Лабораторная работа № 8. Указатели. Динамические массивы

Цель работы: получить практические навыки разработки программ на языке C++ с использованием указателей и адресов, управление динамической памятью.

Теоретическое обоснование

1. Указатели

Указатель – это переменная, содержащая адрес некоторого объекта, например, другой переменной, точнее адрес первого байта этого объекта. Это дает возможность косвенного доступа к этому объекту через указатель.

Пусть x – переменная типа int. Обозначим через px указатель. Унарная операция & выдает адрес объекта, так что оператор

px = &x;

присваивает переменной px адрес переменной x.

Говорят, что px «указывает» на x. Операция & применима только к адресным выражениям, так что конструкции вида &(x-1) и &3 незаконны.

Унарная операция * называется операцией разадресации или операцией разрешения адреса. Эта операция рассматривает свой операнд как адрес и обращается по этому адресу, чтобы извлечь объект, содержащийся по этому адресу.

Следовательно, если **y** тоже имеет тип **int**, то

y = *px;

присваивает y содержимое того, на что указывает px. Так, последовательность

px = &x;

y = *px;

присваивает y то же самое значение, что и оператор y = x;

Все эти переменные должны быть описаны:

int x, y; int *px;

Последнее – описание указателя. Его можно рассматривать как мнемоническое. Оно говорит, что комбинация *px имеет тип int, или, иначе, px есть указатель на int. Это означает, что если px появляется в виде *px, то это эквивалентно переменной типа int.

Из описания указателя следует, что он может указывать только на определенный вид объекта (в данном случае int). Разадресованный указатель может входить в любые выражения там, где может появиться объект того типа, на который этот указатель ссылается. Так, оператор

y = *px + 2;

присваивает y значение, на 2 больше, чем x.

Заметим, что приоритет унарных операций * и & таков, что эти операции связаны со своими операндами более крепко, чем арифметические операции, так что выражение

y = *px + 2

берет то значение, на которое указывает px, прибавляет 2 и присваивает результат переменной y.

Если px указывает на x, то

*px = 3;

полагает x равным 3, а

*px + = 1;

увеличивает x на 1 так же, как и выражение (*px)++

Круглые скобки здесь необходимы. Если их опустить, то есть написать *px++, то, поскольку унарные операции, подобные * и ++, выполняются справа налево, это выражение увеличит px, а не ту переменную, на которую он указывает.

Если py – другой указатель на int, то можно выполнить присвоение

py = px;

Здесь адрес из px копируется в py. В результате py указывает на то же, что и px.

Как уже известно, массив – это совокупность элементов одного типа, которые расположены в памяти ЭВМ подряд, один за другим. Признаком объявления массива являются квадратные скобки. Объявить массив из 8 элементов типа double можно так:

double arr [8];

Чтобы обратиться к элементу этого массива, нужно применить операцию индексирования arr [i]. Здесь i – целое выражение, которое называется индексом. Нумеруются элементы массива начиная с 0, и поэтому вышеприведенное описание означает, что в памяти ЭВМ зарезервировано место под 8 переменных типа double, а сами эти переменные есть arr [0], arr [1], …, arr [7].

В языке C++ имя массива является константным указателем на нулевой элемент этого массива:

int mas[30];

int *pm;

pm = &mas[0];

Последний оператор можно записать и так: pm = mas;

Операция индексирования массива [ ] имеет 2 операнда – имя массива, т. е. указатель, и индекс, т. е. целое: arr[i]. В языке C++ любое выражение указатель[индекс] трактуется как

*(указатель + индекс)

и всегда автоматически преобразуется к такому виду при компиляции.

Таким образом, arr[5] эквивалентно *(arr + 5), и это можно записать даже как 5[arr], так как это все равно проинтерпретируется как *(5+a). Здесь

складываются указатель arr и целое 5. В связи с этим рассмотрим так называемую арифметику над указателями, или адресную арифметику.

Если к указателю **pa** прибавляется целое приращение i, то приращение масштабируется размером памяти, занимаемой объектом, на который указывает указатель **pa**.

Таким образом, **pa + i** – это адрес i-го элемента после **pa**, причем считается, что размер всех этих i элементов равен размеру объекта, на который указывает **pa**.

Если a – массив, то

a + i – адрес i-го элемента этого массива, т. е. &a[i] равен a + i и a[i] равняется *(a + i).

doble b[100];

double *pb = b;

pb++;// Это эквивалентно pb = pb + 1.

// Здесь указатель pb будет указывать на элемент массива b[1].

pb += 4; // Здесь pb указывает на элемент массива b[5].

Однако нельзя написать b++ или b = b + i, так как имя массива b – это константный указатель и его изменять нельзя.

Если p и q указывают на элементы одного и того же массива, то отношения сравнения, такие как > >= и т. д., работают обычным образом. Например, p < q истинно, т. е. равно 1, если p указывает на более ранний элемент массива, чем q. Любой указатель можно сравнить с помощью операций == и != с так называемым нулевым указателем NULL, который ни на что не указывает. Однако не рекомендуется сравнивать указатели, указывающие на различные массивы.

