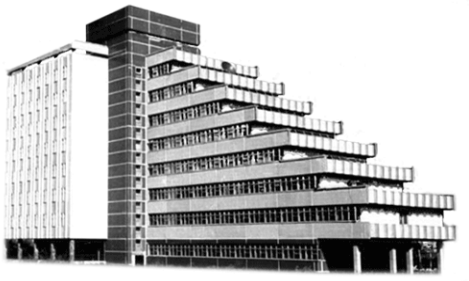
МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Учебный Центр Информационных Технологий «Информатика»



Лабораторная работа №2

по дисциплине «Информатика и программирование I часть»

Направление подготовки: 230105 - «Программное обеспечение вычислительной техники и автоматизированных систем»

Выполнил слушатель: Аветисян Арташес Робертович
Вариант: 13
Дата сдачи: 04.0102020
Преподаватель: Прытков Д. В.

Новосибирск, 2020г.

1 Задание

Найти в массиве наибольшее число подряд идущих одинаковых элементов (например {1,5,3,6,6,6,6,6,3,4,4,5,5,5} = 5).

2 Теоретический материал

Операторы в Си и управляющие структуры

Оператор – это наименьшая автономная часть языка программирования, команда.

Все операторы языка СИ разделены на категории:

- условные операторы, к которым относятся оператор условия if и оператор выбора switch;
- операторы цикла (for, while, do while);
- операторы перехода (break, continue, return, goto);
- Логические операторы (И, ИЛИ, НЕ);
- другие операторы (оператор “выражение”, пустой оператор);
- Составные операторы.

Все операторы языка СИ, кроме составных операторов, заканчиваются точкой с запятой “;”.

Условный оператор if

Оператор if позволяет выполнять или не выполнять определенные участки кода, в зависимости от того является ли истинным или ложным условие этого оператора. Одно из самых важных назначений оператора if это то, что он позволяет программе совершить действие на выбор, в зависимости от того, какие данные ввел пользователь. Условный оператор if может использоваться в форме полной или неполной развилки.

Оператор множественного выбора switch

Оператор if позволяет осуществить выбор только между двумя вариантами. Для того, чтобы производить выбор одного из нескольких вариантов необходимо использовать вложенный оператор if. С этой же целью можно использовать оператор ветвления switch.

Оператор switch выполняется следующим образом:

- вычисляется целочисленное выражение в скобках оператора switch;
- полученное значение сравнивается с метками (константами) в опциях case, сравнение производится до тех пор, пока не будет найдена метка, соответствующая вычисленному значению целочисленного выражения;
- выполняется оператор соответствующей метки case;
- если соответствующая метка не найдена, то выполнится оператор в опции default.

Опция break; осуществляет выход из оператора switch и переход к следующему за ним оператору. При отсутствии опции break будут выполняться все операторы, начиная с помеченного данной меткой и кончая оператором в опции default.

Оператор for

Оператор for – это наиболее общий способ организации цикла.

```
for ( выражение 1 ; выражение 2 ; выражение 3 ) тело
```

Выражение 1 обычно используется для установления начального значения переменных, управляющих циклом. Выражение 2 – это выражение, определяющее условие, при котором тело цикла будет выполняться. Выражение 3 определяет изменение переменных, управляющих циклом после каждого выполнения тела цикла.

Оператор while

Оператор цикла while называется циклом с предусловием. Его синтаксис:

```
while (выражение) тело ;
```

Выражение в скобках может принимать ненулевое (истинное) или нулевое (ложное) значение. Если оно истинно, то выполняется тело цикла и выражение вычисляется снова. Если выражение ложно, то цикл while заканчивается. Оператор while удобно использовать в ситуациях, когда тело оператора не всегда нужно выполнять.

Оператор do while

Основным отличием между циклами while и do – while является то, что тело в цикле do – while выполняется по крайней мере один раз. Тело цикла будет выполняться до тех пор, пока выражение в скобках не примет ложное значение. Если оно ложно при входе в цикл, то его тело выполняется ровно один раз. Его синтаксис:

```
do тело while (выражение);
```

Чтобы прервать выполнение цикла до того, как условие станет ложным, можно использовать оператор break.

Оператор continue

Оператор continue, как и оператор break, используется только внутри операторов цикла, но в отличие от него выполнение программы продолжается не с оператора, следующего за прерванным оператором, а с начала прерванного оператора.

