Лабораторная работа №6

Указатели. Динамическое распределение памяти

Цель работы: Изучить операции работы с указателями и научиться использовать функции динамического выделения памяти для одномерных массивов.

Теоретические сведения

Указатели в С используются для связи переменных с машинными адресами. В программах указатели используются для доступа к памяти и манипуляций с адресами. Если v-переменная, то &v- это адрес или место в памяти, где хранится ее значение.
Объявляется указатель следующим образом:
type *name;
где type – тип, на который будет указывать указатель;
* - звездочка определяет тип “указатель”;
name – имя переменной.
Например: Объявление int *p; говорит о том, что переменная p будет иметь тип «указатель на целое». Если p-указатель, то *p – значение переменной, на которую указывает p.

```
int  i=5,j;
int *p;
p=&i;    //p указывает на i
j=*p;    // j=5
```

Адреса переменных можно использовать в качестве аргументов функций. В результате значения переменных могут изменяться в вызывающем окружении. Указатели используются в списке параметров для определения адресов переменных, значения которых могут изменяться. Например:

```
void order(int *, int *);
int main()
{
        int i=7, j=3;                       // инициализация переменных i и j
        printf(“\n i=%3d j=%3d”,i, j);
        order(&i,&j);                       // вызов функции order
        printf(“\n i=%3d j=%3d”,i, j);
}
void order(int *p, int *q)
{
        int temp;
        if(*p > *q)                         // перемена значений по адресам p и q
        {
                temp=*p;
                *p=*q;
                *q=temp;
        }
```

}

В языке С для обработки элементов массивов удобно использовать указатель на этот массив. Любой доступ к элементу массива может быть выполнен при помощи указателя. Например:

```
int mas[10];   //  массив
int *ptr;      // указатель на int
```

в результате присваивания

```
ptr=&mas[0];
```

ptr будет указывать на нулевой элемент массива mas, иначе говоря, ptr будет содержать адрес элемента mas[0]. Если ptr указывает на некоторый элемент массива, то ptr+1 указывает на следующий элемент массива. Поскольку имя массива есть адрес начального элемента массива, то присваивание ptr=&mas[0]; можно реализовать и так: ptr=mas;

Используя указатели, память под массивы можно отводить динамически, т.е. размещать в свободной памяти (free store). Свободная память – это предоставляемая системой область памяти для объектов, время жизни которых напрямую управляется программистом. В языке С функции для динамического выделения памяти объявлены в стандартной библиотеке в заголовочном файле stdlib.h – malloc(), calloc(), realloc(), free(); Функции динамического управления памятью делятся на функции динамического выделения и освобождения памяти. К функциям выделения памяти относятся malloc() и calloc(). Функция освобождения памяти – free().

Прототип функции

void *malloc(unsigned size); Эта функция выделяет область памяти размером size байт. Функция malloc возвращает указатель на начало выделенного блока памяти, если для выделения блока не хватает памяти, то возвращается NULL.

Прототип функции

void *calloc(unsigned num, unsigned size);

Функция calloc выделяет блок памяти и возвращает указатель на первый байт блока. Размер выделяемой памяти равен величине num*size, т.е. функция выделяет память, необходимую для хранения массива из num элементов по size байт каждый. Если память не выделена, то функция calloc возвращает NULL.

Прототип функции

void *realloc(void *ptr, unsigned size);

Эта функция изменяет размер динамически выделенной памяти, на которую указывает *ptr, на size(новый размер). Значение size задает новый размер блока. Если указатель не является значением, которое ранее было определено функциями malloc, calloc или realloc, то функция ведет себя неопределенно.

Прототип функции void free(void *ptr);

Функция освобождает область памяти, ранее выделенную при помощи функций malloc, calloc или realloc.

Указатель на указатель для работы с многомерными массивами

Цель работы: Научиться использовать указатель на указатель при работе с двухмерными массивами.

Теоретические сведения

Можно объявлять переменные, имеющие тип «указатель на указатель». Например: int **mas; Указатель на указатель – это адрес ячейки, хранящий адрес указателя. При определении указатель на указатель можно инициализировать. Например:

```
int mm=10;              // переменная типа int
int *ptr=&mm;           //  указатель на переменную типа int
int **pptr=&ptr;        // указатель на указатель
```

Для доступа к переменной mm теперь можно использовать операции взятия по адресу и индексы: ptr[0], *ptr, pptr[0][0], **pptr.