Указатели можно вычитать.

Если p и q указывают на элементы одного и того же массива, то p - q дает количество элементов массива между p и q.

2. Многомерные массивы

Двумерный массив рассматриваются как массив элементов, каждый из которых является одномерным массивом. Трехмерный – как массив, элементами которого являются двумерные массивы и т. д.

После объявления

int a[5][6][7];

в программе могут появиться выражения:

a[i][j][j] – объект типа int;

a[2][0] – объект типа int* – одномерный массив из 7 целых;

a[1] – двумерный массив из 6*7 = 42 целых;

a- – сам трехмерный массив

Так как элементом массива a является двумерный подмассив размером 6*7, то при выполнении выражения a + 1 происходит смещение на величину элемента массива a, т. е. переход от a[0] к a[1]. Значение адреса при этом увеличивается на 6*7*sizeof(int) = 84.

Для двумерного массива mas выражение mas[i][j] интерпретируется как *(*(mas + i) +j). Здесь mas[i] – константный указатель на i-ю строку массива mas.

В памяти массивы хранятся по строкам, т. е. при обращении к элементам в порядке их размещения в памяти быстрее всего меняется самый правый индекс.

Так, для массива c[2][3] его шесть элементов расположены в памяти так:

c[0][0] c[0][1] c[0][2] c[1][0] c[1][1] c[1][2].

Многомерные массивы также можно инициализировать при описании:

int d[2][3]={ 1, 2, 0, 5 };

В этом случае первые 4 элемента массива получат указанные значения, а остальные два инициализируются нулями.

Если инициализируется многомерный массив, то самую первую размерность можно не задавать. В этом случае компилятор сам вычисляет размер массива:

```
int f [ ][2] = { 2, 4, 6, 1 };                    // массив  f [2][2];
int a [ ][2][2] = { 1, 2, 3, 4, 5, 6, 7, 8 };     // массив  a [2][2][2].
```

Инициализирующее выражение может иметь вид, отражающий факт, что массив является, например, двумерным:

```
int c[2][3]={ {1, 7}, {-5, 3} };
```

В этом случае в матрице с инициализированы нулевой и первый столбцы, а второй столбец, т. е. элементы c[0][2] и c[1][2], инициализируется нулями.

3. Указатели и многомерные массивы

Рассмотрим разницу между объектами a и b, описанными следующим образом:

```
double a[20][20];
double * b[20];
```

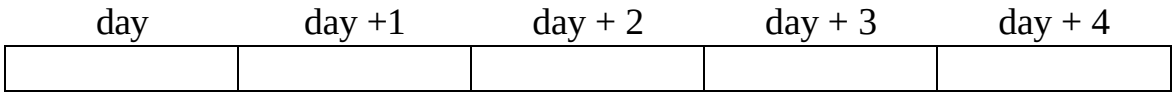
И a и b можно использовать одинаково в том смысле, что как a[6][5], так и b[6][5] являются обращениями к отдельному значению типа double. Но a – настоящий массив: под него отводится 400 ячеек памяти и для нахождения любого указанного элемента проводятся обычные вычисления с индексами, которые требуют умножения. Для b описание выделяет только 20 указателей. Каждый из них должен быть определен так, чтобы он указывал на массив double.

Если предположить, что каждый из них указывает на массив из 20 элементов, то тогда где-то будет отведено 400 ячеек памяти плюс еще 20 ячеек для указателей. Таким образом, массив указателей использует несколько больше памяти и может требовать наличия явного шага инициализации. Но при этом возникают два преимущества: доступ к эле- менту

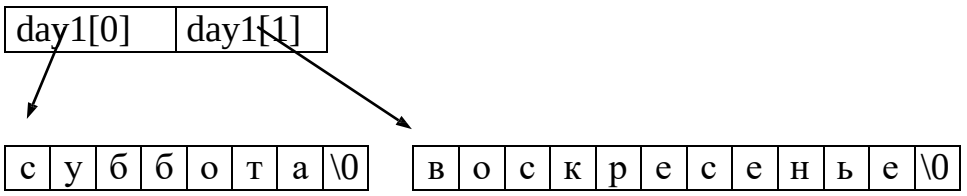
осуществляется косвенно через указатель, а не посредством умножения и сложения, а строки массива могут иметь различные длины. Это означает, что каждый элемент b не должен обязательно указывать на вектор из 20 элементов. Эта разница демонстрируется в следующем примере:

```
char day [5][12] = {// В каждой строке 12 символов.
"понедельник",
"вторник",
"среда",
"четверг",
"пятница"
};
```

Здесь константные указатели day[0], day[1], …, day[4] адресуют участки памяти одинаковой длины – 12 байт каждый:




```
const char * day1[2] =
{"суббота",                  // 7 символов + '/0'
  "воскресенье"              // 11 символов + '/0'
};
```



Переменные-указатели day1[0] и day1[1] адресуют участки памяти соответственно в 8 и 12 байт.

