поиска и устранения ошибки.

Таблица 1. Пример протокола тестирования задачи на определение количества вхождений слова в строке.

№ п/п	Входные данные	Ожидаемые выходные данные	Действительные выходные данные	Тест пройден
1	foo bar foo bar	bar: 2 foo: 2	bar: 2 foo: 2	Да
2	test record created	test: 1 record: 1 created: 1	test: 1 record: 1 created: 1	Да
3	1 2 3 4 5 1 2 3 4 5 6 7 8 9	1: 2 2: 2 3: 2 4: 2 5: 2	1: 2 2: 2 3: 2 4: 2 5: 2	Да
4	Hello World	Hello: 1 World: 1	Hello: 1 World: 1	Да

Критерии оценивания

Оценка 4

Выполнено **Задание 2, 3**. Отчёт содержит описание задачи, структуру проекта и результаты тестов. Отчёт должен быть опубликован в репозитории или должна быть ссылка на него в файле Readme. Отчёт и исходный код проекта может содержать ошибки. Максимальная задержка на предоставление отчёта, исходного кода проекта, включая публикацию в репозитории и задании курса, а также его защиту — 2 недели.

Оценка 5-6

Выполнено **Задание 1-3**. Отчёт содержит описание задач, структуру проектов и результаты тестов. Отчёт должен быть опубликован в репозитории или должна быть ссылка на него в файле Readme. Отчёт и исходный код проектов может содержать ошибки. Максимальная задержка на предоставление отчёта, исходного кода проекта, включая публикацию в репозитории и задании курса, а также его защиту — 1 неделя.

Оценка 7-8

Выполнено **Задание 1-4**. Отчёт содержит описание задач, структуру проектов и результаты тестов. Отчёт должен быть опубликован в репозитории или должна быть ссылка на него в файле Readme. Отчёт, исходный код и ответы на контрольные вопросы могут содержать незначительные ошибки. Максимальная задержка на предоставление отчёта, исходного кода проекта, включая публикацию в репозитории и задании курса, а также его защиту — 1 неделя.

Оценка 9

Выполнено **Задание 1-4**. Отчёт содержит описание задач, структуру проектов и результаты тестов, а так же ответы на контрольные вопросы. Отчёт должен быть опубликован в репозитории или должна быть ссылка на него в файле Readme. Лабораторная работа опубликована в репозитории и в задании курса в срок, а так же защищена в срок. Не содержит ошибок.

ЗАДАНИЕ 1. КОНСОЛЬНЫЕ ПРИЛОЖЕНИЯ В REPL.IT

Для решения задания 1 необходимо использовать:

- шаблон репозитория проекта <https://github.com/mariyad/project-forlab3-tp>
- онлайн среду разработки <https://repl.it/>, в которой авторизоваться с учетной записью на github.

Порядок выполнения задания:

1. Познакомьтесь с руководством [Руководство по созданию проекта в Github и Repl.it](#) по созданию репозитория и активации аккаунта для сервиса repl.it.
2. Создайте репозиторий из шаблона репозитория согласно рекомендациям руководства из п. 1 текущего задания. В качестве имени репозитория введите имя согласно шаблону lab3-task1-gr13-petrov, указав корректно номер своей группы и фамилию вместо petrov.
3. Импортируйте проект в repl.it согласно рекомендациям из руководства в п. 1 текущего задания. Далее все изменения выполняются в среде repl.it и публикуются на github.
4. Создать ветку в репозитории для данной задачи согласно Требования к репозиториям, например development, и переключиться в неё. **Все изменения по проекту выполнять в новой ветке!**

5. Отредактируйте файл Readme в созданном проекте, изменив текст в разделах Overview, Usage, Building. Добавьте в Readme свою фамилию, имя и номер подгруппы вместо текста *{add your last name, first name and group number}*. Закоммитить изменения.
6. Переименовать файл `hello.c`, например в `main.c`.
7. Отредактировать файл `.replit` и указать название файла, который компилируется, например `main.c`, и имя бинарного файла, например `labrabota3-1`, а так же путь к этому файлу. Т.е. отредактировать строку `run = "gcc -o hello /home/runner/project2/src/hello.c"` к виду, например, `"gcc -o labrabota3-1 /home/runner/lab3-task1-gr13-petrov/src/main.c"`, где `labrabota3-1` — это исполняемый файл, `/home/runner/lab3-task1-gr13-petrov/src/main.c` — пусть к исходному файлу с учётом, что `/home/runner/` — домашний каталог, `lab3-task1-gr13-petrov/` — каталог проекта из Вашего репозитория, совпадающий с названием репозитория. Проверить путь к текущему каталогу можно командой `pwd` на вкладке Shell в `repl.it` (см. п. 16 в руководстве из пункта 1).
8. Отредактировать `.gitignore` и указать имя бинарного файла, который должен получиться в результате `labrabota3-1` и который не должен быть опубликован в репозиторий, т. е. Проигнорирован при публикации изменений.
9. Закоммитить изменения и опубликовать их в репозиторий на github.
10. Изучить документацию Стандарты кодирования для оформления кода и при разработке программы применить правила оформления кода согласно стандартам.
11. Коммитить изменения в репозиторий пошагово, например добавили новую функцию, проверили, коммитим.
12. В каждом задании указано, какую подзадачу должна решать разрабатываемая функция. Исходные данные программа получает через аргументы командной строки. Если в задании необходимо в качестве исходных данных использовать массивы, их нужно сформировать с помощью генератора псевдослучайных чисел (размерности массивов программа получает через аргументы командной строки). На экран вывести исходные данные и полученные результаты.
13. Провести тесты (контрольный расчет) для проверки работоспособности алгоритма решения задачи и включить их в отчёт.
14. После завершения работы над заданием опубликовать изменения в репозиторий, т. е. выполнить `git push`.
15. Переключиться в репозиторий на github и, используя `pull request`, слить изменения из ветки `development` в ветку `master` (см. [Как сделать pull request на Github](#)).

Дополнительные материалы:

Познакомьтесь с кратким руководством по созданию простой программы и компиляции в `repl.it`:

- <https://www.youtube.com/watch?v=1CZBf1y4IvA>;
- <https://docs.repl.it/tutorials/01-introduction-to-the-repl-it-ide>.

ЗАДАНИЕ 2. ИЗУЧАЕМ КУРС GitHub Actions: Hello World.

1. Познакомьтесь с описанием функционала в Сборка проекта с Github Action и в руководстве [Как пользоваться Github Actions](#).
2. Откройте курс «Hello GitHub Actions» по ссылке <https://github.com/skills/hello-github-actions>. Курс опубликован на <https://skills.github.com>.
3. Подпишитесь на курс и в процессе подписки создайте публичный репозиторий с

именем tmp-lab3-task2.

4. Изучите курс, материалы которого помогут использовать Github Actions в заданиях 3 и 4.

ЗАДАНИЕ 3. ИСПОЛЬЗУЕМ СТРУКТУРЫ ДЛЯ ПРЕДСТАВЛЕНИЯ ДАННЫХ

Repl.it не использовать для выполнения данного задания!