Выделить память под двухмерный массив используя указатель на указатель можно следующим образом:

```
int **ptr;
int n;          // количество строк
int m;          // количество столбцов
printf("\n Введите количество строк и столбцов\n");
scanf("%d%d",&n,&m);
ptr=(int **)calloc(n,sizeof(int *));
for(int i=0; i < n; i++)
        ptr[i]=(int *)calloc(m,sizeof(int));
printf("\n Введите элементы массива\n");
for(int i=0; i < n; i++)
        for(int j=0; j < m; j++)
                scanf("%d",(ptr+i)+j);
printf("\n Исходный массив\n");
for(int i=0; i < n; i++)
{
        printf("\n");
        for(int j=0; j < m; j++)
                printf("%4d",ptr[i][j]);
}
```

Порядок выполнения работы

1. Изучить краткие теоретические сведения.
2. Составить блок-схему алгоритма.
3. По разработанной блок-схеме алгоритма написать программу. Память под массивы выделять динамически как указатель на указатель.
4. Отладить и выполнить программу.

Варианты заданий

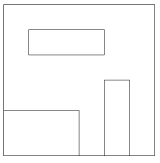
1. Рассортировать отрицательные элементы каждого столбца матрицы по возрастанию. Положительные элементы оставить на своих местах.

2. Даны натуральное число n, действительное число x, действительная матрица размера n*2n. Получить последовательность $b_1,\ldots,b_n$ из нулей и единиц, где $b_i=1$, если элементы i-ой строки матрицы не превосходят x, и $b_i=0$ в противном случае.

3. Даны целочисленная матрица размера n*m, целые числа k,k1 (1 <=k <=n, 1 <= k1 <=n, k ≠ k1). Преобразовать матрицу так, чтобы строка с исходным номером k непосредственно следовала за строкой с исходным номером k1, сохранив порядок следования остальных строк.

4. Дана действительная квадратная матрица порядка n. Рассмотрим те элементы, которые расположены в строках, начинающихся с отрицательного элемента. Найти суммы этих элементов, которые расположены соответственно ниже, выше и на главной диагонали. Суммы найденных элементов хранить в массиве. Память под массивы выделять динамически.

5. На квадратном листе клетчатой бумаги размера n*n клеток нарисовано несколько прямоугольников. Различные прямоугольники не накладываются друг на друга и не соприкасаются. Определить число прямоугольников. Память под массив выделять динамически.



6. Таблица футбольного чемпионата задана квадратной матрицей порядка n, в которой все элементы, принадлежащие главной диагонали равны 0, а каждый элемент, не принадлежащий главной диагонали, равен 2, 1 или 0 (число очков набранных в игре: 2 – выигрыш, 1 – ничья, 0 – проигрыш).

а) найти число команд, имеющих больше побед, чем поражений;

б) определить номера команд, прошедших чемпионат без поражений;

в) выяснить, имеется ли хотя бы одна команда, выигравшая более половины игр. Память под массивы отводить динамически.

7. В действительной квадратной матрице порядка n найти наибольший по модулю элемент. Получить квадратную матрицу порядка n-1 путем выбрасывания из исходной матрицы какой-нибудь строки и столбца, на пересечении которых расположен элемент с найденным значением.

8. Даны действительные числа $a_1,\ldots a_n$, действительная квадратная матрица порядка n. Получить действительную матрицу размера n*(n+1), вставив в исходную матрицу между j и j+1 столбцами новый столбец с элементами $a_1,\ldots a_n$.

9. Латинским квадратом порядка n называется квадратная таблица размера n*n, каждая строка и каждый столбец которой содержит числа 1,2,…,n. Дана целочисленная квадратная матрица; определить, является ли она латинским квадратом.

10. Дана действительная квадратная матрица размера n. Расположить элементы матрицы следующим образом:

1	2	3	4	5
16	17	18	19	6
15	24	25	20	7
14	23	22	21	8
13	12	11	10	9

11. В квадратной матрице порядка n найти наибольший по модулю элемент. Получить квадратную матрицу порядка n-1 путем выбрасывания из исходной матрицы какой-нибудь строки и столбца, на пересечении которых расположен элемент с найденным значением.

12. Дана матрица. Поменять местами максимальный элемент среди всех отрицательных элементов матрицы на минимальный элемент среди всех положительных.

13. Рассортировать отрицательные элементы каждого столбца матрицы по возрастанию. Положительные элементы оставить на своих местах.

14. В произвольной матрице - отсортировать по убыванию элементы последовательности, расположенные после второго отрицательного числа.

15. Определить, является ли данный квадратный массив симметричным относительно своей главной диагонали.