Пример 1:

Вычисление суммы элементов массива с использованием указателя

```
#include<iostream>


void main()
{
	int a[5] = { 3,18,25,7,12 };
	int sum = 0, i;
	int* p;
```

```
	p = a; // аналогично p = &a[0];
	for (i = 0; i < 5; i++)
	{
		sum += *p; // аналогично sum+=a[i] ; sum+=*(a+i) ;
		p++; // смещение указателя к следующему элементу
	}
	std:: cout << "sum=" << sum;
	}
```

Второй вариант того же цикла:

```
#include<iostream>


void main()
{
	int a[5] = { 3,18,25,7,12 };
	int sum = 0, i;
	int* p;
	p = a; // аналогично p = &a[0];
	for (i = 0; i < 5; i++)
	{
		sum += p[i];
	}// аналогично sum + =*(p+i); sum+=*(a+i) ;

	std:: cout << "sum=" << sum;
}
```

4. Операция sizeof

Эта операция выполняется на стадии компиляции. Результатом этой операции является число байтов, необходимое для размещения объекта в памяти. Существует два варианта синтаксиса этой операции. В первом из них единственный операнд операции определяет некоторый тип языка, и он должен быть заключен в скобки:

sizeof (float)

sizeof (int)

Во втором операнд задает некоторое выражение, и здесь использование скобок необязательно:

sizeof a;

sizeof *ip;

sizeof array[i];

Заметим, что при получении размеров массивов, несмотря на то, что имя массива является указателем, результатом операции

sizeof array,

где array – имя некоторого массива, является длина в байтах этого массива. Это свойство можно использовать для вычисления числа элементов в массиве:

```
const int n = 20; int array [n];
. . .
int num=sizeof array / sizeof(int)        // num равно 20.
```

Результатом операции sizeof над ссылкой является длина типа, с которым сопоставлена ссылка, т. е. sizeof(double &) и sizeof(double) эквивалентны.

5. Операции для работы с динамической памятью

Операция выделения памяти new

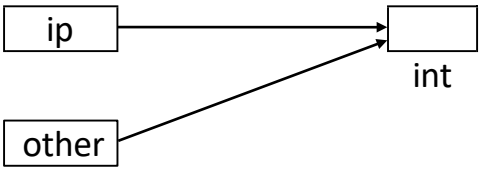
Часто выражение, содержащее операцию new, имеет следующий вид: указатель_на_тип_= new имя_типа (инициализатор)

Инициализатор – это необязательное инициализирующее выражение, которое может использоваться для всех типов, кроме массивов.

При выполнении оператора int *ip = new int;

создаются 2 объекта: динамический безымянный объект и указатель на него с именем ip, значением которого является адрес динамического объекта. Можно создать и другой указатель на тот же динамический объект:

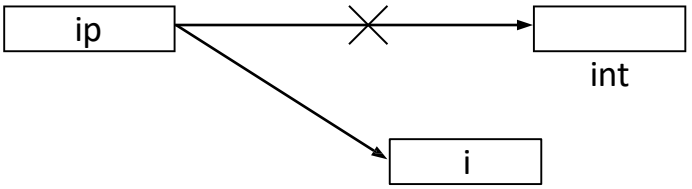
int *other = ip;



Если указателю ip присвоить другое значение, то можно потерять

доступ к динамическому объекту:

```
int *ip = new(int);
int  i = 0;
ip = &i;
```



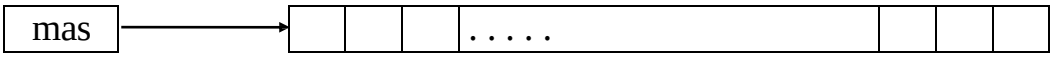
Теперь динамический объект по-прежнему будет существовать, но обратится к нему уже нельзя. Такие объекты называются мусором.

При выделении памяти объект можно инициализировать:

```
int *ip = new int(3);
```

Можно динамически распределить память и под массив:

```
double *mas = new double [50];
```



Теперь с этой динамически выделенной памятью можно работать как с обычным массивом:

```
*(mas + 5) = 3.27;
mas[6] = mas[5] + sin(mas[5]);
```

В случае успешного завершения операция new возвращает указатель со значением, отличным от нуля.

По умолчанию, если операция new не сработала (т. е. не найден непрерывный свободный фрагмент памяти нужного размера и память не выделилась), то генерируется так называемое исключение bad_alloc. Если это исключение будет неправильно обработано, то программа просто прекратит своё выполнение (произойдёт сбой) с ошибкой необработанного исключения.

