ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ



МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ВЫСШАЯ ШКОЛА ПЕЧАТИ И МЕДИАИНДУСТРИИ

*Институт Принтмедиа и информационных технологий*

*Кафедра Информатики и информационных технологий*

направление подготовки

09.03.02 «Информационные системы и технологии»,

ЛАБОРАТОРНАЯ РАБОТА № 9

Дисциплина: ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Тема: Указатели и массивы

Выполнил(а): студент(ка) группы 181-722

Голиков Р.О.

(Фамилия И.О.)

Дата, подпись \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *\_\_\_\_\_\_\_\_\_\_\_*

(Дата) (Подпись)

Проверил: Ктн Доцент Арсентьев Д.А.*\_\_\_\_\_\_\_\_\_\_*

(Фамилия И.О. степень, звание) (Оценка)

Дата, подпись \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *\_\_\_\_\_\_\_\_\_\_\_*

(Дата) (Подпись)

Замечания: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва

2019

**Задание 1**

Напишите программу, которая принимает группу чисел от пользователя и помещает их в массив типа float. После того как числа будут помещены в массив, программа должна подсчитать их среднее арифметическое и вывести результат на дисплей. Используйте указатели везде, где только возможно.

**Код:**

#include <iostream>

using namespace std;

int main()

{

float flarr[100]; //array for numbers

char ch; //user decision

int num = 0; //counts numbers input

do

{

cout << "Enter number: "; //get numbers from user

cin >> \*(flarr + num++); //until user answers 'n'

cout << " Enter another (y/n)? ";

cin >> ch;

} while (ch != 'n');

float total = 0.0; //total starts at 0

for (int k = 0; k < num; k++) //add numbers to total

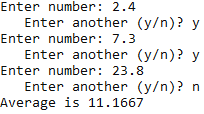
total += \*(flarr + k);

float average = total / num; //find and display average

cout << "Average is " << average << endl;

return 0;

}

**Вывод:**  


**Задание 2**

Используйте класс String, который приведён ниже. Добавьте к нему метод upit(), который будет переводить символы строки в верхний регистр. Вы можете использовать библиотечную функцию toupper(), которая принимает отдельный символ в качестве аргумента и возвращает символ, переведенный в верхний регистр (если это необходимо). Эта функция использует заголовочный файл Cctype. Добавьте в функцию main() необходимые строки для тестирования метода upit().

**Код:**

#include <iostream>

#include <cstring> //for strcpy(), etc

#include <cctype> //for toupper()

using namespace std;

class String //user-defined string type

{

private:

char\* str; //pointer to string

public:

String(const char\* s) //constructor, one arg

{

int length = strlen(s); //length of string argument

str = new char[length + 1]; //get memory

strcpy\_s(str, length \* sizeof(char) + sizeof(char), s); //copy argument to it

}

~String() //destructor

{

delete str;

}

void display() //display the String

{

cout << str;

}

void upit(); //uppercase the String

};

void String::upit() //uppercase each character

{

char\* ptrch = str; //pointer to this string

while (\*ptrch) //until null,

{

\*ptrch = toupper(\*ptrch); //uppercase each character

ptrch++; //move to next character

}

}

int main()

{

String s1 = "He who laughs last laughs best.";

cout << "\ns1="; //display string

s1.display();

s1.upit(); //uppercase string

cout << "\ns1="; //display string

s1.display();

cout << endl;

return 0;

}

**Вывод:**  


**Задание 3**

Используйте массив указателей на строки, представляющие собой названия дней недели, из примера ниже. Напишите функции для сортировки этих строк в алфавитном порядке, используя в качестве основы функции bsort() и order() из программы сортировки, представленной ниже. Сортировать необходимо указатели на строки, а не сами строки.

