УВАЖАЕМЫЕ СТУДЕНТЫ! Изучите теоретические сведения к лабораторной работе, выполните практическое задание.

Результаты работы, фотоотчет, предоставить преподавателю на e-mail: r.bigangel@gmail.com до 08.05.2023.

Требования к отчету:

Отчет предоставляется преподавателю в электронном варианте и должен содержать:

- **название работы, постановку цели, вывод;**
- **ответы на контрольные вопросы, указанные преподавателем.**

При возникновении вопросов по приведенному материалу обращаться по следующему номеру телефона: (072)111-37-59, (Viber, WhatsApp), vk.com: https://vk.com/daykini

ВНИМАНИЕ!!! При отправке работы, не забывайте указывать ФИО студента, наименование дисциплины, дата проведения занятия (по расписанию).

Лабораторная работа № 20

Тема: «Указатели»

Цель: изучить понятие указатели, научиться использовать указатели при обработки динамических данных. Научиться использовать операции динамического выделения и освобождения памяти на примере работы с одномерными и двумерными массивами, а также косвенное обращение к элементам массива.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

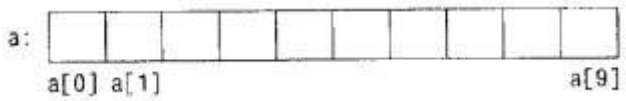
Массивы указателей

В Си существует связь между указателями и массивами, и связь эта настолько тесная, что эти средства лучше рассматривать вместе. Любой доступ к элементу массива, осуществляемый операцией индексирования, может быть выполнен с помощью указателя. Вариант с указателями в общем случае работает быстрее, но разобраться в нем, особенно непосвященному, довольно трудно.

Объявление

int a[10];

Определяет массив a размера 10, т. е. блок из 10 последовательных объектов с именами a[0], a[1], ..., a[9].



Запись a[i] отсылает нас к i-му элементу массива. Если pa есть указатель на int, т. е. объявлен как

int *pa;

то в результате присваивания

pa = &a[0];

pa будет указывать на нулевой элемент a, иначе говоря, pa будет содержать адрес элемента a[0].

Теперь присваивание

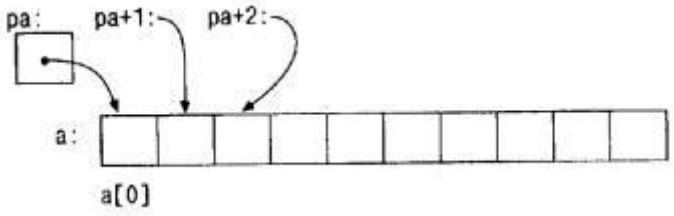
x = *pa;

будет копировать содержимое a[0] в x.

Если pa указывает на некоторый элемент массива, то pa+1 по определению указывает на следующий элемент, pa+i - на i-й элемент после pa, а pa-i - на i-й элемент перед pa. Таким образом, если pa указывает на a[0], то

*(pa+1)

есть содержимое a[1], a+i - адрес a[i], а *(pa+i) - содержимое a[i].



Сделанные замечания верны безотносительно к типу и размеру элементов массива a. Смысл слов "добавить 1 к указателю", как и смысл любой арифметики с указателями, состоит в том, чтобы pa+1 указывал на следующий объект, а pa+i - на i-й после pa.

Между индексированием и арифметикой с указателями существует очень тесная связь. По определению значение переменной или выражения типа массив есть адрес нулевого элемента массива. После присваивания

pa = &a[0];

pa и a имеют одно и то же значение. Поскольку имя массива является синонимом расположения его начального элемента, присваивание pa=&a[0] можно также записать в следующем виде:

pa = a;

Еще более удивительно (по крайней мере на первый взгляд) то, что a[i] можно записать как *(a+i). Вычисляя a[i], Си сразу преобразует его в *(a+i); указанные две формы записи эквивалентны. Из этого следует, что полученные в результате применения оператора & записи &a[i] и a+i также будут эквивалентными, т. е. и в том и в другом случае это адрес i-го элемента после a. С другой стороны, если pa - указатель, то его можно использовать с индексом, т. е. запись pa[i] эквивалентна записи *(pa+i). Короче говоря, элемент массива можно изображать как в виде указателя со смещением, так и в виде имени массива с индексом.

Между именем массива и указателем, выступающим в роли имени массива, существует одно различие. Указатель - это переменная, поэтому можно написать pa=a или pa++. Но имя массива не является переменной, и записи вроде a=pa или a++ не допускаются.