Иногда процесс генерации исключения операцией new (как и сбой программы) нежелателен, поэтому есть альтернативная форма этой операции, которая возвращает нулевой указатель (NULL или nullptr), если

память не может быть выделена. Нужно просто добавить константу std::nothrow между ключевым словом new и типом данных:

```
int *ptr = new(std::nothrow) int;
```

Здесь указатель ptr станет нулевым, если динамическое выделение памяти не произойдет.

Поэтому хорошей практикой является проверка всех запросов на выделение памяти для обеспечения того, что эти запросы будут выполнены успешно и память выделится:

```
int *ptr = new(std::nothrow) int;
if(!ptr){
cout << "Could not allocate memory"; exit(1); // Например так!
}
```

Операция освобождения памяти delete

Операция delete освобождает для дальнейшего использования в программе участок памяти, ранее выделенной операцией new:

```
delete ip;      // Удаляет динамический объект типа int,
// если было ip = new int;
delete []mas;          // удаляет динамический массив длиной 50,
// если было double *mas = new double[50];
```

Совершенно безопасно применять операцию к указателю NULL. Результат же повторного применения операции delete к одному и тому же указателю не определен. Обычно происходит ошибка, приводящая к зацикливанию.

Чтобы избежать подобных ошибок, можно применять следующую конструкцию:

```
int *ip = new int[500];
. . .
if(ip){delete []ip;
```

ip = NULL;}

else

{ cout <<" память уже освобождена \n";

}

Пример 2. Заполнить массив случайными числами

```
#include <iostream>

using namespace std;

int main()
{
    srand((unsigned)time(NULL));
    int N;
    cout << "Введите размерность массива:" << endl;
    cout << "N = ";
    cin >> N;
    int* A = new int[N];
    int* ptr = A; //Объявляем указатель на массив
    cout << "Случайный массив [-100, 100]:" << endl;
    for (int i = 0; i < N; i++)
    {
        A[i] = -100 + (rand() % 201); //Заполняем массив случайными числами
        cout << *(ptr + i) << " "; //Выводим массив с помощью указателя
    }
    cout << endl;
    delete[] A;
    system("pause");
    return 0;
      }
```

Пример 3. Распределить память для матрицы из m строк и n столбцов:

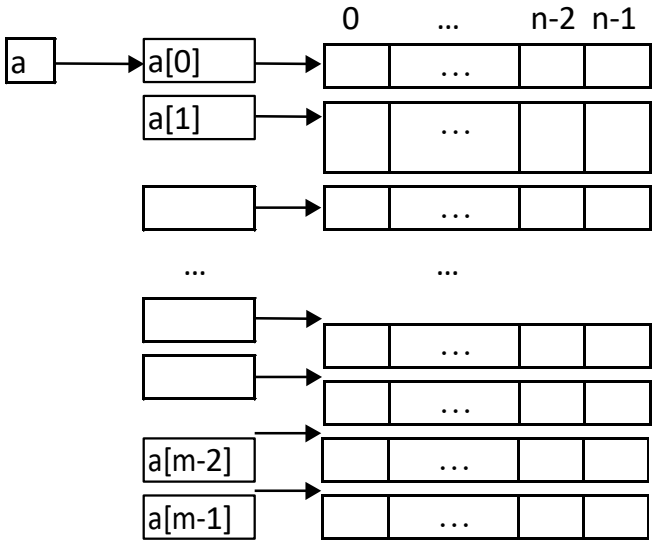
```
#include <iostream>

using namespace std;

int main()
{
      int m, n;
      cout << "Задайте число строк и столбцов матрицы: \n";
      cin >> m >> n;
      double** a = new double* [m];   // Массив из m указателей на double
      for (int i = 0; i < m; i++ )    // Распределяется строка матрицы:
      if ((a[i] = new(std::nothrow) double[n]) == NULL)
      {
            cout << "Нет памяти!\n"; exit(1);
      }
```

}

Теперь к элементам этой матрицы можно обращаться обычным образом:

a[i][j] или *(a[i] + j) или *(*(a + i) + j)

Изобразить распределение памяти, соответствующее вышеприведенному фрагменту, можно следующим образом:



Освободить память здесь можно так:

```
for (i = 0; i < m; i++)
delete []a[i];
delete []a;
```

Или так:

```
for(i = 0; i < m; i++){
delete []a[i];
a[i] = NULL;
}
delete []a;
```

Практическая часть

Пример к заданию 1.

Ввести значение 2-х целых переменных a и b. Направить два указателя на эти переменные. С помощью указателя увеличить значение переменной a в 2 раза, а b уменьшить в 2 раза.