**Код:**

#include <iostream>

#include <cstring> //for strcmp(), etc.

using namespace std;

const int DAYS = 7; //number of pointers in array

int main()

{

void bsort(const char\*\*, int); //prototype

//array of pointers to char

const char\* arrptrs[DAYS] = { "Sunday", "Monday", "Tuesday",

"Wednesday", "Thursday",

"Friday", "Saturday" };

cout << "\nUnsorted:\n";

for (int j = 0; j < DAYS; j++) //display unsorted strings

cout << \*(arrptrs + j) << endl;

bsort(arrptrs, DAYS); //sort the strings

cout << "\nSorted:\n";

for (int j = 0; j < DAYS; j++) //display sorted strings

cout << \*(arrptrs + j) << endl;

return 0;

}

void bsort(const char\*\* pp, int n) //sort pointers to strings

{

void order(const char\*\*, const char\*\*); //prototype

int j, k; //indexes to array

for (j = 0; j < n - 1; j++) //outer loop

for (k = j + 1; k < n; k++) //inner loop starts at outer

order(pp + j, pp + k); //order the pointer contents

}

void order(const char\*\* pp1, const char\*\* pp2) //orders two pointers

{ //if string in 1st is

if (strcmp(\*pp1, \*pp2) > 0) //larger than in 2nd,

{

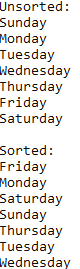
const char\* tempptr = \*pp1; //swap the pointers

\*pp1 = \*pp2;

\*pp2 = tempptr;

}

}

**Вывод:**  


**Задание 4**

Добавьте деструктор в программу связного списка. Он должен удалять все элементы списка при удалении объекта класса linklist. Элементы должны удаляться по очереди, в соответствии с их расположением в списке. Протестируйте деструктор путем вывода сообщения об удалении каждого из элементов списка; удалено должно быть также количество элементов, какое было положено в список (деструктор вызывается автоматически для каждого существующего объекта).

**Код:**

#include <iostream>

using namespace std;

struct link //one element of list

{

int data; //data item

link\* next; //pointer to next link

};

class linklist //a list of links

{

private:

link\* first; //pointer to first link

public:

linklist() //no-argument constructor

{

first = NULL;

} //no first link

~linklist(); //destructor

void additem(int d); //add data item (one link)

void display(); //display all links

};

void linklist::additem(int d) //add data item

{

link\* newlink = new link; //make a new link

newlink->data = d; //give it data

newlink->next = first; //it points to next link

first = newlink; //now first points to this

}

void linklist::display() //display all links

{

link\* current = first; //set ptr to first link

while (current != NULL) //quit on last link

{

cout << " "<< current->data; //print data

current = current->next; //move to next link

}

}

linklist::~linklist() //destructor

{

link\* current = first; //set ptr to first link

while (current != NULL) //quit on last link

{

link\* temp = current; //save ptr to this link

current = current->next; //get ptr to next link

cout << temp->data << " ";

delete temp; //delete this link

}

}

int main()

{

linklist li; //make linked list

li.additem(25); //add four items to list

li.additem(36);

li.additem(49);

li.additem(64);

li.display(); //display entire list

cout << endl;

return 0;

}

**Вывод:**  
****

**Задание 5**

Предположим, что в функции main() определены три локальных массива одинакового размера и типа (скажем, float). Первые два уже инициализированы значениями. Напишите функцию addarrays(), которая принимает в качестве аргументов адреса трех массивов, складывает соответствующие элементы двух массивов и помещает результат в третий массив. Четвертым аргументом этой функции может быть размерность массивов. На всем протяжении программы используйте указатели.

**Код:**

#include <iostream>

#include <iomanip>

using namespace std;

const int SIZE = 5;

void addarrays(float \*arr1, float \*arr2, float \*arr3, int size)

{

for (int i = 0; i < size; i++)

\*(arr3 + i) = \*(arr1 + i) + \*(arr2 + i);

}

int main()

{

float arr1[SIZE], arr2[SIZE], arr3[SIZE];

for (int i = 0; i < SIZE; i++)

{

\*(arr1 + i) = i \* i - i \* 2.4;

cout << setw(8) << \*(arr1 + i) << " ";

}

cout << endl;

for (int i = 0; i < SIZE; i++)

{

\*(arr2 + i) = i \* 5 / 2.3;

cout << setw(8) << \*(arr2 + i) << " ";

}

cout << endl;

addarrays(arr1, arr2, arr3, SIZE);

for (int i = 0; i < SIZE; i++)

cout << setw(8) << \*(arr3 + i) << " ";

return 0;

}

**Вывод:**  


**Задание 6**

Создайте свою версию библиотечной функции strcmp(s1, s2), которая сравнивает две строки и возвращает -1, если s1 идет первой по алфавиту, О, если в s1 и s2 одинаковые значения, и 1, если s2 идет первой по алфавиту. Назовите вашу функцию compstr(). Она должна принимать в качестве аргументов два указателя на строки char \*, сравнивать эти строки посимвольно и возвращать число int. Напишите функцию main() для проверки работы вашей функции с разными строками. Используйте указатели во всех возможных ситуациях.