1. Изучить материалы по структурам данных на языке C:
 - <https://prog-cpp.ru/c-struct/>
 - <https://metanit.com/c/tutorial/6.1.php>
 - <https://younglinux.info/c/structure>
2. Создайте репозиторий, используя ссылку репозитория для **задания 3** лабораторной работы 3 с сайта <https://edufpmi.bsu.by> для Вашей группы.
3. Файл README.md должен содержать разделы Overview, Usage, Building, Additional Notes. В Additional Notes добавить ссылки на репозитории из задания 1, 2.
4. При работе над заданием следуйте правилам создания веток, изложенным в документации Требования к репозиториям.
5. Изучите документацию Сборка приложений по сборке проектов с помощью утилиты make. Создайте файл Makefile.
6. Используя редактор nano, напишите программу с использованием структуры согласно варианту. Программа должна состоять из нескольких файлов с разделением полномочий на клиенты, серверы и интерфейсы согласно модели КИС — 2. Организация программы согласно модели КИС. Все файлы находятся в одном каталоге, например src.
7. Требуется следовать стандарту оформления кода для языка C и файла сборки Makefile с помощью утилиты make.
8. Выполнить сборку с помощью make и Github Actions согласно рекомендациям из Сборка проекта с Github Action и руководства [Как пользоваться Github Actions](#). Одним из шагов для запуска рабочего процесса должен быть запуск утилиты make.
9. Провести тесты (контрольный расчет) для проверки работоспособности алгоритма и включить их в отчет.

ЗАДАНИЕ 4. ЗАПИСЬ И ЧТЕНИЕ ТЕКСТОВЫХ ФАЙЛОВ

Repl.it не использовать для выполнения данного задания!

1. Создайте репозиторий, используя ссылку репозитория для **задания 4** лабораторной работы 3 с сайта <https://edufpmi.bsu.by> для Вашей группы.
2. Файл README.md должен содержать разделы Overview, Usage, Building, Additional Notes. В Additional Notes добавить ссылки на репозитории из задания 1, 2, 3.
3. Внести в них изменения с учетом использования файловой структуры проекта согласно модели КИС.
4. При работе над заданием следуйте правилам создания веток, изложенным в документации Требования к репозиториям.
5. Создайте структуру проекта согласно Структура проекта согласно модели КИС и правил сборки.
6. Используя редактор nano, напишите программу для обработки текстовых файлов. В ходе выполнения требуется:
 - Создать текстовый файл с произвольной информацией.
 - Организовать просмотр содержимого файла.

- Организовать чтение и обработку данных из файла в соответствии с индивидуальным заданием.
 - Сохранить полученные результаты в новый текстовый файл.
3. Программа должна состоять из нескольких файлов и иметь структуру согласно модели КИС.
 4. Требуется следовать стандарту оформления кода для языка C и файлов сборки с помощью утилиты make.
 5. Выполнить сборку с помощью make и Github Actions согласно рекомендациям из Сборка проекта с Github Action и руководства Как пользоваться Github Actions.
 6. Провести тесты (контрольный расчет) для проверки работоспособности алгоритма и включить их в отчет.

Варианты

Вариант 1

1. Решить задачу, используя функцию.

Составить программу для нахождения общего числа некоторой буквы (буква вводится с клавиатуры) в нескольких предложениях (предложение задаётся строкой в языке Си). Функция считает количество появлений заданной буквы в строке.

3. Описать структуру с именем GROUP, содержащую поля: Name – фамилия и инициалы, DAT – дата рождения (год, месяц, число), SES – успеваемость (массив из трех элементов).

Написать программу, выполняющую:

- ввод с клавиатуры данных в массив GR5, состоящий из 10 структур типа GROUP;
- вывод на экран записей, упорядоченных по возрастанию поля SES;
- вывод списка студентов, возраст которых на 01.12.2010 года не превышает 20 лет;
- если таких студентов нет – выдать сообщение.

4. **«Человек»:** фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц, число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира).

Вывести сведения о самом молодом человеке.

Вариант 2

1. Решить задачу, используя функцию.

Дан массив натуральных чисел размерности n. Найти число в массиве, имеющее максимальную сумму цифр в десятичном представлении. Если существует несколько чисел с одинаковой максимальной суммой, вывести на печать их все. Функция должна возвращать сумму цифр числа в десятичном представлении.

3. Описать структуру с именем STUDENT, содержащую поля: Name – фамилия и инициалы, Kurs – курс, SES – успеваемость (массив из пяти элементов).

Написать программу, выполняющую:

- ввод с клавиатуры данных в массив STUD, состоящий из 10 структур типа STUDENT, записи должны быть упорядочены по алфавиту;
- вывод на экран записей, упорядоченного списка студентов, средний бал которых

- превышает общий средний бал;
- если таких студентов нет – выдать сообщение.

4. **«Автомобиль»:** марка; цвет; серийный номер; регистрационный номер; год выпуска; год техосмотра; цена.

Вывести данные про автомобили, которым больше 2 лет.

Вариант 3

1. Решить задачу, используя функцию.

Два простых числа называются «близнецами», если они отличаются друг от друга на 2 (например, 41 и 43). Напечатать все пары «близнецов», не превышающих заданного значения **a**. Функция должна проверять, является ли целое число простым или нет (простым является число, если оно делится без остатка только на себя и на 1).

3. Описать структуру с именем STUD, содержащую поля: Name – фамилия и инициалы, GROUP – название группы (факультет, курс, номер группы), SES – успеваемость (массив из четырех элементов).

Написать программу, выполняющую:

- ввод с клавиатуры данных в массив STUD1, состоящий из 10 структур типа STUD, записи должны быть упорядочены по алфавиту;
- вывод на экран данных о студентах, включенных в массив, средний балл которых превышает 4,2. Список упорядочить по возрастанию среднего балла. Сохранить информацию о положении студента в исходном списке;
- если таких студентов нет – выдать сообщение.

4. **«Государство»:** название страны; столица; государственный язык; население; площадь территории; денежная единица; государственный строй; глава государства.

Вывести данные про государства, население которых больше 20 млн жителей.

Вариант 4

1. Решить задачу, используя функцию.

Дан массив натуральных чисел размерности **n**. Вывести на печать все числа этого массива, являющиеся степенями пяти. Функция должна проверять, является ли число степенью пяти или нет.

3. Описать структуру с именем NOTE, содержащую поля: Name – фамилия и инициалы, TELE – номер телефона, DATE – дата рождения (год, месяц, число).

Написать программу, выполняющую:

- ввод с клавиатуры данных в массив BLOCKNOTE, состоящий из 10 структур типа NOTE, записи должны быть упорядочены по возрастанию даты рождения;
- вывод на экран сведений о человеке, номер телефона которого введен с клавиатуры;
- если такого человека нет – выдать сообщение.

4. **«Абитуриент»:** фамилия; имя; отчество; пол; национальность; дата рождения (год, месяц, число); домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); оценки по ЦТ; проходной балл.

Вывести данные про абитуриентов, проходной балл которых равен больше 225 .

Вариант 5

1. Решить задачу, используя функцию.

Дан массив натуральных чисел размерности n . Вывести на печать все числа этого массива, являющиеся полными квадратами. Функция должна проверять, является ли число полным квадратом или нет.

3. Описать структуру с именем NOTE1, содержащую поля: Name – фамилия и инициалы, TELE – номер телефона, DATE – дата рождения (год, месяц, число).

Написать программу, выполняющую:

- ввод с клавиатуры данных в массив BLOCK, состоящий из 9 элементов типа NOTE1, записи должны быть упорядочены по инициалам;
- вывод на экран информации о людях, чьи дни рождения приходятся на месяц, значение которого введено с клавиатуры; если такого человека нет – выдать сообщение.

4. **«Рабочий»:** фамилия; имя; отчество; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); национальность; дата рождения (год, месяц, число); № цеха; табельный номер; образование; год поступления на работу.

Вывести данные про рабочих, поступивших на работу в 2020 году.

Вариант 6

1. Решить задачу, используя функцию.

Дан массив натуральных чисел размерности n . Отсортировать его в порядке неубывания количества единиц в двоичной записи чисел. Функция должна считать количество единиц в двоичной записи числа.