Реализация

```
#include <iostream>

using namespace std;

void main(void)
{
	setlocale(LC_ALL, "Russian");
	int a, b;

	cout << "Введите 2 целых числа через пробел: ";
	cin >> a >> b;

	cout << endl << "Входные значения" << endl;
	cout << "\t - значение переменной a: " << a << endl;
	cout << "\t - значение переменной b: " << b << endl;

	// направляем два указателя на эти переменные
	int* pa = &a;
	int* pb = &b;

	// с помощью направленных указателей
	*pa *= 2; // увеличиваем значение a в 2 раза
	*pb /= 2; // уменьшаем значение b в 2 раза

	// выводим на экран измененные значения переменных a и b
	cout << endl << "Выходные значения" << endl;
	cout << "\t - значение переменной a: " << a << endl;
	cout << "\t - значение переменной b: " << b << endl;

	cout << endl << "Для завершения работы программы нажмите ENTER...";

	}
```

Пример к заданию 2.

Описать 2 указателя на целый тип. Выделить для них динамическую память. Присвоить произвольные значения в выделенные ячейки в операторе присвоения.

Реализация

```
#include <iostream>


using namespace std;

void main(void)
{
```

```
	// руссификация диалогов
	setlocale(LC_ALL, "Russian");

	// объявляем 2 указателя на целый тип
	int* a, * b;

	// выделяем для них динамическую память
	a = new int;
	b = new int;

	// присваиваем произвольные значения в выделенные ячейки в операторе
присвоения
	*a = 70;
	*b = -25;

	// выводим на экран значения ячеек памяти, на которые указывают a и b
	cout << endl << "Значения выделенных ячеек" << endl;
	cout << "\t - указатель *a ссылается на значение: " << *a << endl;
	cout << "\t - указатель *b ссылается на значение: " << *b << endl;

	// удаляем память из-под динамических переменных
	delete a;
	delete b;

	cout << endl << "Для завершения работы программы нажмите ENTER...";
	}
```

Пример к заданию 3.

Создать динамические массивы, используя указатели. Дан массив r(n). Каждый элемент равный 0 в нем заменить на 1 Остальные оставить прежними.

Реализация

```
#include <iostream>

using namespace std;

void main(void)
{
	const int MAX_SIZE = 20; // максимально допустимое количество элементов в
массиве r
	int n; // активное количество элементов массива r
	int* r; // одномерный динамический целочисленный массив r(n)

	setlocale(LC_ALL, "Russian");

	// запрашиваем от пользователя активное количество элементов массива r с
проверкой на границы
	do
	{
		cout << "Введите количество элементов массива из отрезка [1 .. " <<
MAX_SIZE << "]: ";
		cin >> n;
	} while ((n < 1) || (n > MAX_SIZE));
```

```
	// выделяем память под динамический массив r, состоящий из n элементов
целого типа
	r = new int[n];

	// запрашиваем от пользователя ввод значений элементов массива с клавиатуры
	cout << endl << "Введите с клавиатуры " << n << " элем. целого типа" <<
endl;
	for (int i = 0; i < n; i++)
	{
		cout << "\t - введите элемент r[" << (i + 1) << "]: ";
		cin >> r[i];
	}

	// выводим значения элементов массива r на экран в строку
	cout << endl << "Элементы исходного массива:\n";
	for (int i = 0; i < n; i++)
		cout << "\t" << r[i];

	// начинается обработка: все элементы равные 0 заменяем на 1
	for (int i = 0; i < n; i++)
		if (r[i] == 0) // если текущий элемент равен 0, то
			r[i] = 1; // заменяем его на 1

			// выводим обработанный массив на экран
	cout << endl << "\nЭлементы обработанного массива:\n";
	for (int i = 0; i < n; i++)
		cout << "\t" << r[i];

	// удаляем динамическую память из-под элементов массива r
	delete[]r;

	cout << endl << "\nДля завершения работы программы нажмите ENTER...";

	}
```

Пример к заданию 4.

Создать динамические массивы, используя указатели. Дан массив x (n). Переписать в массив y(n) элементы массива x, большие 3 (со сжатием., без пустых элементов внутри). Затем упорядочить возрастанию новый массив.