Оператор return

Оператор return завершает выполнение функции, в которой он задан, и возвращает управление в вызывающую функцию, в точку, непосредственно следующую за вызовом. Использование оператора return необходимо либо для немедленного выхода из функции, либо для передачи возвращаемого значения.

Оператор goto

```
goto имя-метки;

...

имя-метки: оператор;
```

Оператор goto передает управление на оператор, помеченный меткой имя-метки. Помеченный оператор должен находиться в той же функции, что и оператор goto, а используемая метка должна быть уникальной, т.е. одно имя-метки не может быть использовано для разных операторов программы. Имя-метки – это идентификатор.

Использование оператора безусловного перехода goto в практике программирования на языке Си не рекомендуется, так как он затрудняет понимание программ и возможность их модификаций.

Оператор break

Оператор break обеспечивает прекращение выполнения самого внутреннего из объединяющих его операторов switch, do, for, while. После выполнения оператора break управление передается оператору, следующему за прерванным.

Логические операторы

Логические операторы – это операторы, которые принимают в качестве аргументов логические значений (ложь или истину) и возвращают логическое значение. Как и обычные операторы, они могут быть одноместными (унарными, т.е. принимать один аргумент), двуместными (бинарные, принимают два аргумента), трёхместными и т.д.

Оператор НЕ (NOT, Логическое отрицание) используется для того, чтобы инвертировать значение аргумента. т.е., если ему передали истину, то он вернёт ложь, если получил ложь в качестве аргумента, то вернёт истину. В Си отрицание представлено оператором !.

Оператор И (AND, логическое умножение) возвращает истину тогда и только тогда, когда оба аргумента являются истиной. В Си логическое умножение представлено оператором &&.

Оператор логическое ИЛИ (логическое сложение, OR) истинен тогда, когда истиной является хотя бы один его аргумент. В Си ИЛИ представлен оператором ||.

Оператор “выражение”

Любое выражение, которое заканчивается точкой с запятой, является оператором. Выполнение оператора выражение заключается в вычислении выражения. Полученное значение выражения никак не используется, поэтому, как правило, такие выражения вызывают побочные эффекты.

```
++ i; //Этот оператор представляет выражение, которое увеличивает значение переменной i на единицу.
```

Пустой оператор

Пустой оператор состоит только из точки с запятой. При выполнении этого оператора ничего не происходит. Он обычно используется в операторах do, for, while, if в строках, когда место оператора не требуется, но по синтаксису требуется хотя бы один оператор и при необходимости пометить фигурную скобку.

Составной оператор

Составной оператор представляет собой несколько операторов и объявлений, заключенных в фигурные скобки:

```
{   [объявление]
            :
            оператор; [оператор];
            :
```

```
}
```

Выполнение составного оператора заключается в последовательном выполнении составляющих его операторов.

Массивы в Си

Массив – это непрерывный участок памяти, содержащий последовательность объектов одинакового типа, обозначаемый одним именем. Массивы — очень полезные сущности, особенно в тех случаях, когда необходимо под одним именем объединить группы переменных одного и того же типа, обращаться к таким переменным можно через целочисленный индекс.

Элемент массива – значение, хранящееся в определенной ячейке памяти, расположенной в пределах массива, а также адрес этой ячейки памяти.

Каждый элемент массива характеризуется тремя величинами:

- адресом элемента – адресом начальной ячейки памяти, в которой расположен этот элемент;
- индексом элемента (порядковым номером элемента в массиве);
- значением элемента.

Синтаксис объявления массива:

```
int instanceArray[10];
```

Это целочисленный массив размером 10 ячеек. Чтобы обратиться к конкретному элементу массива, нужно просто указать имя массива и в квадратных скобках — порядковый номер. Единственно, следует знать, что первый элемент массива имеет индекс — 0, тогда индекс последнего элемента равен размеру массива минус один. Например, индексы для массива из 10 элементов будут соответствовать этому диапазону: от 0 до 9.

Многомерные массивы

Многомерные массивы — это массивы, у которых есть более одного индекса. Вместо одной строки элементов, многомерные массивы можно рассматривать как совокупность элементов, которые распределены по двум или более измерениям.