**Код:**

#include <iostream>

using namespace std;

int strcmp(const char \*str1, const char \*str2)

{

for (int i = 0; i < strlen(str1); i++)

{

if(\*(str1 + i) == \*(str2 + i)) continue;

return \*(str1 + i) > \*(str2 + i) ? 1 : -1;

}

return 0;

}

int main()

{

const char \*ch1 = "String1";

const char \*ch2 = "String2";

const char \*ch3 = "AString";

const char \*ch4 = "astring";

const char \*ch5 = "astring";

cout << strcmp(ch1, ch2) << endl;

cout << strcmp(ch1, ch3) << endl;

cout << strcmp(ch3, ch4) << endl;

cout << strcmp(ch4, ch5) << endl;

return 0;

}

**Вывод:**  


**Задание 7**

Модифицируйте класс person из программы, представленной ниже, чтобы он включал в себя не только имя человека, но и сведения о его зарплате в виде поля salary типа float. Вам будет необходимо изменить методы setName() и printName() на setData() и printData(), включив в них возможность ввода и вывода значения salary, как это можно сделать с именем. Вам также понадобится метод getSalary(). Используя указатели, напишите функцию salsortQ, которая сортирует указатели массива persPtr по значениям зарплаты. Попробуйте вместить всю сортировку в функцию salsort(), не вызывая других функций, как это сделано в программе PERS0RT.

**Код:**

#include <iostream>

#include <string>

using namespace std;

class person

{

protected:

string name;

float salary;

public:

void setData()

{

cout << "Enter name: "; cin >> name;

cout << "Enter salary: "; cin >> salary;

}

void printData()

{

cout << endl << name << " " << salary;

}

string getName()

{

return name;

}

float getSalary()

{

return salary;

}

};

void salsort(person \*\*pp, int n)

{

for (int i = 0; i < n; i++)

for (int j = 0; j < n - i - 1; j++)

if ((\*(pp + j))->getSalary() > (\*(pp + j + 1))->getSalary())

{

person\* temp = \*(pp + j);

\*(pp + j) = \*(pp + j + 1);

\*(pp + j + 1) = temp;

}

}

int main()

{

person \*persPtr[100];

int n = 0;

char choice;

do {

persPtr[n] = new person;

persPtr[n]->setData();

n++;

cout << "Enter another (y/n)? ";

cin >> choice;

} while (choice == 'y');

cout << "\nUnsorted list:";

for (int j = 0; j < n; j++)

{

persPtr[j]->printData();

}

salsort(persPtr, n);

cout << "\nSorted list:";

for (int j = 0; j < n; j++)

{

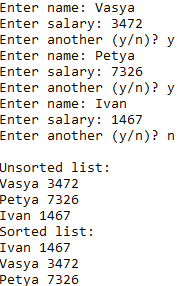
persPtr[j]->printData();

}

cout << endl;

return 0;

}

**Вывод:**  


**Задание 8**

Исправьте функцию additem() из программы связного списка так, чтобы она добавляла новый элемент в конец списка, а не в начало. Для того чтобы добавить элемент, вам необходимо будет пройти по цепи до конца списка, а затем изменить указатель последнего элемента так, чтобы он указывал на новый элемент.

**Код:**

#include <iostream>

using namespace std;

struct link

{

int data;

link\* next;

};

class linklist

{

private:

link\* first;

public:

linklist()

{

first = NULL;

}

void additem(int d);

void display();

};

void linklist::additem(int d)

{

link\* newlink = new link;

newlink->data = d;

newlink->next = NULL;

if (first != NULL)

{

link\* current = first;

while (current->next != NULL)

{

current = current->next;

}

current->next = newlink;

}

else first = newlink;

}

void linklist::display()

{

link\* current = first;

while (current != NULL)

{

cout << current->data << endl;

current = current->next;

}

}

int main()

{

linklist li;

li.additem(25);

li.additem(36);

li.additem(49);

li.additem(64);

li.display();

return 0;

}

**Вывод:**  


**Задание 9**

Допустим, что нам нужно сохранить 100 целых чисел так, чтобы иметь к ним легкий доступ. Допустим, что при этом у нас есть проблема: память нашего компьютера так фрагментирована, что может хранить массив, наибольшее количество элементов в котором равно десяти (такие проблемы действительно появляются, хотя обычно это происходит с объектами, занимающими большое количество памяти). Вы можете решить эту проблему, определив 10 разных массивов по 10 элементов в каждом и массив из 10 указателей на эти массивы. Массивы будут иметь имена а0, a1, а2 и т. д. Адрес каждого массива будет сохранен в массиве указателей типа int\*, который называется ар. Вы сможете получить доступ к отдельному целому используя выражение ap[j] [к], где j является номером элемента массива указателей, а к — номером элемента в массиве, на который этот указатель указывает. Это похоже на двумерный массив, но в действительности является группой одномерных массивов. Заполните группу массивов тестовыми данными (скажем, номерами 0, 10, 20 и т. д.), а затем выведите их, чтобы убедиться, что все работает правильно.

**Код:**

#include <iostream>

#include <iomanip>

using namespace std;

int main()

{

int a1[10], a2[10], a3[10], a4[10], a5[10];

int a6[10], a7[10], a8[10], a9[10], a10[10];

int \*ap[] = { a1, a2, a3, a4, a5, a6, a7, a8, a9, a10};

for (int i = 0; i < 10; i++)

for (int j = 0; j < 10; j++)

ap[i][j] = i \* j;

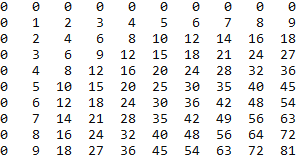
for (int i = 0; i < 10; i++, cout << endl)

for (int j = 0; j < 10; j++, cout << " ")

cout << setw(4) << ap[i][j] ;

return 0;

}

**Вывод:**  


**Задание 10**

Описанный в упражнении 9 подход нерационален, так как каждый из 10 массивов объявляется отдельно, с использованием отдельного имени, и каждый адрес получают отдельно. Вы можете упростить программу, используя операцию new, которая позволит вам выделить память для массивов в цикле и одновременно связать с ними указатели. Перепишите программу упражнения 9, используя этот подход. Доступ к отдельному элементу массивов вы сможете получить, используя то же выражение, что и в упражнении 9, или аналогичное выражение с указателями.

**Код:**

#include <iostream>

#include <iomanip>

using namespace std;

int main()

{

int \*ap[10];

for (int i = 0; i < 10; i++)

{

\*(ap + i) = new int[10];

for (int j = 0; j < 10; j++)

\*(\*(ap + i) + j) = i \* j;

}

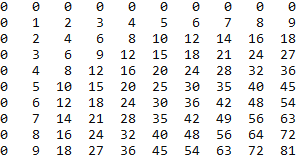
for (int i = 0; i < 10; i++, cout << endl)

for (int j = 0; j < 10; j++, cout << " ")

cout << setw(4) << \*(\*(ap + i) + j);

return 0;

}

**Вывод:**  


**Задание 11**

Создайте класс, который позволит вам использовать 10 отдельных массивов из упражнения 10 как один одномерный массив, допуская применение операций массива. То есть мы можем получить доступ к элементам массива, записав в функции main() выражение типа a[j], а методы класса могут получить доступ к полям класса, используя двухшаговый подход. Перегрузим операцию [ ], чтобы получить нужный нам результат. Заполним массив данными и выведем их. Хотя для интерфейса класса использованы операции индексации массива, вам следует использовать указатели внутри методов класса.

**Код:**

#include <iostream>

#include <iomanip>

using namespace std;

class segarray

{

private:

int \*ptr[10];

public:

segarray()

{

for (int i = 0; i < 10; i++)

\*(ptr + i) = new int[10];

}

int& operator[](int index)

{

return \*(\*(ptr + index / 10) + index % 10);

}

};

int main()

{

segarray arr;

for (int i = 0; i < 100; i++)

{

arr[i] = i;

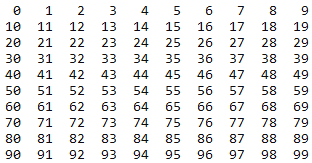
}

for (int i = 0; i < 100; i++)

cout << (i % 10 == 0 ? "\n" : " ") << setw(3) << arr[i];

return 0;

}

**Вывод:**  


**Задание 12**

Указатели сложны, поэтому давайте посмотрим, сможем ли мы сделать работу с ними более понятной (или, возможно, более непонятной), используя их симуляцию в классе.

Для разъяснения действия наших доморощенных указателей мы смоделируем память компьютера с помощью массивов. Так как доступ к массивам всем понятен, то вы сможете увидеть, что реально происходит, когда мы используем для доступа к памяти указатели.

Мы будем использовать один массив типа char для хранения всех типов переменных. Именно так устроена память компьютера: массив байтов (тип char имеет тот же размер), каждый из которых имеет адрес (или, в терминах массива, индекс). Однако C++ не позволит нам хранить данные типа float или int в массиве типа char обычным путем (мы можем использовать объединения, но это другая история). Поэтому мы создадим симулятор памяти, используя отдельный массив для каждого типа данных, которые мы хотим сохранить. В этом упражнении мы ограничимся одним типом float, и нам понадобится массив для него. Назовем этот массив fmemory. Однако значения указателей (адреса) тоже хранятся в памяти, и нам понадобится еще один массив для их хранения. Так как в качестве модели адресов мы используем индексы массива, то нам потребуется массив типа int, назовем его pmemory, для хранения этих индексов.

Индекс массива fmemory (назовем его fmem\_top) показывает на следующее по очереди доступное место, где можно сохранить значение типа float. У нас есть еще похожий индекс массива pmemory (назовем его pmem\_ top). Не волнуйтесь о том, что наша «память» может закончиться. Мы предполагаем, что эти массивы достаточно большие, чтобы хранить все, что мы захотим, и нам не надо заботиться об управлении памятью.

Создадим класс Float, который мы будем использовать для моделирования чисел типа float, которые будет храниться в fmemory вместо настоящей памяти. Класс Float содержит поле, значением которого является индекс массива fmemory, хранящего значения типа float. Назовем это поле addr. В классе также должны быть два метода. Первый — это конструктор, имеющий один аргумент типа float для инициализации значения. Конструктор помещает значение аргумента в элемент массива fmemory, на который указывает указатель fmem\_top, а затем записывает значение fmem\_top в массив addr. Это похоже на то, как компоновщик и компилятор хранят обычные переменные в памяти. Второй метод является перегружаемой операцией &. Он просто возвращает значение указателя (индекса типа int) в addr.

Создадим второй класс ptrFloat. Объект этого класса содержит адрес (индекс) в pmemory. Метод класса инициализирует этот «указатель» значением типа int. Второй метод перегружает операцию \* (операция разыменования). Его действия более сложны. Он получает адрес из массива pmemory, в котором хранятся адреса. Затем полученный адрес используется как индекс массива fmemory для получения значения типа float, которое располагалось по нужному нам адресу.

Таким образом мы моделируем действия операции разыменования (\*). Заметим, что вам нужно возвращаться из этой функции по ссылке, чтобы можно было использовать операцию \* слева от знака равно.

Классы Float и ptrFloat похожи, но класс Float хранит данные типа float в массиве, представляющем собой память, а класс ptrFloat хранит поля типа int (являющиеся у нас указателями, но на самом деле индексами массива) в другом массиве, который тоже представляет собой память.

**Код:**

#include <iostream>

using namespace std;

const int MAX = 100;

float fmemory[MAX];

int fmem\_top = 0;

int pmemory[MAX];

int pmem\_top = 0;

class Float

{

private:

int addr;

public:

Float(float value)

{

addr = fmem\_top;

fmemory[fmem\_top++] = value;

}

int operator&()

{

return addr;

}

};

class ptrFloat

{

private:

int addr;

public:

ptrFloat(int ptr)

{

addr = pmem\_top;

pmemory[pmem\_top++] = ptr;

}

float& operator\*()

{

return fmemory[pmemory[addr]];

}

};

int main()

{

Float varl = -1.234f; // определяем и инициализируем

Float var2 = 5.678f; // две вещественные переменные

ptrFloat ptrl = &varl; // определяем и инициализируем

ptrFloat ptr2 = &var2; // два указателя

cout << "\*ptrl = " << \*ptrl << endl; // получаем значения переменных

cout << "\*ptr2 = " << \*ptr2 << endl; // и выводим на экран

\*ptrl = 7.123f; // присваиваем новые значения

\*ptr2 = 8.456f; // переменным, адресованным через указатели

cout << "\*ptrl = " << \*ptrl << endl; // снова получаем значения

cout << "\*ptr2 = " << \*ptr2 << endl; // и выводим на экран

return 0;

}

**Вывод:**