Реализация

```
#include <iostream>

using namespace std;

void main(void)
{
	const int MAX_SIZE = 20; // максимально допустимое количество элементов в
массивах
	int n; // активное количество элементов массива x(n)
	int m; // активное количество элементов массива y(m)
	int* x; // одномерный динамический целочисленный массив x(n)
	int* y; // одномерный динамический целочисленный массив y(m)
```

```
	setlocale(LC_ALL, "Russian");

	// запрашиваем от пользователя активное количество элементов массива x с проверкой на границы
	do
	{
		cout << "Введите количество элементов массива X(n) из отрезка [1 .. " << MAX_SIZE << "]: ";
		cin >> n;
	} while ((n < 1) || (n > MAX_SIZE));

	// выделяем память под динамический массив x, состоящий из n элементов целого типа
	x = new int[n];

	// запрашиваем от пользователя ввод значений элементов массива x(n) с клавиатуры
	cout << endl << "Введите с клавиатуры " << n << " элем. целого типа" << endl;
	for (int i = 0; i < n; i++)
	{
		cout << "\t - введите элемент X[" << (i + 1) << "]: ";
		cin >> x[i];
	}

	// выводим значения элементов массива x на экран в строку
	cout << endl << "Элементы исходного массива X(n):\n";
	for (int i = 0; i < n; i++)
		cout << "\t" << x[i];

	// считаем количество элементов в массиве x, значение которых больше 3
	m = 0;
	for (int i = 0; i < n; i++)
		if (x[i] > 3) // если текущий элемент массива больше 3, то
			m++; // увеличиваем счетчик чисел, больших 3, на +1

			// выделяем память под динамический массив y(m)
	y = new int[m];

	//  обработка: переписываем элементы из массива x, большие 3, в массив y
	int k = -1; // индекс элементов в массиве y для вставки элемента из массива x
	for (int i = 0; i < n; i++)
		if (x[i] > 3) // если текущий элемент массива x больше 3, то
			y[++k] = x[i]; // переносим его в массив y

	if (!m)
		cout << endl << endl << "В массиве y нет ни одного элемента! Сортировка  невозможна." << endl;
	else
	{
		// выводим полученный массив y(m) на экран
		cout << endl << "\nЭлементы массива Y (до сортировки):\n";
		for (int i = 0; i < m; i++)
			cout << "\t" << y[i];

		// начинается сортировка массива y(m) методом выбора по возрастанию
		for (int i = 0; i < (m - 1); i++)
		{
			int index_min = i;
			for (int j = (i + 1); j < m; j++)
```

```
                        if (y[j] < y[index_min])
                              index_min = j;
                  if (i != index_min)
                  {
                        int swap = y[i];
                        y[i] = y[index_min];
                        y[index_min] = swap;
                  }
            }

            // выводим отсортированный массив y(m) на экран
            cout << endl << "\nЭлементы массива Y (после сортировки):\n";
            for (int i = 0; i < m; i++)
                  cout << "\t" << y[i];
}

// удаляем динамическую память из-под элементов массивов x, y
delete[]x;
delete[]y;

cout << endl << "\nДля завершения работы программы нажмите ENTER...";
}
```

Выполнить следующие задания.

Варианты заданий

Задание №1. Указатели и адреса

1. Ввести значение 2-х целых переменных a и b. Направить два указателя на эти переменные. С помощью указателя увеличить значение переменной a в 2 раза. Затем поменять местами значения переменных a и b через их указатели.

2. Ввести значение 2-х целых переменных a и b. Направить два указателя на эти переменные. С помощью указателя увеличить значение переменной a в 2 раза если a>b иначе b уменьшить в 2 раза.

3. Ввести значение 2-х вещественных переменных a и b. Направить два указателя на эти переменные. С помощью указателя увеличить значение

переменной a в 3 раза. Затем поменять местами значения переменных a и b через их указатели.

4. Ввести значение 2-х вещественных переменных a и b. Направить два указателя на эти переменные. Если a>b, то с помощью указателя увеличить значение переменной a на 3 и b уменьшить в 3 раза, в противном случае a уменьшить в 2 раза и b увеличить на 3.

5. Ввести значение 2-х символьных переменных a и b. Направить два указателя на эти переменные. С помощью указателя изменить значение переменной a. Затем поменять местами значения переменных a и b через их указатели.

6. Ввести значение 2-х целых переменных a и b. Направить два указателя на эти переменные. Большее из них с помощью указателя увеличить в 5 раз и меньшее уменьшить на 5.

7. Ввести значение 3-х целых переменных a и b и c. Направить указатели на эти переменные. С помощью указателя увеличить значение переменной a в 2 раза. Затем поменять местами значения переменных c и b через их указатели.

8. Ввести значение 3-х вещественных переменных a и b и c. Направить указатели на эти переменные. С помощью указателя увеличить значение переменной c в 3 раза. Затем поменять местами значения переменных a и c через их указатели.

9. Ввести значение 2-х вещественных переменных a и b. Направить два указателя на эти переменные. Большее из них с помощью указателя увеличить на7 и меньшее уменьшить на 3.

10. Ввести значение 2-х символьных переменных a и b. Направить два указателя на эти переменные. Затем поменять местами значения переменных a и b через их указатели.

11. Ввести значение 2-х целых переменных a и b. Направить два указателя на эти переменные. Затем поменять местами значения переменных a и b через их указатели.

12. Ввести значение 2-х вещественных переменных a и b. Направить два указателя на эти переменные. Затем поменять местами значения переменных a и b через их указатели.

13. Ввести значение 2-х вещественных переменных a и b. Направить два указателя на эти переменные. С помощью указателя увеличить значение переменной a в 3 раза , a b уменьшить в 3 раза.

14. Ввести значение 2-х вещественных переменных a и b. Направить два указателя на эти переменные. С помощью указателя увеличить значение переменной a в 3 раза , a b уменьшить в 3 раза.

Задание №2 Указатели и Динамическая память с помощью оператора new()

1. Описать 2 указателя на целый тип. Выделить для них динамическую память. Ввести значения в выделенную память с клавиатуры. Уменьшить в 2 раза 1-ую переменную.