Если имя массива передается функции, то последняя получает в качестве аргумента адрес его начального элемента. Внутри вызываемой функции этот аргумент является локальной переменной, содержащей адрес. Мы можем воспользоваться отмеченным фактом и написать еще одну версию функции strlen, вычисляющей длину строки.

```
/* strlen: возвращает длину строки */
int strlen(char *s)
{
    int n;
```

```
    for (n = 0; *s != '\0'; s++)
        n++;
    return n;
}
```

Так как переменная s - указатель, к ней применима операция ++; s++ не оказывает никакого влияния на строку символов функции, которая обратилась к strlen. Просто увеличивается на 1 некоторая копия указателя, находящаяся в личном пользовании функции strlen. Это значит, что все вызовы, такие как:

```
strlen("Здравствуй, мир"); /* строковая константа */
strlen(array);          /* char array[100]; */
strlen(ptr);            /* char *ptr; */
```

правомерны.

Формальные параметры

```
char s[];
```

и

```
char *s;
```

в определении функции эквивалентны. Мы отдаем предпочтение последнему варианту, поскольку он более явно сообщает, что s есть указатель. Если функции в качестве аргумента передается имя массива, то она может рассматривать его так, как ей удобно - либо как имя массива, либо как указатель, и поступать с ним соответственно. Она может даже использовать оба вида записи, если это покажется уместным и понятным.

Функции можно передать часть массива, для этого аргумент должен указывать на начало подмассива. Например, если a - массив, то в записях

```
f(&a[2])
```

или

```
f(a+2)
```

функции f передается адрес подмассива, начинающегося с элемента a[2]. Внутри функции f описание параметров может выглядеть как

f(int arr[]) {...}

или

f(int *arr) {...}

Следовательно, для f тот факт, что параметр указывает на часть массива, а не на весь массив, не имеет значения.

Если есть уверенность, что элементы массива существуют, то возможно индексирование и в "обратную" сторону по отношению к нулевому элементу; выражения p[-1], p[-2] и т.д. не противоречат синтаксису языка и обращаются к элементам, стоящим непосредственно перед p[0]. Разумеется, нельзя "выходить" за границы массива и тем самым обращаться к несуществующим объектам.

Дополнительная информации по теме:

1. https://code-live.ru/post/cpp-array-tutorial-part-2/

2. https://metanit.com/cpp/tutorial/4.5.php

3. https://habr.com/ru/post/251091/

Задание к лабораторной работе:

1. Ознакомиться с теоретическим материалом.

2. Проверить свою теоретическую подготовку по контрольным вопросам.

3. В соответствии с вариантом составить блок-схемы и программы для решения индивидуальной задачи.

4. Подготовить отчет по лабораторной работе.

Задание:

1: В компьютер вводятся фамилии и рост студентов группы. Вывести на экран фамилии тех студентов, рост которых больше 170 см (для определения кандидатов в баскетбольную команду).

2: При продаже грампластинок ведется учет количества проданных пластинок с классической музыкой, эстрадной и детских. Составить программу, ведущую этот учет за рабочий день.

3: Заданна последовательность натуральных чисел N, Найти в этой последовательности взаимно простые числа.

Примечание: Числа a, в, c,.... d называются взаимно простыми

числами, если наибольший обший их делитель (НОД) равен 1, Числа 5, , 10, 7,3-взаимно простые, т.к. НОД (5;10;7;3)=1

4: Написать программу нахождения наибольшего общего делителя двух целых чисел, используя алгоритм Евклида.

5: Найти НОД заданных трех натуральных чисел.

6: Найти наименьшее общее кратное (НОК) нескольких (3,4, ...) заданных натуральных чисел.

7: Дано натуральное число N. Выбросить из записи числа цифры 3 и 7, оставив прежним порядок остальных цифр.

8: Написать программу сложения двух простых дробей. (Числители и знаменатели вводятся в переменные M, N, P и Q с клавиатуры). Результат сложения вывести на экран также в виде простой дроби.

9: Дано натуральное число N. Заменить в нем все «двойки» на «пятерки».

10: Даны два числа. Написать программу, в результате выполнения которой выводится первое число, если оно больше второго, и оба числа, если это не так.

11: Даны два числа. Написать программу, в результате выполнения которой первое число заменяется нулем, если оно больше второго.

12: Даны три числа a,. b, c. Написать программу, в результате выполнения которой числа удвоятся, если a > b > c, и числа будут замены на их абсолютные величины в прочих случаях:

13: Написать программу, в результате выполнения которой выясняется, входит ли цифра 2 в запись данного целого числа n.

14: Составить программу, по которой для любого введенного целого а определяется его четность. Выводится число 1, если а нечетное, и число 2, если а четное.

15: Натуральное число называется совершенным, если оно равно сумме всех своих делителей, за исключением самого себя. Число 6 является совершенным, так как 6=1 +2+3, число 8 не является совершенным, так как 8 ≠ 1+2+4. Написать программу вывода всех совершенных чисел, меньших заданного числа N.