В многомерных массивах требуется некоторое время на вычисление каждого индекса. Это означает, что доступ к элементу в многомерных массивах происходит медленнее, чем доступ в одномерных массивах. По этой и другим причинам, если возникает необходимость в многомерных массивах, для них чаще всего память выделяется динамически с использованием функции динамического выделения памяти.

Передача массива в функцию

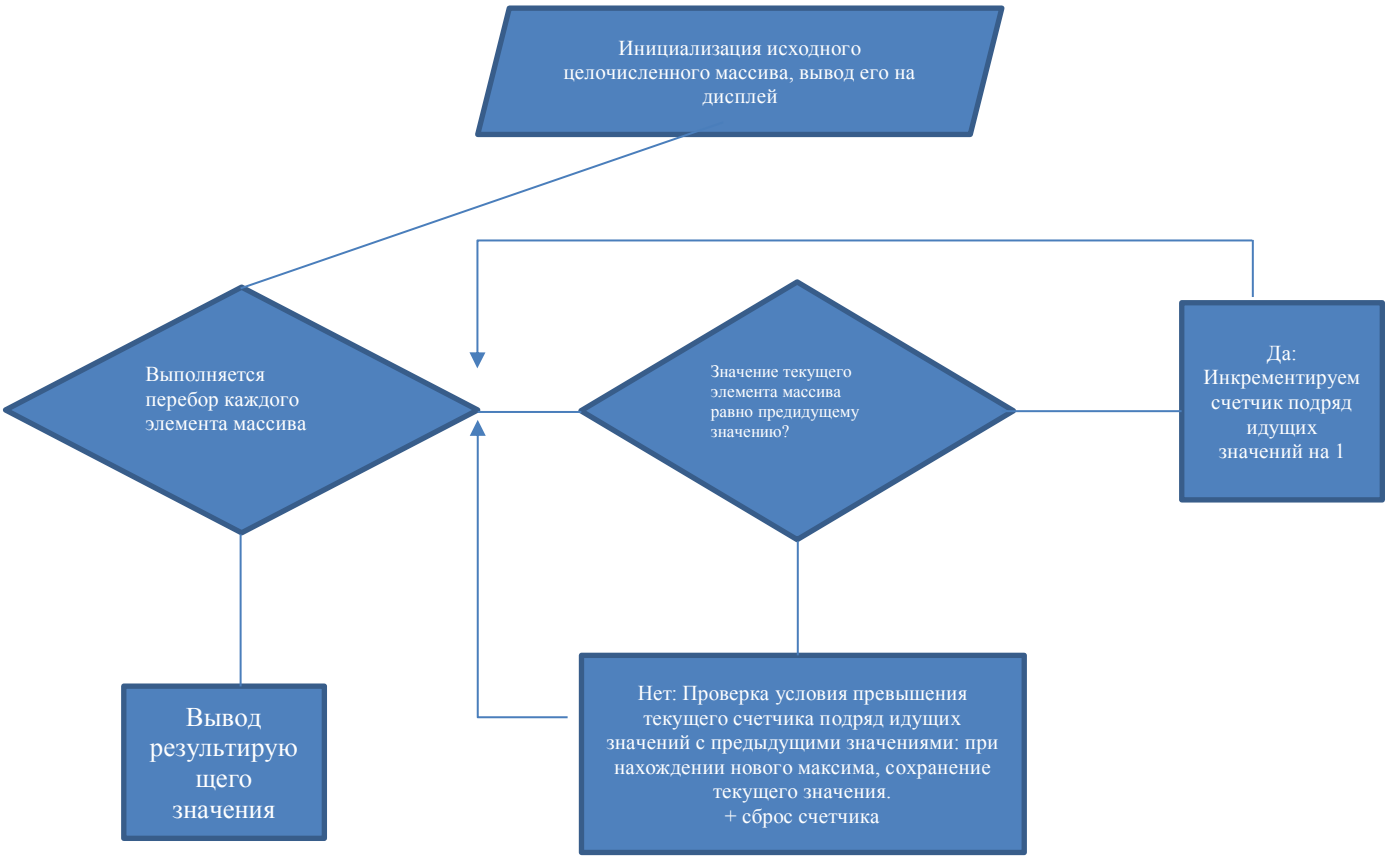
Обработку массивов удобно организовывать с помощью специальных функций. Для обработки массива в качестве аргументов функции необходимо передать:

- адрес массива,
- размер массива.