2. Описать 2 указателя на вещественный тип. Выделить для них динамическую память. Ввести значения в выделенную память с клавиатуры. Увеличить в 2 раза 1-ую переменную.

3. Описать 3 указателя на символьный тип. Выделить для них динамическую память. Ввести значения в выделенную память с клавиатуры.

4. Описать 2 указателя на логический тип. Выделить для них динамическую память. Присвоить значения true и false в выделенную память.

5. Описать 3 указателя на вещественный тип. Выделить для них динамическую память. Присвоить произвольные значения в выделенные ячейки в операторе присвоения. Уменьшить в 2 раза 1-ую переменную.

6. Описать 1 указатель на символьный тип. Выделить для него динамическую память. Присвоить произвольное значение в выделенную ячейку в операторе присвоения.

7. Описать 2 указателя на целый тип. Выделить для них динамическую память. Ввести значения в выделенную память с клавиатуры. Поменять местами их значения.

8. Описать 2 указателя на вещественный тип. Выделить для них динамическую память. Ввести значения в выделенную память с клавиатуры. Поменять местами их значения.

9. Описать 3 указателя на символьный тип. Выделить для них динамическую память. Ввести значения в выделенную память с клавиатуры. Поменять местами значения первых 2-х переменных.

10. Описать 2 указателя на логический тип. Выделить для них динамическую память. Присвоить значения true и false в выделенную память . Поменять местами их значения.

11. Описать 2 указателя на целый тип. Выделить для них динамическую память. Присвоить произвольные значения в выделенные ячейки в операторе присвоения. Поменять местами их значения.

12. Описать 3 указателя на вещественный тип. Выделить для них динамическую память. Присвоить произвольные значения в выделенные ячейки в операторе присвоения. Поменять местами значения первых 2-х переменных.

13. Описать 1 указатель на символьный тип. Выделить для него динамическую память. Присвоить произвольное значение в выделенную ячейку в операторе присвоения.

14. Описать 1 указатель на целый тип. Выделить для него динамическую память. Ввести значения в выделенную память с клавиатуры. Затем увеличить ее на 2.

Задание №3 Динамические массивы

1. Создать динамические массивы, используя указатели. Задан одномерный массив a (n). Найти количество, все номера и произведение элементов массива меньших 1.

2. Создать динамические массивы, используя указатели. Дано 2 массива x(n) и y(m). Сколько раз встречается второй элемент первого массива x(n) во втором массиве y(m).

3. Создать динамические массивы, используя указатели. В каком из двух данных массивов p (n) q(n) больше отрицательных элементов?

4. Создать динамические массивы, используя указатели. Дан массив p(n). Каждый положительный элемент в нем возвести в квадрат. Остальные элементы оставить прежними.

5. Создать динамические массивы, используя указатели. Задан одномерный массив a (n). Найти номер последнего элемента меньшего заданного числа betta, количество положительных элементов и сумму элементов больших 3.

6. Создать динамические массивы, используя указатели. В каком из двух данных массивов p (n) q(n) больше положительных элементов?

7. Создать динамические массивы, используя указатели. Дано 2 массива x(n) и y(m). Сколько раз встречается первый элемент первого массива x(n) во втором массиве.

8. Создать динамические массивы, используя указатели. Задан одномерный массив a (n). Найти номер последнего элемента равного 5 и

переставить его с первым элементом массива. Найти среднее арифметическое элементов массива больших заданного числа alfa.

9. Создать динамические массивы, используя указатели. Задан одномерный массив a (n). Найти номер последнего положительного элемента и переставить его с первым элементом массива. Найти количество и сумму элементов отрицательных массива.

10. Создать динамические массивы, используя указатели. Дано 2 массива x(n) и y(m). Сколько раз встречается последний элемент первого массива x(n) во втором массиве y(m).

11. Создать динамические массивы, используя указатели. В каком из двух данных массивов p (n) q(n) больше нулевых элементов?

12. Создать динамические массивы, используя указатели. Задан одномерный массив a (n). Найти все номера и среднее арифметическое отрицательных элементов массива.

13. Создать динамические массивы, используя указатели. Дано 2 массива x(n) и y(m). Сколько раз встречается второй элемент второго массива y(m) в первом массиве.

14. Создать динамические массивы, используя указатели. Дано 2 массива x(n) и y(m). Сколько раз встречается первый элемент второго массива y(m) в первом массиве.

15. Создать динамические массивы, используя указатели. В каком из двух данных массивов p (n) q(n) больше элементов, равных 1?

Задание №4. Составить алгоритмы и программы к следующим задачам.

1. Создать динамические массивы, используя указатели. Дан массив b (n). Переписать в массив C(n) положительные элементы массива

b(n) умноженные на 5 (со сжатием, без пустых элементов внутри). Затем упорядочить по возрастанию новый массив.

2. Создать динамические массивы, используя указатели. Дан массив a (n). Переписать в массив b(n) только положительные элементы массива a, умноженные на 3 (со сжатием, без пустых элементов внутри). Затем упорядочить по возрастанию новый массив.

3. Создать динамические массивы, используя указатели. Дан массив x (n). Переписать в массив y(n) отрицательные элементы массива x умноженные на 2 (со сжатием, без пустых элементов внутри) Затем упорядочить по возрастанию новый массив.

4. Создать динамические массивы, используя указатели. Дан массив b (n). Переписать в массив C(n) отрицательные элементы массива b(n) умноженные на 4 (со сжатием, без пустых элементов внутри). Затем упорядочить по возрастанию новый массив.

5. Создать динамические массивы, используя указатели. Дан массив b (n). Переписать в массив C(n) положительные элементы массива b(n) деленные на 5 (со сжатием, без пустых элементов внутри). Затем упорядочить по возрастанию новый массив.

6. Создать динамические массивы, используя указатели. Дан массив a (n). Переписать в массив b(n) только положительные элементы массива a, деленные на 3 (со сжатием, без пустых элементов внутри). Затем упорядочить по возрастанию новый массив.

7. Создать динамические массивы, используя указатели. Дан массив x (n). Переписать в массив y(n) отрицательные элементы массива x деленные на 2 (со сжатием, без пустых элементов внутри). Затем упорядочить по возрастанию новый массив.

8. Создать динамические массивы, используя указатели. Дан массив b (n). Переписать в массив C(n) отрицательные элементы массива

b(n). (со сжатием, без пустых элементов внутри). Затем упорядочить по возрастанию новый массив..

9. Создать динамические массивы, используя указатели. Дан массив с (n). Переписать в массив x (n) все ненулевые элементы массива умноженные на 4 (со сжатием, без пустых элементов внутри). Затем упорядочить по возрастанию новый массив.

10. Создать динамические массивы, используя указатели. Дан массив с (n). Переписать в массив x (n) все ненулевые элементы массива возведенные в квадрат (со сжатием, без пустых элементов внутри). Затем упорядочить методом по возрастанию новый массив.

11. Создать динамические массивы, используя указатели. Дан массив с (n). Переписать в массив x (n) все ненулевые элементы массива (со сжатием,

Без пустых элементов внутри) Затем упорядочить по возрастанию новый массив.

12. Создать динамические массивы, используя указатели. Дан массив с (n). Переписать в массив x ненулевые элементы массива с разделенные на 5 (со сжатием, без пустых элементов внутри). Затем упорядочить по возрастанию новый массив.

13. Дан массив b (n) . Переписать в массив C(n) корни квадратные из положительных элементов массива b(n) деленные на 5 (со сжатием, без пустых элементов внутри). Затем упорядочить по возрастанию новый массив.

14. Создать динамические массивы, используя указатели. Дан массив b (n). Переписать в массив C(n) корни квадратные из положительных элементов массива b(n) (со сжатием, без пустых элементов внутри) Затем упорядочить по возрастанию новый массив.

Контрольные вопросы к защите к лабораторной работе

1. Что такое указатель в C++ и для чего он используется?
2. Как связаны имя массива и указатель? Почему имя массива называется константным указателем?
3. Объясните смысл операций & (взятие адреса) и * (разыменование). Приведите примеры их использования.
4. Что такое «арифметика указателей»? Как выполняется операция сложения указателя с целым числом?
5. В чем разница между объявлениями int *p и int arr[10] с точки зрения работы с памятью и модификации?
6. Каковы цели и особенности использования операций new и delete?
7. В чем заключается разница между операторами delete и delete[]? Что произойдет, если их перепутать?
8. Как динамически выделить память для одномерного массива? Приведите пример.
9. Опишите процесс создания и удаления динамической матрицы (двумерного массива) с использованием указателей.
10. Что такое «нулевой указатель» (NULL, nullptr)? Для чего он нужен и почему рекомендуется его использовать?
11. Что может произойти, если не освободить динамически выделенную память после использования?
12. Как с помощью операции sizeof можно определить количество элементов в статическом массиве?
13. Почему после освобождения памяти с помощью delete рекомендуется присваивать указателю значение nullptr?

14. Что такое «висячий указатель» (dangling pointer) и как избежать его появления?
15. В чем разница между обычной формой операции new и формой new(std::nothrow)? Как обработать ошибку выделения памяти в каждом из случаев?
16. Как с помощью указателей организовать передачу массива в функцию?
17. Объясните, как работает обращение к элементу массива arr[i] с точки зрения работы с указателями.
18. Что такое «сжатие массива» и в каких практических задачах оно применяется?
19. Какие преимущества и недостатки у массива указателей по сравнению с обычным двумерным массивом?
20. Почему при работе с динамической памятью важно проверять, был ли указатель успешно выделен, перед его использованием?