3. Описать структуру с именем NOTE2, содержащую поля: Name – фамилия и инициалы, TELE – номер телефона, DATE – дата рождения (год, месяц, число).

Написать программу, выполняющую:

- ввод с клавиатуры данных в массив BLOCK2, состоящий из 7 элементов типа NOTE1, записи должны быть упорядочены по первым трем цифрам номера телефона;
- вывод на экран информации о человеке, чья фамилия введена с клавиатуры;
- если такого нет – выдать сообщение.

4. **«Военнослужащий»:** фамилия; имя; отчество; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); национальность; дата рождения (год, месяц, число); должность; звание.

Вывести данные про военнослужащих в звании “лейтенант”.

Вариант 7

1. Решить задачу, используя функцию.

Дата некоторого дня характеризуется тремя натуральными числами (число, месяц, год). Определите количество дней, прошедших между двумя датами (учитывать високосные годы, даты начинаются с 1970 г.). Функция должна считать, сколько дней прошло с 1 января 1970 г. до текущей даты.

3. Описать структуру с именем PERSON, содержащую поля: Name – фамилия и инициалы, FAC – факультет, GROUP – группа, DATE – дата поступления в ВУЗ (год, месяц, число).

Написать программу, выполняющую:

- ввод с клавиатуры данных в массив VUZ, состоящий из 10 элементов типа PERSON, записи должны быть упорядочены по дате поступления в ВУЗ;
- вывод на экран информации о студентах, упорядоченной по факультетам, группам, дате поступления. В каждой группе фамилии должны быть расположены в алфавитном порядке.

4. **«Владелец телефона»:** фамилия; имя; отчество; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); № телефона.

Вывести данные про владельцев телефона номер, которого начинается на 621.

Вариант 8

1. Решить задачу, используя функцию.

Дан массив натуральных чисел размерности n . Отсортировать массив в порядке неубывания сумм цифр десятичной записи чисел. Функция должна считать сумму десятичных цифр натурального числа.

3. Описать структуру с именем ZNAK, содержащую поля: Name – фамилия и имя, ZOD – знак зодиака, DATE – дата рождения (массив из трех чисел: год, месяц, число).

Написать программу, выполняющую:

- ввод с клавиатуры данных в массив MASS, состоящий из 10 элементов типа ZNAK, записи должны быть упорядочены по дате дня рождения;
- вывод на экран информации о людях, родившихся под знаком зодиака, наименование которых вводится с клавиатуры;
- если такого нет – выдать сообщение.

4. **«Владелец автомобиля»:** фамилия; имя; отчество; номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира) марка автомобиля; номер автомобиля; номер техпаспорта.

Вывести данные про автомобили марки "Ваз".

Вариант 9

1. В заданной строке после каждого символа вставить число, соответствующее коду этого символа.

3. Описать структуру с именем MARSHRUT, содержащую поля:

номер маршрута; начальный пункт маршрута; конечный пункт маршрута; длина маршрута.

Написать функции:

- создания массива не более чем из 10 записей (структур) сведений о маршрутах (ввод данных с клавиатуры);
- определения маршрута с максимальной длиной;
- расположения записей по возрастанию номеров маршрутов;
- вывода сведений о маршрутах, которые начинаются или заканчиваются в пункте, название которого вводится с клавиатуры.

Все необходимые данные для функций должны передаваться в качестве их параметров. Использование глобальных параметров не допускается. Создать проект, который демонстрирует работу всех функций.

4. **«Пациент»:** фамилия; имя; отчество; пол; национальность; рост; вес; дата

рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); номер больницы; отделение; номер медицинской карты; диагноз; группа крови.

Вывести данные про пациентов с 18 отделения.

Вариант 10

1. Решить задачу, используя функцию.

Получить все шестизначные «счастливые» номера и подсчитать их количество. Функция должна проверять, является ли число «счастливым» или нет. «Счастливым» называется такое шестизначное число, у которого сумма первых трех цифр равна сумме последних трех.

3. Описать структуру с именем STUDENT, содержащую поля:

фамилия и инициалы студента; номер группы; успеваемость (массив из четырех оценок на экзаменах в 5-бальной системе).

Написать функции:

- создания массива 7 записей (структур) данных о студентах (ввод данных с клавиатуры);
- вычисления среднего бала каждого студента;
- расположения записей по убыванию среднего бала;
- вывода сведений о студентах, имеющих оценки только 4 и 5; удаления из списка студента с минимальным средним балом.

Все необходимые данные для функций должны передаваться в качестве их параметров. Использование глобальных параметров не допускается. Создать проект, который демонстрирует работу всех функций.

4. «Покупатель»: *фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); номер кредитной карточки; банковского счета.*

Вывести данные о покупателях города Брест.

Вариант 11

1. Решить задачу, используя функцию.

Дан массив натуральных чисел размерности n . Определить количество чисел-палиндромов в нем. Палиндром («перевертыш») - число, десятичная запись которого читается одинаково слева направо и справа налево (например, 14541). Функция должна проверять, является ли число палиндромом или нет.

3. Описать структуру с именем TOVAR, содержащую поля:

название товара; количество единиц товара; стоимость товара; дата поступления товара в виде структуры (год, месяц, день).

Написать функции:

- создания массива SPISOK не более чем из 10 записей (структур) данных о товарах (ввод данных с клавиатуры);
- вычисления средней стоимости товара;
- расположения записей по возрастанию стоимости товаров;
- вывода сведений о товарах, поступивших более 10 месяцев назад.

Все необходимые данные для функций должны передаваться в качестве их параметров. Использование глобальных параметров не допускается. Создать проект,

который демонстрирует работу всех функций.

4. **«Студент»:** фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); ВУЗ; курс; группа; средний бал; специальность.

Вывести сведения про всех студентов у которых средний балл ниже 7.0 баллов.

Вариант 12

1. Решить задачу, используя функцию.

Дан массив натуральных чисел размерности n . Для каждого числа переставьте его десятичные числа так, чтобы получить максимально возможное число, записанное теми же цифрами. Функция должна переставлять цифры одного натурального числа,

3. Описать структуру с именем MARSHRUT1, содержащую поля:

номер маршрута; начальный пункт маршрута; конечный пункт маршрута; длина маршрута.

Написать функции:

- создания массива не более чем из 10 записей (структур) сведений о маршрутах (ввод данных с клавиатуры);
- определения маршрута с минимальной длиной;
- расположения записей по возрастанию длины маршрута;
- вывода сведений о маршрутах, которые начинаются или заканчиваются в пункте, название которого вводится с клавиатуры.

Все необходимые данные для функций должны передаваться в качестве их параметров. Использование глобальных параметров не допускается. Создать проект, который демонстрирует работу всех функций.

4. **«Школьник»:** фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); школа; класс.

Вывести сведения про всех учеников пятых классов.

Вариант 13

1. Решить задачу, используя функцию.

Дан массив из n натуральных чисел. Определить количество чисел, десятичная запись которых не содержит одинаковых цифр. Функция должна проверять, содержит ли десятичная запись числа одинаковые цифры или нет.

3. Описать структуру с именем ABON, содержащую поля:

фамилия и инициалы абонента; номер телефона; дата подключения телефона в виде структуры (год, месяц, день); начисленная сумма оплаты; сумма на счёту абонента.

Написать функции:

- создания массива не более чем из 12 записей (структур) данных об абонентах (ввод данных с клавиатуры);
- расположения записей по алфавиту (с учетом инициалов для абонентов с одинаковыми фамилиями);
- добавить 20 р. на счета абонентов, которых подключили более 10 лет назад;
- вывода сведений об абонентах, у которых сумма на счёту отрицательная после

вычета начислений;

- вывода сведений об абоненте, номер телефона которого вводится с клавиатуры.

Все необходимые данные для функций должны передаваться в качестве их параметров. Использование глобальных параметров не допускается. Создать проект, который демонстрирует работу всех функций.

4. «**Жанры кино**»: *название; режиссер (фамилия; имя); год выхода; страна; стоимость; доход; прибыль, жанр*. Вывести данные о детских фильмах, выпущенных после 2018 года.

Вариант 14

1. Решить задачу, используя функцию.

Даны два предложения, в которых имеются буквы **ш**. Найти, в каком из них эта буква имеет больший порядковый номер (при счёте от начала предложения). Если в предложении имеется несколько букв **ш**, то должна быть учтена последняя из них. (Определить функцию для нахождения порядкового номера буквы последнего вхождения в предложение некоторой буквы.)

3. Описать структуру с именем *WORKER*, содержащую поля:

фамилия, имя, отчество, должность, дата рождения, пол, дата приёма на работу.

Написать функции:

- создания массива не менее чем из 10 записей (структур) данных об сотрудниках (ввод данных с клавиатуры);
- определения количества сотрудников пенсионного возраста (мужчинам больше 65-ти лет, женщинам — 60);
- вывода результатов сортировки по полу и по алфавиту;
- вывода сведений о самом молодом сотруднике.
- вывода количества сотрудников каждого пола.

Все необходимые данные для функций должны передаваться в качестве их параметров. Использование глобальных параметров не допускается. Создать проект, который демонстрирует работу всех функций.

4. «**Товар**»: *наименование; стоимость; срок хранения; сорт; дата выпуска; срок годности.*

Вывести данные про товары срок годности которых истекает в этом году.

Вариант 15

1. Решить задачу, используя функцию.

В каждом из двух классов учатся по 18 человек. Известны средние оценки каждого ученика каждого класса, подсчитанные по ряду предметов (все значения для каждого класса разные). Определить, в каком классе у "третьего из самых успевающих учеников" средняя оценка больше.

3. Описать структуру с именем *ABON1*, содержащую поля:

фамилия и инициалы абонента; номер телефона; дата подключения телефона в виде структуры (год, месяц, день); тарифный план, начисленная сумма оплаты; сумма на счёту абонента.

Написать функции:

- создания массива не более чем из 12 записей (структур) данных об абонентах (ввод данных с клавиатуры);

- расположения записей по алфавиту (с учетом инициалов для абонентов с одинаковыми фамилиями);
- добавить 10 р. на счета абонентов, которых подключили более 10 лет назад;
- вывода сведений об абонентах, у которых сумма на счету отрицательная после вычета начислений;
- вывода сведений по количеству абонентов для каждого тарифного плана.

Все необходимые данные для функций должны передаваться в качестве их параметров. Использование глобальных параметров не допускается. Создать проект, который демонстрирует работу всех функций.

4. «**Рейс**»: марка автомобиля; номер автомобиля; пункт назначения; грузоподъёмность (в тоннах); стоимость единицы груза; общая стоимость груза.

Вывести данные про автомобили, грузоподъёмность которых больше 2 тонн.

Вариант 16

1. Решить задачу, используя функцию.

Составить программу для нахождения общего количества заданной буквы в трёх заданных предложениях. (Определить функцию для расчёта количества некоторой буквы в предложении.)

3. Описать структуру с именем *PRODUCT*, содержащую поля:

наименование, цена, дата производства, срок годности, количество, производитель.

Написать функции:

- создания массива продуктов не менее 12 записей (структур) сведений о продуктах (ввод с клавиатуры);
- вывести сведения с сортировкой по дате производства по возрастанию;
- вывести сведения о товарах, срок годности которых истекает через двое суток;
- вывести сведения о количестве товаров для каждого производителя;
- вывести сведения о товаре с максимальной ценой, срок годности которого не истек.

Все необходимые данные для функций должны передаваться в качестве их параметров. Использование глобальных параметров не допускается. Создать проект, который демонстрирует работу всех функций.

4. «**Книга**»: название; автор (фамилия; имя); год выхода; издательство; себестоимость; цена; прибыль.

Вывести данные про книги авторов, фамилия которых начинается с буквы "К".

Вариант 17

1. Решить задачу, используя функцию.

В каждом из двух классов учатся по 23 человека. Известны значения роста каждого ученика этих классов. Определить, в каком классе "третий из самых высоких учеников" выше.

3. Описать структуру с именем *FOOTBALL*, содержащую поля:

фамилия, дата рождения, амплуа, количество игр, количество забитых мячей, место рождения.

Написать функции:

- создания массива не более чем из 12 записей (структур) данных о футболистах (ввод данных с клавиатуры);
- вывести сведения о футболистах, старших 20-ти лет и забивающих за игру не

менее 0,4 мяча;

- вывести сведения о футболистах, место рождения которых Брест или Минск;
- вывести сведения о самом молодом футболисте;
- вывести сведения о футболисте, у кого ближайший день рождения к текущей дате.

Все необходимые данные для функций должны передаваться в качестве их параметров. Использование глобальных параметров не допускается. Создать проект, который демонстрирует работу всех функций.

4. **«Здание»:** *адрес; тип здания; количество этажей; количество квартир; срок эксплуатации; срок до капитального ремонта (25 лет - срок эксплуатации).*

Вывести данные про здания срок эксплуатации, которых больше 50 лет.

Вариант 18

1. Решить задачу, используя функцию.

Дата некоторого дня характеризуется тремя натуральными числами: *g* (год), *m* (порядковый номер месяца) и *n* (число). По заданным *g*, *n* и *m* определить:

- а) дату предыдущего дня;
- б) дату следующего дня.

Определить функцию, подсчитывающую количество дней в том или ином месяце.

Рассмотреть случай, когда заданный год не является високосным;

2) заданный год может быть високосным.

3. Описать структуру с именем *CINEMA*, содержащую поля:

название фильма, дата и время сеанса, продолжительность сеанса, жанр, бюджет.

Написать функции:

- создания массива не менее чем из 12 записей (структур) данных о фильмах (ввод данных с клавиатуры);
- вывести сведения о фильмах с максимальной и минимальной продолжительностью;
- вывести данные о фильмах, начинающихся до 18:00 и продолжительностью сеанса менее 1 часа 30 минут;
- вывести сведения о фильмах жанра «Комедия» и с максимальным бюджетом;
- вывести сведения о фильмах с сортировкой по жанру и по бюджету в рамках жанра.

Все необходимые данные для функций должны передаваться в качестве их параметров. Использование глобальных параметров не допускается. Создать проект, который демонстрирует работу всех функций.

4. **«Учёный»:** *фамилия; имя; отчество; пол; национальность; дата рождения (год, месяц, число); учёная степень, должность, номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира).*

Вывести сведения об учёных кандидатах технических наук.

Вариант 19

1. Решить задачу, используя функцию.

Дата некоторого дня характеризуется тремя натуральными числами: *g* (год), *m* (порядковый номер месяца) и *n* (число). По заданным *g*, *n* и *m* определить:

- а) дату предыдущего дня;
- б) дату следующего дня.

Определить функцию, подсчитывающую количество дней в том или ином месяце.

Рассмотреть случай, когда заданный год является високосным.

3. Описать структуру с именем *WORKER2*, содержащую поля:

код работника, фамилия работника, должность, пол, дата подписания контракта, срок действия контракта, оклад.

Написать функции:

- создания массива не более чем из 12 записей (структур) данных о работниках (ввод данных с клавиатуры);
- вывести сведения о работниках, подписавших контракт менее года назад;
- вывести сведения о работниках, с которыми дважды заключали контракт;
- вывести средний срок действия контракта для определённой должности;
- вывести сведения о количестве работников мужского и женского пола.

Все необходимые данные для функций должны передаваться в качестве их параметров. Использование глобальных параметров не допускается. Создать проект, который демонстрирует работу всех функций.

4. «**Международная компания**»: *название; интернет сайт; адрес главного офиса (почтовый индекс, страна, область, район, город, улица, дом, квартира) продолжительность пребывания на мировом рынке; количество сотрудников; количество филиалов в Европе.*

Вывести международные компании, количество сотрудников у которых больше 10000.

Вариант 20

1. Решить задачу, используя функцию.

Даны два предложения. Найти общее количество букв н в них. (Определить функцию для расчета количества букв н в предложении.)

3. Описать структуру с именем *AIRPLANE*, содержащую поля:

№ Авиарейса, время вылета, время прилёта, направление, марка самолёта, расстояние.

Написать функции:

- создания массива не более чем из 12 записей (структур) данных об авиарейсах (ввод данных с клавиатуры);
- вывести данные об авиарейсе с максимальной длительностью полёта;
- вывести сведения обо всех авиарейсах по определённому направлению;
- вывести сведения среднее расстояние для всех авиарейсов;
- вывести количество авиарейсов для каждой марки самолёта.

Все необходимые данные для функций должны передаваться в качестве их параметров. Использование глобальных параметров не допускается. Создать проект, который демонстрирует работу всех функций.

4. «**Пенсионер**»: *фамилия; имя; отчество; пол; национальность; дата рождения (год, месяц, число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира).*

Вывести сведения про всех пенсионеров, которые на пенсии больше 5 лет.

Вариант 21

1. Решить задачу, используя функцию.

Известны марки и стоимость 7 моделей автомобилей. Определить марку автомобиля,

стоимость которого является "средней" (т. е. величина которой оказалась в середине массива в случае его сортировки).

3. Описать структуру с именем *BASKETBALL*, содержащую поля:

фамилия, дата рождения, клуб, амплуа, количество игр, количество забитых очков, количество фолов, место рождения.

Написать функции:

- создания массива не более чем из 12 записей (структур) данных о баскетболистах (ввод данных с клавиатуры);
- вывести сведения о баскетболистах, младше 20-ти лет и забивающих за игру в среднем более 18 очков;
- вывести сведения о баскетболистах, место рождения которых Витебск и не имеющих фолов;
- вывести сведения о самом молодом баскетболисте;
- вывести сведения о баскетболисте с максимальным количеством очков.

Все необходимые данные для функций должны передаваться в качестве их параметров. Использование глобальных параметров не допускается. Создать проект, который демонстрирует работу всех функций.

4. «**Зоопарк**»: *Название животного; количество вида; адрес зоопарка (почтовый индекс, страна, область, район, город, улица, дом, квартира); общее количество животных, количество работников.*

Вывести сведения про зоопарки, в которых есть уссурийские тигры.

Вариант 22

1. Решить задачу, используя функцию.

Найти все трехзначные простые числа. (Определить функцию, позволяющую распознавать простые числа.)

3. Описать структуру с именем *TRAIN*, содержащую поля:

№ поезда, направление, время прибытия, время отбытия, расстояние, количество вагонов, тип вагона, количество пассажиров в вагоне.

Написать функции:

- создания массива не более чем из 12 записей (структур) данных о поездах (ввод данных с клавиатуры);
- вывести данные о поездах, пребывающих в пути более суток;
- вывести суммарное количество пассажиров для поездов с купейными вагонами;
- вывести сведения обо всех поездах, следующих в Гродно;
- вывести сведения о поезде с максимальным количеством вагонов.

Все необходимые данные для функций должны передаваться в качестве их параметров. Использование глобальных параметров не допускается. Создать проект, который демонстрирует работу всех функций.

4. «**Программное обеспечение**»: *название; название компании производителя; год выхода; версия, период поддержки, цена.*

Вывести данные о программном обеспечении, которое дороже 200 рублей и период поддержки которого более 3 лет.

Вариант 23

1. Решить задачу, используя функцию.

Даны две последовательности целых чисел: a_1, a_2, \dots, a_8 и b_1, b_2, \dots, b_8 . Найти количество чётных чисел в первой из них и количество нечётных во второй.

(Определить функцию, позволяющую распознавать чётные числа.)

3. Описать структуру с именем *AIRPLANE2*, содержащую поля:

№ Авиарейса, время вылета, время прилёта, направление, марка самолёта, расстояние.

Написать функции:

- создания массива не менее чем из 12 записей (структур) данных об авиарейсах (ввод данных с клавиатуры);
- вывести данные об авиарейсе с максимальной скоростью;
- вывести сведения обо всех авиарейсах для определённой марки самолёта;
- вывести сведения об авиарейсе с минимальным расстоянием;
- вывести количество авиарейсов для каждого направления.

Все необходимые данные для функций должны передаваться в качестве их параметров. Использование глобальных параметров не допускается. Создать проект, который демонстрирует работу всех функций.

4. **«Область»:** *название области; областной центр; население (мужчины и женщины; мужчины; женщины); площадь территории; руководитель областной администрации.*

Вывести данные об областях Беларуси, население которых меньше 1,2 млн. жителей.

Вариант 24

1. Решить задачу, используя функцию.

Составить программу для удаления всех букв на четных позициях в нескольких предложениях (предложение задается строкой в языке Си). Функция считает количество удаленных букв в строке.

2. Описать структуру с именем *GROUP*, содержащую поля: *Name* – фамилия и инициалы, *DAT* – дата поступления (год, месяц, число), *SES* – успеваемость (массив из трех элементов).

Написать программу, выполняющую:

- ввод с клавиатуры данных в массив, состоящий из 8 структур типа *GROUP*;
- вывод на экран записей, упорядоченных по возрастанию поля *SES*;
- вывод списка студентов, возраст которых на 01.12.2020 года не превышает 20 лет;
- если таких студентов нет – выдать сообщение.

4. **«Автомобиль»:** *марка; цвет; серийный номер; регистрационный номер; год выпуска; год техосмотра; цена.*

Вывести данные об автомобилях марки *Audi*, которым больше 3 лет.

Вариант 25

1. Заданы две одинаковые по длине строки. Построить новую строку, в которой на четных местах расположены элементы первой строки, а на нечетных – элементы второй строки.

3. Описать структуру с именем *STUDENT*, содержащую поля: *Name* – фамилия и инициалы, *Kurs* – курс, *SES* – успеваемость (массив из пяти элементов).

Написать программу, выполняющую:

- ввод с клавиатуры данных в массив *STUD*, состоящий из 10 структур типа *STUDENT*, записи должны быть упорядочены по алфавиту;

- вывод на экран записей, упорядоченного списка студентов, средний бал которых превышает общий средний бал;
- если таких студентов нет – выдать сообщение.

4. **«Государство»:** название страны; столица; материк; государственный язык; население; площадь территории; денежная единица; государственный строй; глава государства.

Вывести данные о государствах Евразии, население которых меньше 20 млн жителей.

Вариант 26

1. Получить из заданной строки две строки, состоящие из символов первой строки, имеющих соответственно четные и нечетные индексы.

3. Описать структуру с именем NOTE1, содержащую поля: Name – фамилия и инициалы, TELE – номер телефона, DATE – дата рождения (год, месяц, число).

Написать программу, выполняющую:

- ввод с клавиатуры данных в массив BLOCK, состоящий из 9 элементов типа NOTE1, записи должны быть упорядочены по инициалам;
- вывод на экран информации о людях, чьи дни рождения приходятся на месяц, значение которого введено с клавиатуры; если такого человека нет – выдать сообщение.

4. **«Владелец телефона»:** фамилия; имя; отчество; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); № телефона, модель телефона.

Вывести данные о владельцах телефонов, у которых модель телефона Samsung S20 Ultra.

Вариант 27

1. В заданной строке расположить в обратном порядке все слова. Разделителями слов считаются пробелы.

3. Описать структуру с именем ZNAK, содержащую поля: Name – фамилия и имя, ZOD – знак зодиака, DATE – дата рождения (массив из трех чисел: год, месяц, число), пол.

Написать программу, выполняющую:

- ввод с клавиатуры данных в массив MASS, состоящий из 10 элементов типа ZNAK, записи должны быть упорядочены по дате дня рождения;
- вывод на экран информации о мужчинах, родившихся под знаком зодиака, наименование которого вводится с клавиатуры;
- если такого нет – выдать сообщение.

4. **«Рабочий»:** фамилия; имя; отчество; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); национальность; дата рождения (год, месяц, число); № цеха; табельный номер; образование; год поступления на работу.

Вывести данные о рабочих, имеющих среднее профессиональное образование.

Вариант 28

1. Задана строка. Получить предпоследнее слово этой строки. Разделителем слов считаются один или несколько пробелов.

3. Описать структуру с именем MARSHRUT, содержащую поля:
*номер маршрута; начальный пункт маршрута; конечный пункт маршрута;
длина маршрута.*

Написать функции:

- создания массива не более чем из 10 записей (структур) сведений о маршрутах (ввод данных с клавиатуры);
- 13 определения маршрута с максимальной длиной;
- расположения записей по возрастанию номеров маршрутов;
- вывода сведений о маршрутах, которые начинаются или заканчиваются в пункте, название которого вводится с клавиатуры.

Все необходимые данные для функций должны передаваться в качестве их параметров. Использование глобальных параметров не допускается. Создать проект, который демонстрирует работу всех функций.

4. **«Товар»:** *наименование; количество; цена; дата изготовления (год, месяц, число); изготовитель (название), номер телефона изготовителя; адрес (почтовый индекс, страна, область, район, город, улица, дом, офис).*

Определить общую и максимальную стоимость всех товаров, выпущенных в текущем году и вывести сведения об этих товарах.

Вариант 29

1. Решить задачу, используя функцию.

Дата некоторого дня характеризуется тремя натуральными числами (число, месяц, год). Определите количество дней, прошедших между двумя датами (учитывать високосные годы, даты начинаются с 1970 г.). Функция должна считать, сколько дней прошло с 1 января 1970 г. до текущей даты.

3. Описать структуру с именем NOTE3, содержащую поля: Name – фамилия и инициалы, TELE – номер телефона, SEX – пол, DATE – дата рождения (год, месяц, число).

Написать программу, выполняющую:

- ввод с клавиатуры данных в массив BLOCK2, состоящий из 7 элементов типа NOTE1, записи должны быть упорядочены по первым трем цифрам номера телефона;
- вывод на экран информации о человеке, чья фамилия введена с клавиатуры;
- если такого нет – выдать сообщение.

4. **«Владелец телефона»:** *фамилия; имя; отчество; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); № телефона.*

Вывести данные про владельцев телефона номер, которого начинается на 525.

Вариант 30

1. Решить задачу, используя функцию.

Дан массив натуральных чисел размерности n . Отсортировать массив в порядке убывания сумм цифр десятичной записи чисел. Функция должна считать сумму десятичных цифр натурального числа.

3. Описать структуру с именем MARSHRUT, содержащую поля:

*номер маршрута; начальный пункт маршрута; конечный пункт маршрута;
длина маршрута.*

Написать функции:

- создания массива не более чем из 10 записей (структур) сведений о маршрутах (ввод данных с клавиатуры);
- определения маршрута с максимальной длиной;
- расположения записей по возрастанию номеров маршрутов;
- вывода сведений о маршрутах, которые начинаются или заканчиваются в пункте, название которого вводится с клавиатуры.

Все необходимые данные для функций должны передаваться в качестве их параметров. Использование глобальных параметров не допускается. Создать проект, который демонстрирует работу всех функций.

4. «Школьник»: фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); школа; класс.

Вывести сведения про всех учеников седьмых классов.

Вариант 31

1. Решить задачу, используя функцию.

Дан массив натуральных чисел размерности n . Определить количество чисел-палиндромов в нем. Палиндром («перевертыш») — число, десятичная запись которого читается одинаково слева направо и справа налево (например, 14541). Функция должна проверять, является ли число палиндромом или нет.

3. Описать структуру с именем STUDENT, содержащую поля:

фамилия и инициалы студента; номер группы; успеваемость (массив из четырех оценок на экзаменах в 5-бальной системе).

Написать функции:

- создания массива 6 записей (структур) данных о студентах (ввод данных с клавиатуры);
- вычисления среднего бала каждого студента;
- расположения записей по убыванию среднего бала;
- вывода сведений о студентах, имеющих оценки только 4 и 5; удаления из списка студента с минимальным средним балом.

Все необходимые данные для функций должны передаваться в качестве их параметров. Использование глобальных параметров не допускается. Создать проект, который демонстрирует работу всех функций.

4. «Пациент»: фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); номер больницы; отделение; номер медицинской карты; диагноз; группа крови.

Вывести данные про пациентов с 4 отделения.

Вариант 32

1. Решить задачу, используя функцию.

Получить все шестизначные «счастливые» номера и подсчитать их количество. Функция должна проверять, является ли число «счастливым» или нет. «Счастливым» называется такое шестизначное число, у которого сумма первых трех цифр равна сумме последних трех.

3. Описать структуру с именем WORKER, содержащую поля:

фамилия, имя, отчество, должность, дата рождения, пол, дата приёма на работу.

Написать функции:

- создания массива не менее чем из 10 записей (структур) данных об сотрудниках (ввод данных с клавиатуры);
- определения количества сотрудников пенсионного возраста (мужчинам больше 65-ти лет, женщинам — 58);
- вывода результатов сортировки по полу и по алфавиту;
- вывода сведений о самом молодом сотруднике.
- вывода количества сотрудников каждого пола.

Все необходимые данные для функций должны передаваться в качестве их параметров. Использование глобальных параметров не допускается. Создать проект, который демонстрирует работу всех функций.

4. **«Кинолента»:** *название; режиссер (фамилия; имя); год выхода; страна; стоимость; доход; прибыль, жанр кино.* Вывести данные о фильмах режиссёра Даниила Козловского.

Вариант 33

1. Решить задачу, используя функцию.

В каждом из двух классов учатся по 20 человек. Известны средние оценки каждого ученика каждого класса, подсчитанные по ряду предметов (все значения для каждого класса разные). Определить, в каком классе у "пятого из самых успевающих учеников" средняя оценка больше.

3. Описать структуру с именем ABON1, содержащую поля:

фамилия и инициалы абонента; номер телефона; дата подключения телефона в виде структуры (год, месяц, день); тарифный план, начисленная сумма оплаты; сумма на счёту абонента.

Написать функции:

- создания массива не более чем из 10 записей (структур) данных об абонентах (ввод данных с клавиатуры);
- расположения записей по алфавиту (с учетом инициалов для абонентов с одинаковыми фамилиями);
- добавить 10 р. на счета абонентов, которых подключили более 10 лет назад;
- вывода сведений об абонентах, у которых сумма на счету отрицательная после вычета начислений;
- вывода сведений по количеству абонентов для каждого тарифного плана.

Все необходимые данные для функций должны передаваться в качестве их параметров. Использование глобальных параметров не допускается. Создать проект, который демонстрирует работу всех функций.

4. **«Покупатель»:** *фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц, число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); номер кредитной карточки; банковского счета.*

Вывести данные о покупателях города Пинск.

Вариант 34

1. Решить задачу, используя функцию.

В каждом из двух классов учатся по 25 человек. Известны значения веса и роста

каждого ученика этих классов. Определить, в каком классе "третий из самых полных учеников" выше по росту.

3. Описать структуру с именем *PRODUCT*, содержащую поля:

наименование, цена, дата производства, срок годности, количество, производитель.

Написать функции:

- создания массива продуктов не менее 10 записей (структур) сведений о продуктах (ввод с клавиатуры);
- вывести сведения с сортировкой по дате производства по возрастанию;
- вывести сведения о товарах, срок годности которых истекает через двое суток;
- вывести сведения о количестве товаров для каждого производителя;
- вывести сведения о товаре с максимальной ценой, срок годности которого не истек.

Все необходимые данные для функций должны передаваться в качестве их параметров. Использование глобальных параметров не допускается. Создать проект, который демонстрирует работу всех функций.

4. **«Товар»:** *наименование; стоимость; срок хранения; сорт; дата выпуска; срок годности.*

Вывести данные про товары срок годности которых истекает в этом году.

Вариант 35

1. Решить задачу, используя функцию.

Составить программу для нахождения общего количества заданной буквы в трёх заданных предложениях. (Определить функцию для расчёта количества некоторой буквы в предложении.)

3. Описать структуру с именем *CINEMA*, содержащую поля:

название фильма, дата и время сеанса, продолжительность сеанса, жанр, бюджет.

Написать функции:

- создания массива не менее чем из 12 записей (структур) данных о фильмах (ввод данных с клавиатуры);
- вывести сведения о фильмах с максимальной и минимальной продолжительностью;
- вывести данные о фильмах, начинающихся после 18:00 и продолжительностью сеанса менее 1 часа 30 минут;
- вывести сведения о фильмах жанра «Комедия» и с максимальным бюджетом;
- вывести сведения о фильмах с сортировкой по жанру и по бюджету в рамках жанра.

Все необходимые данные для функций должны передаваться в качестве их параметров. Использование глобальных параметров не допускается. Создать проект, который демонстрирует работу всех функций.

4. **«Рейс»:** *марка автомобиля; номер автомобиля; пункт назначения; грузоподъёмность (в тоннах); стоимость единицы груза; общая стоимость груза.*

Вывести данные про автомобили, грузоподъёмность которых больше 2 тонн.

Вариант 36

1. Решить задачу, используя функцию.

Дан массив натуральных чисел размерности *n*. Вывести на печать все числа

этого массива, не являющиеся полными квадратами. Функция должна проверять, является число полным квадратом или нет.

3. Описать структуру с именем PERSON, содержащую поля: Name – фамилия и инициалы, FAC – факультет, SPEC – специальность, GROUP – группа, DATE – дата поступления в ВУЗ (год, месяц, число).

Написать программу, выполняющую:

- ввод с клавиатуры данных в массив VUZ, состоящий из 10 элементов типа PERSON, записи должны быть упорядочены по дате поступления в ВУЗ; вывод на экран информации о студентах, упорядоченной по факультетам, специальностям, группам, дате поступления. В каждой группе фамилии должны быть расположены в алфавитном порядке.

4. **«Военнослужащий»:** фамилия; имя; отчество; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); национальность; дата рождения (год, месяц, число); должность; звание.

Вывести данные про военнослужащих в звании “сержант” и белорусов по национальности.

Вариант 37

1. Заданы две одинаковые по длине строки. Построить новую строку, в которой на нечётных местах расположены элементы первой строки, а на чётных – элементы второй строки.

3. Описать структуру с именем CONTACT, содержащую поля: Name – фамилия и инициалы, TELE – номер телефона, DATE – дата рождения (год, месяц, число).

Написать программу, выполняющую:

- ввод с клавиатуры данных в массив BLOCKNOTE, состоящий из 10 структур типа CONTACT, записи должны быть упорядочены по возрастанию даты рождения;
- вывод на экран сведений о человеке, номер телефона которого введён с клавиатуры;
- если такого человека нет – выдать сообщение.

4. **«Государство»:** название страны; столица; государственный язык; население; площадь территории; денежная единица; государственный строй; глава государства; часть света.

Вывести данные о государствах Европы, население которых больше 8 млн жителей.

Вариант 38

1. Получить из заданной строки две строки, состоящие из символов первой строки, имеющих соответственно четные и нечетные индексы. Не учитывать пробелы в первой строке.

3. Описать структуру с именем STUD, содержащую поля: Name – фамилия и инициалы, GROUP – название группы (факультет, курс, номер группы), SES – успеваемость (массив из четырёх элементов).

Написать программу, выполняющую:

- ввод с клавиатуры данных в массив STUD1, состоящий из 10 структур типа STUD,

записи должны быть упорядочены по алфавиту;

- вывод на экран данных о студентах, включенных в массив, средний балл которых превышает 4,2. Список упорядочить по возрастанию среднего балла. Сохранить информацию о положении студента в исходном списке;
- если таких студентов нет – выдать сообщение.

4. **«Человек»:** фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира).

Вывести сведения о самом молодом человеке.

Вариант 39

1. Задана строка. Получить четвертое с начала этой строки слово. Разделителем слов считаются один или несколько пробелов.

3. Описать структуру с именем STUDENT, содержащую поля: Name – фамилия и инициалы, Kurs – курс, SES – успеваемость (массив из пяти элементов).

Написать программу, выполняющую:

- ввод с клавиатуры данных в массив STUD, состоящий из 10 структур типа STUDENT, записи должны быть упорядочены по алфавиту;
- вывод на экран записей, упорядоченного списка студентов, средний бал которых превышает общий средний бал;
- если таких студентов нет – выдать сообщение.

4. **«Автомобиль»:** марка; цвет; серийный номер; регистрационный номер; год выпуска; год техосмотра; цена.

Вывести данные про автомобили, которым больше 3 лет.

Вариант 40

1. В заданной строке, разделённых пробелами и запятыми, расположить в обратном порядке все слова. Разделителями слов считаются запятая и пробел.

3. Описать структуру с именем GROUP, содержащую поля: Name – фамилия и инициалы, DAT – дата рождения (год, месяц, число), SES – успеваемость (массив из трех элементов).

Написать программу, выполняющую:

- ввод с клавиатуры данных в массив GR5, состоящий из 10 структур типа GROUP;
- вывод на экран записей, упорядоченных по возрастанию поля SES;
- вывод списка студентов, возраст которых на 1 декабря прошлого года не превышает 20 лет;
- если таких студентов нет – выдать сообщение.

4. **«Товар»:** наименование; количество; цена; дата изготовления (год, месяц число); изготовитель (название), номер телефона изготовителя; адрес (почтовый индекс, страна, область, район, город, улица, дом, офис).

Определить среднюю и минимальную стоимость всех товаров, выпущенных в прошлом году и вывести сведения об этих товарах.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Приведите примеры облачных IDE.

2. Опишите процесс подключения репозитория в repl.it.
3. Что такое «сборка»?
4. Какие утилиты могут использоваться для сборки?
5. Для чего служит Makefile?
6. Что такое цель в Makefile? Приведите пример.
7. Что такое связка в Makefile? Приведите пример.
8. Что такое зависимость в Makefile? Приведите пример.
9. Что такое правило в Makefile? Приведите пример.
10. Что такое макроопределение в Makefile? Приведите пример.
11. Какая связка используется для очистки проекта?
12. В чем заключается модель КИС для проектов на языке C?
13. Что такое стандарты кодирования и почему рекомендуется им следовать?
14. Какой должна быть структура проекта согласно модели КИС и правилам сборки?
15. Что такое Github Actions и для каких задач применяется данный функционал?
16. Что такое workflow в GitHub Actions? Приведите пример.
17. Что такое event в GitHub Actions? Приведите пример.
18. Что такое job в GitHub Actions? Приведите пример.
19. Что такое step в GitHub Actions? Приведите пример.
20. Что такое action в GitHub Actions? Приведите пример.

Материалы к занятию

Как быстро ознакомиться с отличительными характеристиками языка программирования?

Проект [Learn X in Y minutes](#) позволяет быстро ознакомиться с особенностями выбранного языка программирования.

Интегрированные веб-среды разработки и обучения

Распространенные веб-среды разработки программ.

- *IDEOne* — средство трансляции программ на 50 языках программирования, компиляторы которых расположены на сервере. Интерактивный ввод/вывод отсутствует. Представляет собой онлайн-компилятор, а не полноценную среду разработки.
- *Cloud9 IDE* — полноценная Web-среда, поддерживающая несколько языков, в том числе Python и Ruby. Предоставляет пользователям хранилище проектов на сервере, но интерактивный ввод/вывод не обеспечивается.
- *Compilr* — инструмент компиляции проектов, написанных на 10 языках программирования, в том числе Си, C++, C#, Python, Java, JavaScript, VB и PHP. Обеспечивает интерактивный ввод/вывод и предоставляет хранилище проектов на сервере.
- *CodeRun Studio* — полноценная бесплатная веб-среда разработки для языков C#, JavaScript, PHP с возможностью хранения проектов на сервере. Однако среди многочисленных типов проектов отсутствуют консольные приложения для C#.
- *SourceLair* — минималистский по интерфейсу веб-сервис, аналогичный по функциональности IDEOne. Позволяет компилировать и запускать программы на 15 языках программирования. Вывод при этом осуществляется на той же интернет-странице, ввод надо формировать предварительно в отдельной вкладке, интерактивный ввод/вывод отсутствует. Отсутствует также возможность сохранять проекты на сервере.

[Подробнее](#)

Команды терминала в Mac OS X. Редактор текстов nano

Простой консольный текстовый редактор nano сделан в стиле редактора pico для unix. Но в отличие от последнего имеет возможность поиска по редактируемому файлу и возможность начального позиционирования курсора. В Mac OS X дает возможность простого редактирования системных файлов.

Обычно запускается в терминале командой **nano имяФайла**, но обычно

используется для редактирования системных файлов с командой `sudo`

```
sudo nano ИмяФайла
```

Полный синтаксис команды запуска:

```
nano [ [параметры] [+строка,колонка] имя файла]
```

Основные параметры

?	подсказка
-B	(bаскир) при сохранении файла предыдущая версия сохраняется с суффиксом ~
-E	преобразовывать символы табуляции в пробелы
-L	не добавлять пустые строки в конце файла
-V	показывает текущую версию редактора nano
-i	обозначать новые строки (у меня не работает)
-m	включается поддержка мыши, если она поддерживается операционной системой (у меня не работает)
-v	открыть файл в режиме только для чтения (read-onli)
-x	отключить подсказку "горячих клавиш"

Основные команды редактора nano для Mac OS X показываются внизу экрана

Сохранить Файл	<i>ctrl-O</i>
Вырезать	<i>ctrl-K</i>
Вставить	<i>ctrl-U</i>
Найти	<i>ctrl-W</i>
Выйти	<i>ctrl-X</i>
Перелистнуть страницу вниз	<i>ctrl-V</i>
Перелистнуть страницу вверх	<i>ctrl-Y</i>
Копировать в буфер и удалить текущую строку	<i>ctrl-K</i>
Вставить скопированную строку	<i>ctrl-U</i>
Показать позицию курсора и данные о файле	<i>ctrl-C</i>
показывает	
<ul style="list-style-type: none">• номер строки и общее количество строк• номер колонки и общее количество колонок• номер символа и общее количество символов	
Подсказка/help	<i>ctrl-G</i>

Чтение и запись файлов

Основные понятия

Файл - это упорядоченная последовательность однотипных компонентов, расположенных на внешнем носителе. Файлы предназначены только для хранения информации, а обработка этой информации осуществляется программами. Использование файлов целесообразно в случае:

- долговременного хранения данных;
- доступа различных программ к одним и тем же данным;
- обработки больших массивов данных, которые невозможно целиком разместить в оперативной памяти компьютера.

Файл, не содержащий ни одного элемента, называется пустым. Создается файл путем добавления новых записей в конец первоначально пустого файла. Длина файла, т.е. количество элементов, не задается при определении файла. При вводе и выводе данные рассматриваются как поток байтов. Физически поток – это файл или устройство (клавиатура или дисплей). В Си поток можно открыть для чтения и/или записи в текстовом или бинарном (двоичном) режиме. В текстовых файлах не употребляются первые 31 символ кодовой таблицы ASCII (управляющие), а символы конца строки 0x13 (возврат каретки, CR) и 0x10 (перевод строки LF) преобразуются при вводе в одиночный символ перевода строки \n (при выводе выполняется обратное преобразование). Эти символы добавляются в конце каждой строки, записываемой в текстовый файл. При обнаружении в текстовом файле символа с кодом 26 (0x26), т.е. признака конца файла, чтение файла в текстовом режиме заканчивается, хотя файл может иметь продолжение.

В двоичном режиме эти преобразования не выполняются, чтение продолжается, пока не встретится физический конец файла. При чтении символы не преобразуются и не анализируются.

Функция открытия потока `fopen` возвращает указатель на предопределенную структуру типа `FILE` (содержащую всю необходимую для работы с потоком информацию) при успешном открытии потока, или `NULL` в противном случае.

В заголовочном файле `stdio.h` содержится описание файлового типа `FILE`, с которым связывается файловая переменная (указатель на файл). При открытии файла указатель на файл связывается с конкретным файлом на диске и определяется режим открытия файла:

- `r (r+)` - файл открывается для чтения (чтения и записи);
- `w (w+)` - открывается пустой файл для записи (чтения и записи). Если файл с таким именем существует, он стирается;
- `a (a+)` - файл открывается для дополнения в конец (чтения и дополнения).

Режим открытия может также содержать символы `t` (текстовый файл) и `b` (двоичный файл), указывающий на вид открываемого файла: `rb`, `wb`, `ab`, `rt`, `at`, `rb+`, `wb+`, `ab+` и т.д.

Закрытие файла (текстового или бинарного) выполняется функцией `fclose()`,

установка указателя на начало файла - функцией `rewind()`. Если при попытке чтения данных из файла встречается символ конца файла, то возвращается специальное значение EOF. Функции `feof()`, `ferror()` сообщают о причинах, по которым операция ввода/вывода не выполнялась. Запись данных в файл и чтение данных из файла можно выполнять разными способами:

1. функциями форматного ввода-вывода `fscanf()`, `fprintf()`;
2. функциями неформатного ввода-вывода `fread()`, `fwrite()`.

Если требуется сохранять и восстанавливать числовые данные без потери точности, то лучше использовать `fread()`, `fwrite()`. Если обрабатывается текстовая информация, которая будет просматриваться обычными текстовыми редакторами, то используется `fgetc()`- посимвольное чтение файла, посимвольная запись в файл - `fputc()` или функции `fscanf()`, `fprintf()`. Для чтения из файла и записи в файл строки используются функции `fgets()` и `fputs()`.