При передаче переменные в качестве аргументов функции данные передаются как копии. Это означает, что если внутри функции произойдет изменение значения параметра, то это никак не повлияет на его значение внутри вызывающей функции.

Если в функцию передается адрес массива, то все операции, выполняемые в функции с данными, находящимися в пределах видимости указанного адреса, производятся над оригиналом данных, поэтому исходный массив может быть изменено вызываемой функцией.

3 Описание алгоритма приложения



4 Описание реализации

Препроцессорные директивы:

#include - используется для включения в исходный код заголовочных файлов библиотек

Подключаемые заголовочные файлы библиотеки:

- stdio.h

Используемые функции:

int main()

Указанная основная функция, являющаяся начальной точкой для выполнения программы. Она обычно управляет выполнением программы, вызывая другие ее функции. Как правило, выполнение программы завершается в конце функции **main**, но по разным причинам это может случиться и в других местах программы. Может быть ипользована для передачи аргументов в приложение.

int printf(const char *format, arg-list)

Функция printf() записывает в stdout аргументы из списка arg-list под управлением строки, на которую указывает аргумент format. Строка, на которую указывает format, состоит из объектов двух различных назначений. Во- первых, это символы, которые сами должны быть выведены на экран. Во-вторых, это спецификаторы формата, определяющие вид, в котором будут выведены аргументы из списка arg-list. Спецификаторы формата состоят из символа процент, за которым следует код формата.

sizeof

Функция strlen() возвращает длину строки, оканчивающейся нулевым символом, на которую указывает str. При определении длины строки нулевой символ не учитывается.

Используемые конструкции и операторы:

sizeof

Унарный оператор, возвращающий длину в байтах переменной или типа

if

Оператор выбора **if** позволяет выполнять или опустить выполнять определенных участков кода, в зависимости от того является ли истинным или ложным условие этого оператора. Одно из самых важных назначений оператора выбора **if** так это то, что он позволяет программе совершить действие на выбор, в зависимости от того, например, какие данные ввел пользователь.

for

Параметрический цикл (цикл с заданным числом повторений), реализуемый при помощи операций инициализация (присваивание параметру цикла начального значения), проверки условия повторения цикла, модификации - изменения значения параметра для следующего прохождения тела цикла

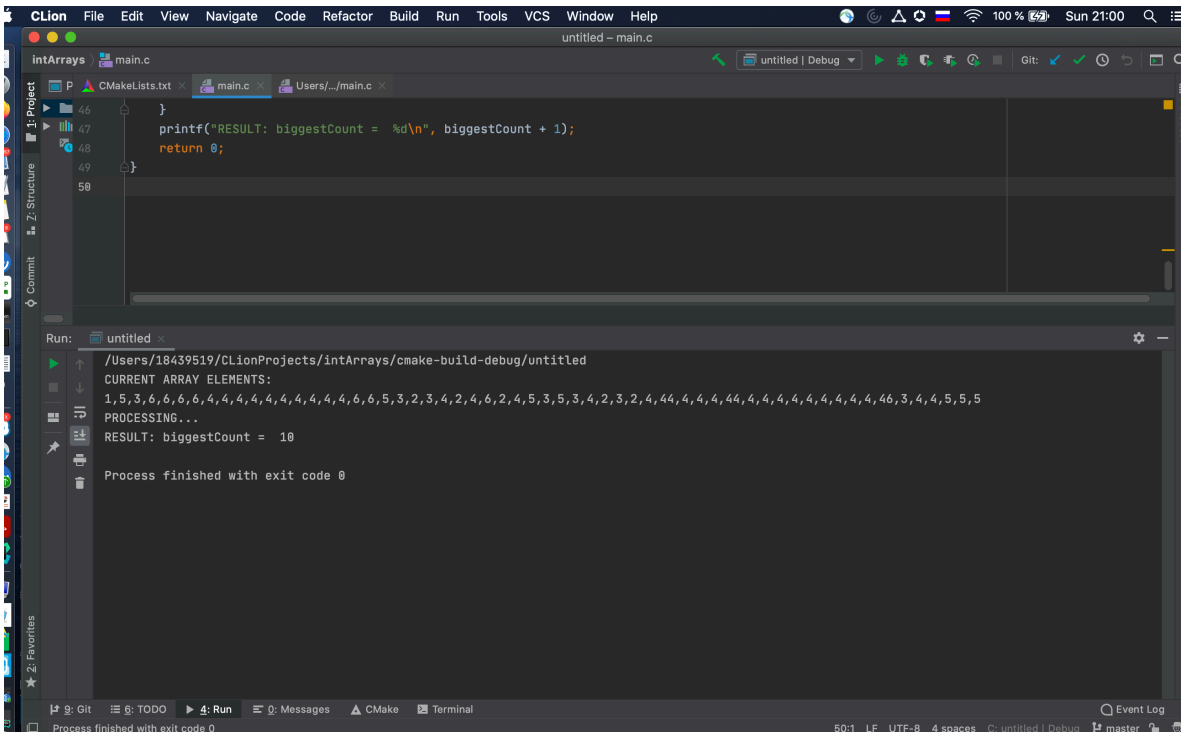
5 Пример работы программы

На рисунке 1 показана фотография дисплея компьютера с развернутым окном среды разработки JetBrains CLion после выполнения разработанного приложения. На третьей строке консоли при помощи функции printf выведена обрабатываемый массив в исходном виде.

На четвертой строке вывод сообщения “**PROCESSING…**”, после чего начинается исполнение основного алгоритма приложения.

На пятой строке выведен результат выполнения алгоритма.

Рисунок 1. Фотография экрана во время выполнения приложения, разработанного согласно задачи лабораторной работы



Выводы

Изучен теоретическим материал по работе опраторами, управляющими конструкциями и целочисленными массивами в Си, который применен в практике для решения задачи в рамках данной лабораторной работы.

Успешно разработан и апробирован алгоритм поиска в массиве наибольшее число подряд идущих одинаковых элементов.

Приложение. Текст программы с комментариями

```
#include <stdio.h>

/**
 * Изучение операторов, управляющих структур, работа с целочисленными массивами в Си
 * Лабораторная №2: вариант 13. Исполнил: Аветисян Арташес Робертович
 * 13. Найти в массиве наибольшее число подряд идущих одинаковых элементов
 * (например {1,5,3,6,6,6,6,6,3,4,4,5,5,5} = 5).
 */


int main() {

    //int rawData[] = {1,5,3,6,6,6,6,6,3,4,4,5,5,5};
    int rawData[] =
{1,5,3,6,6,6,6,4,4,4,4,4,4,4,4,4,4,6,6,5,3,2,3,4,2,4,6,2,4,5,3,5,3,4,2,3,2,4,44,4,4,4,44,4,4,4,4,4,4,4,4,4,4
6,3,4,4,5,5,5};
    int rawDataArraySize = sizeof rawData / sizeof rawData[0];

    printf("CURRENT ARRAY ELEMENTS:\n");
    for (int a = 0; a < rawDataArraySize; a++ ){
        if (a!=rawDataArraySize-1) {
            printf("%d,", rawData[a]);
        } else {
            printf("%d", rawData[a]);
        }
    }

    printf("\nPROCESSING...\n");

    int biggestCount = 0;
    int sameValueCountHolder = 0;
    for (int i = 0; i < rawDataArraySize; i++ ) {
        if (i != 0){
            if (rawData[i-1] == rawData[i]){
                //printf("SAME: rawData[i-1]: %d, rawData[i]: %d\n", rawData[i-1], rawData[i]);
                sameValueCountHolder++;
                //printf("UPDATING: sameValueCountHolder =  %d\n", sameValueCountHolder);
            } else {
                //printf("NOT SAME: rawData[i-1]: %d, rawData[i]: %d\n", rawData[i-1], rawData[i]);
                if (sameValueCountHolder > biggestCount){
                    biggestCount = sameValueCountHolder;
                    //printf("UPDATING: biggestCount =  %d\n", biggestCount);
                }
                sameValueCountHolder = 0;
            };
            //printf("Element[%d] = %d, sameValueCountHolder: %d\n", i, rawData[i],
sameValueCountHolder);
        }
    }
    printf("RESULT: biggestCount =  %d\n", biggestCount + 1);
    return 0;
}
```

Защита: