МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
Национальный университет кораблестроения
имени адмирала Макарова

**С. Б. ПРИХОДЬКО, Л. Н. МАКАРОВА,
Т. Г. СМЫКОДУБ**

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
к выполнению лабораторных работ по дисциплине "Основы программирования"

В двух частях

Часть 2

Рекомендовано Методичною радою НУК

Электронное издание
комбинированного использования на DVD-ROM

НИКОЛАЕВ • НУК • 2018

УДК 004.4(076)
П 77

Автори:
С. Б. Приходько, д-р техн. наук, професор;
Л. М. Макарова, канд. техн. наук;
Т. Г. Смикодуб, ст. викладач

Рецензент І. І. Коваленко, д-р техн. наук, професор

Приходько С. Б.

П 77 Методичні вказівки до виконання лабораторних робіт з дисципліни "Основи програмування" : у 2 ч. Ч. 2 / С. Б. Приходько, Л. М. Макарова, Т. Г. Смикодуб. – Миколаїв : НУК, 2018. – 139 с.

Наведені завдання для лабораторних робіт, стислий виклад теоретичних відомостей, етапи виконання робіт і контрольні питання.

Методичні вказівки призначені для студентів 1 курсу спеціальності 121 "Інженерія програмного забезпечення" (попередня назва "Програмне забезпечення систем"), а також для всіх, хто вивчає основи програмування на мові С++.

УДК 004.4(076)

ПРИХОДЬКО Сергій Борисович
МАКАРОВА Лідія Миколаївна
СМИКОДУБ Тетяна Георгіївна

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт з дисципліни "Основи програмування"

У двох частинах

Частина 2

(*російською мовою*)

Комп'ютерне верстання *А. В. Платонова*
Коректор *М. О. Паненко*

Формат 60×84/16. Ум. друк. арк. 8,1. Об'єм даних 4906 кб.
Тираж 15 прим. Вид. № 15. Зам. № 68.

Видавець і виготівник Національний університет кораблебудування
імені адмірала Макарова
просп. Героїв України, 9, м. Миколаїв, 54025
E-mail : publishing@nuos.edu.ua

Свідоцтво суб'єкта видавничої справи ДК № 2506 від 25.05.2006 р.

ВВЕДЕНИЕ

В Национальном университете кораблестроения имени адмирала Макарова (НУК) дисциплина "Основы программирования" изучается студентами специальности 121 "Инженерия программного обеспечения" (предыдущее название "Программное обеспечение систем") в первом–третьем семестрах. Она относится к циклу профессиональной и практической подготовки бакалавров.

Эти методические указания должны помочь студентам первого курса специальности 121 "Инженерия программного обеспечения" в подготовке и выполнении лабораторных работ по дисциплине "Основы программирования". Они содержат задания для лабораторных работ, краткое изложение теоретических сведений, этапы выполнения работ и контрольные вопросы.

При выполнении каждой лабораторной работы рекомендуется:

– усвоить теоретические сведения;

– разобраться с заданием и этапами выполнения работы;

– выполнить предложенные задания и оформить отчет по работе;

– проверить полноту понимания темы с помощью контрольных вопросов, расположенных в конце каждой работы.

Лабораторную работу необходимо выполнять в соответствии с проведенными этапами выполнения работы. Отчет о лабораторной работе оформляется каждым студентом индивидуально. Отчет должен содержать: название и цель работы; задание (постановку задачи); методику и алгоритм решения задачи; текст программы; результаты работы; анализ результатов и выводы.

РАБОТА № 1
Разработка и реализация программ поиска и сортировки

Цель работы: овладение навыками разработки программ поиска и сортировки элементов массива и выполнение их в NetBeans IDE.

Задания

Задание 1.1. Сформулировать математическую запись фрагмента программы и вычислить значение переменной `x` после выполнения программы, согласно номеру варианта, указанного в таблице 1.1. Элементы массива исчисляются по формуле `a[i]=p[i]-50,` где `p[i+1]=(p[i]*11+7)%100.` `p[0]` равняется `N` – номеру варианта по списку группы, количество элементов в массиве равняется `size=15`.

Таблица 1.1 – Варианты задания 1.1

№	Фрагмент программы

1–5	`for (int i=0; i< size; i++) {`
	`for (int j= size-1; j>i; j--){`
	`if (a[j] < a[j-1]) {`
	`int t = a[j];`
	`a[j] = a[j-1];`
	`a[j-1] = t;`
	`}`
	`}`
	`}`
	`x=a[0];`

Продолж. табл. 1.1

№	Фрагмент программы

6–10	`for (int i=1; i<size; i++) {`
	`  int curr = a[i];`
	`  int j = i-1;`
	`  while (j>=0 && a[j]>curr) {`
	`  a[j+1]=a[j];`
	`  j--;`
	`  }`
	`  a[j+1] = curr;`
	`  }`
	`x=a[0];`

11–15	`  for (int i=0; i<size-1; i++) {`
	`  int nmin = i;`
	`  for (int j=i+1; j<size; j++){`
	`  if (a[j]<a[nmin]) nmin = j;`
	`  }`
	`  if (i != nmin) {`
	`  int t = a[nmin];`
	`  a[nmin] = a[i];`
	`  a[i] = t;`
	`  }`
	`  }`
	`x=a[0];`

16–20	`for (int i=0; i< size; i+=2){`
	`  for (int j= size-1; j>i; j-=2){`
	`  if (a[j] > a[j-2] ) {`
	`  int t = a[j];`
	`  a[j] = a[j-2];`
	`  a[j-2] = t;`
	`  }`
	`  }`
	`  }`
	`x=a[2];`

21–25	`for (int i=0; i<size-1; i++) {`
	`  int nmin = i;`
	`  for (int j=i+1; j<size; j++){`
	`  if (a[j]>a[nmin]) nmin = j;`
	`  }`

Продолж. табл. 1.1

№	Фрагмент программы

21–25	`if (i != nmin) {`
	`int t = a[nmin];`
	`a[nmin] = a[i];`
	`a[i] = t;`
	`}`
	`}`
	`x=a[0];`

26–30	`for (int i=1; i<size; i+=2) {`
	`int curr = a[i];`
	`int j = i-2;`
	`while (j>=0 && a[j]>curr) {`
	`a[j+2]=a[j];`
	`j-=2;`
	`}`
	`a[j+2] = curr;`
	`}`
	`x=a[1];`

Задание 1.2. Составить программу сортировки последовательностей чисел согласно номеру варианта, которые приведены в таблице 1.2, и выполнить ее в NetBeans IDE. Последовательности заполнить с помощью датчика случайных чисел.

Таблица 1.2 – Варианты задания 1.2

№	Задание
1	Дана последовательность целых чисел $a_1, a_2, \ldots, a_{25}$. Расположить элементы последовательности, кратные 3, по возрастанию (остальные элементы остаются на своих местах). Метод сортировки – выбор
2	Дана последовательность действительных чисел $a_1, a_2, \ldots, a_{22}$. Расположить положительные элементы последовательности, которые стоят на нечетных местах по возрастанию (остальные элементы остаются на своих местах). Метод сортировки – выбор
3	Дана последовательность действительных чисел $a_1, a_2, \ldots, a_{22}$. Элементы, которые стоят на четных местах, расположить в порядке возрастания, а на нечетных – в порядке убывания. Метод сортировки – пузырьком

Продолж. табл. 1.2

№	Задание
4	Дана последовательность действительных чисел $a_1, a_2, \ldots, a_{22}$. Расположить элементы последовательности, которые стоят на четных местах в порядке убывания (остальные элементы остаются на своих местах). Метод сортировки – пузырьком
5	Дана последовательность действительных чисел $a_1, a_2, \ldots, a_{22}$. Расположить элементы последовательности, меньшие среднеарифметического значения, по возрастанию (остальные элементы остаются на своих местах). Метод сортировки – вставка
6	Дана последовательность действительных чисел $a_1, a_2, \ldots, a_{22}$, отличных от нуля. Расположить отрицательные элементы последовательности по убыванию, а положительные – по возрастанию. Метод сортировки – выбор
7	Дана последовательность положительных целых чисел $a_1, a_2, \ldots, a_{25}$. Расположить элементы последовательности, кратные 3, поубыванию (остальные элементы остаются на своих местах). Метод сортировки – вставка
8	Дана последовательность целых чисел $a_1, a_2, \ldots, a_{25}$. Расположить четные элементы по возрастанию (остальные элементы остаются на своих местах). Метод сортировки – выбор
9	Дана последовательность действительных чисел $a_1, a_2, \ldots, a_{22}$. Нужно расположить положительные элементы последовательности по убыванию (другие элементы остаются на своих местах). Метод сортировки – вставка
10	Дана последовательность целых чисел $a_1, a_2, \ldots, a_{25}$. Расположить нечетные элементы по убыванию (остальные элементы остаются на своих местах). Метод сортировки – выбор
11	Дана последовательность действительных чисел $a_1, a_2, \ldots, a_{22}$. Расположить элементы последовательности, которые принадлежат диапазону $[p_1, p_2]$, по убыванию (остальные элементы остаются на своих местах). Метод сортировки – пузырьком
12	Дана последовательность действительных чисел $a_1, a_2, \ldots, a_{30}$. Расположить первую половину элементов последовательностипо убыванию, остальные элементы расположить по возрастанию. Метод сортировки – пузырьком
13	Дана последовательность действительных чисел $a_1, a_2, \ldots, a_{22}$. Расположить элементы последовательности, которые содержат цифру N, по возрастанию (остальные элементы остаются на своих местах). Метод сортировки – вставка

Продолж. табл. 1.2

№	Задание
14	Дана последовательность действительных чисел $a_1, a_2, \ldots, a_{20}$. Расположить элементы последовательности, сумма цифр которых равняется N, по убыванию (остальные элементы остаются на своих местах). Метод сортировки – выбор
15	Дана последовательность целых чисел $a_1, a_2, \ldots, a_{30}$. Расположить положительные элементы последовательности, кратные 5, по убыванию (остальные элементы остаются на своих местах). Метод сортировки – выбор

Краткие теоретические сведения

Наиболее простой из способов поиска данных – *линейный поиск*. Данный алгоритм сравнивает каждый элемент массива с ключом, предоставленным для поиска. Алгоритм линейного поиска отлично работает только для небольших или неупорядоченных массивов и является абсолютно надежным.

Если массив содержит упорядоченную последовательность данных, то более эффективен *двоичный (бинарный) поиск*.

Предположим, что переменные `Lb` и `Rb` содержат, соответственно, левую и правую границы отрезка массива, где находится нужный элемент. Поиск всегда будет начинаться с анализа среднего элемента `M` отрезка массива. Если искомое значение меньше `M`, переходим к поиску в левой половине отрезка, где все элементы меньше только что проверенного. Другими словами, значением `Rb` становится (`M-1`) и на следующей итерации работа ведется с половиной массива. Таким образом, в результате каждой проверки вдвое сужается область поиска.

Двоичный поиск – очень мощный метод, например, если длина массива равна 1023, то после первого сравнения область сужается до 511 элементов, а после второго – до 255,

т. е. для поиска в массиве из 1023 элементов достаточно 10 сравнений.

Идея метода *пузырьковой сортировки* состоит в следующем: шаг сортировки заключается в проходе снизу вверх по массиву. По пути просматриваются пары соседних элементов. Если элементы некоторой пары находятся в неправильном порядке, то они меняются местами. После первого прохода по массиву "вверху" оказывается самый "легкий" элемент. Следующий проход делается до второго сверху элемента, таким образом второй по величине элемент поднимается на правильную позицию. Проходы делаются по все уменьшающейся нижней части массива до тех пор, пока в ней не останется только один элемент. На этом сортировка заканчивается, так как последовательность упорядочена по возрастанию.

Основные принципы метода: среднее число сравнений и обменов имеют квадратичный порядок роста, отсюда можно заключить, что алгоритм пузырька очень медленен и малоэффективен. Тем не менее, у него есть громадный плюс: он прост в понимании и имеет множество модификаций. Например, если при прохождении элементов массива не произошло ни одного обмена, это значит, что все пары расположены в правильном порядке, т. е. массив уже отсортирован. Таким образом, первый шаг оптимизации заключается в запоминании, производился ли на данном проходе какой-либо обмен. Если нет – алгоритм заканчивает работу.

Идея метода *сортировки выбором* состоит в том, чтобы создавать отсортированную последовательность путем присоединения к ней одного элемента за другим в правильном порядке. Алгоритм состоит из `n` последовательных шагов, начиная от нулевого и заканчивая (`n-1`). На *i*-м шаге выбираем наименьший из элементов `a[i] ... a[n-1]` и меняем его местами с `a[i]`. Вне зависимости от номера текущего шага *i*, последовательность `a[0] ... a[i]` является упоря-

доченной. Таким образом, на шаге (n-1) вся последовательность, кроме a[n-1], оказывается отсортированной, а a[n-1] стоит на последнем месте по праву: все меньшие элементы уже ушли влево.

Основная идея метода: для нахождения наименьшего элемента из n+1 алгоритм совершает n сравнений. Поскольку число обменов всегда будет меньше числа сравнений, время сортировки возрастает относительно количества элементов. Кроме того, алгоритм не использует дополнительной памяти, все операции происходят "на месте".

Сортировку этим методом можно использовать для массивов, имеющих небольшие размеры.

Сортировка простыми вставками в чем-то похожа на методы, изложенные ранее. Аналогичным образом делаются проходы по части массива, и аналогичным же образом в его начале "вырастает" отсортированная последовательность.

Однако в сортировке пузырьком или выбором можно было четко заявить, что на i-м шаге элементы a[0] ... a[i] стоят на правильных местах и никуда более не переместятся. Здесь же подобное утверждение будет более слабым: последовательность a[0] ... a[i] упорядочена. При этом по ходу алгоритма в нее будут вставляться все новые элементы.

Рассмотрим действия алгоритма на i-м шаге. Пусть последовательность к этому моменту разделена на две части: упорядоченную a[0] ... a[i] и неупорядоченную a[i+1]...a[n-1]. На следующем, (i+1)-м каждом шаге алгоритма берем a[i+1] и вставляем на нужное место в готовую часть массива. Поиск подходящего места для очередного элемента входной последовательности осуществляется путем последовательных сравнений с элементом, стоящим перед ним. В зависимости от результата сравнения элемент либо остается на текущем месте (вставка завершена), либо они меняются местами и процесс повторяется. Таким образом,

в процессе вставки мы "просеиваем" элемент x к началу массива, останавливаясь в случае, когда найден элемент, меньший x или достигнуто начало последовательности.

Хорошим показателем сортировки является весьма естественное поведение: почти отсортированный массив будет досортирован очень быстро. Это, вкупе с устойчивостью алгоритма, делает метод хорошим выбором в соответствующих ситуациях.

Пример выполнения работы

Задание 1.1. Сформулировать математическую запись фрагмента программы и вычислить значение переменной x после его выполнения, где size=22 – размер массива, N=31 – номер варианта по списку группы.

```
for (int i=0; i<size-1; i++) {
      if(a[i]%2==0)
      {int nmin = i;
        for (int j=i+1; j<size; j++){
          if (a[j]<a[nmin]&& (a[j]%2==0))
               nmin = j;
        }
        if (i != nmin) {
          int t = a[nmin];
          a[nmin] = a[i];
          a[i] = t;
        }
     }
       }
x=a[3];
```

Решение

Данный фрагмент программы выполняет сортировку четных элементов массива по возрастанию. Реализация сортировки выполнена методом выбора, переменная x=−38.

Задание 1.2. Дана последовательность целых чисел a_1, a_2, ... , a_{22}. Расположить элементы последовательности, кратные 5, по возрастанию (остальные элементы остаются на своих местах). Сортировку выполнить методом выбора.

Решение

1. Постановка задачи

Дана последовательность целых чисел a_1, a_2, ... , a_{22}. Отсортировать по возрастанию только элементы, кратные 5.

2. Алгоритм решения задачи

Алгоритм решения задачи можно представить в виде такой последовательности действий:

Действие 1. Ввести элементы одномерного массива *a*.

Действие 2. Отсортировать элементы, кратные 5.

Действие 3. Вывести упорядоченный массив на экран.

3. Текст программы

```
#include <iostream>
#include <windows.h>
using namespace std;
void chooseSort(int a[], int size){
for (int i=0; i<size-1; i++) {
      if(a[i]%5==0)
      {int nmin = i;
         for (int j=i+1; j<size; j++){
            if (a[j]<a[nmin]&& (a[j]%5==0))
               nmin = j;
        }
        if (i != nmin) {
             int t = a[nmin];
             a[nmin] = a[i];
             a[i] = t;
         }
    }
```

```
        }
    }
    void printArr(int a[], int size){
         for(int i=0; i<size; i++) {
             cout << a[i] << " ";
         }
         cout << "\n";
    }
    int main(){
      SetConsoleCP(1251);SetConsoleOutputCP(1251);
        int p[22], arr[22],n=22;
        p[0]=31;
        for(int i=0;i<n-1;i++)
             p[i+1]=(p[i]*11 + 7) % 100;
        for(int i=0;i<n;i++)
             arr[i]=p[i]-50;
       cout<<"До сортировки:\n";
         printArr(arr, n);
         chooseSort(arr, n);
       cout<<" После сортировки:\n";
         printArr(arr, n);
         return 0;
    }
```

4. Результаты работы программы

```
До сортировки:
-19 -2 -15 42 -31 -34 33 -30 -23 -46 1 18
5 -38 -11 -14 -47 -10 -3 -26 21 38
После сортировки:
-19 -2 -30 42 -31 -34 33 -15 -23 -46 1 18
-10 -38 -11 -14 -47 5 -3 -26 21 38
```

Контрольные вопросы

1. Какой основной принцип работы алгоритма линейного поиска?
2. К какому массиву можно применять алгоритм бинарного поиска?
3. Какой основной принцип работы алгоритма бинарного поиска?
4. Какой основной принцип работы алгоритма метода пузырька?
5. Как работает алгоритм сортировки выбором?
6. Как работает алгоритм сортировки простыми вставками?

РАБОТА № 2
Разработка и реализация программ для работы с указателями и одномерными динамическими массивами

Цель работы: овладение навыками разработки программ для работы с указателями и одномерными динамическими массивами и выполнение их в NetBeans IDE.

Задания

Задание 2.1. Пусть дан следующий фрагмент программы, представленный в вариантах таблицы 2.1. Объясните, как изменится массив после его выполнения, если программа начинается следующим образом (Указание: вместо N подставить номер варианта по списку группы).

```
#include <iostream>
#include <cmath>
using namespace std;
int* form(int &n) {   n=10+N%10;
 int *a=new int[n];
 *a=N;
 for(int i=1;i<n;i++)
     *(a+i)=*(a+i-1)+1;
 return a;
}
int main(){
```

Таблица 2.1 – Варианты задания 2.1

№	Фрагмент программы

1–5	`int *a, n;  a=form(n);`
	`int **p=new int *[n];`
	`for (inti=0;i<n;i++)`
	`{ p[i]=(a+i);`
	`*p[i]=*p[i]+N;`
	`}`

6–10	`int *a, n,i,*p;`
	`a=form(n);`
	`for (p = &a[0], i = 0 ; i<n; i++ )`
	`{`
	`*(p+i) -=N*N;`
	`}`

11–15	`int *a, n,i,*p;`
	`a=form(n);`
	`for ( p = a, i = 0; p+i<a+n; i++ ){`
	`*(p+i) *=*(p+i);`
	`}`

16–20	`int *a, n,i,*p;`
	`a=form(n);`
	`for ( p = a+n, i=1; i<=n; i++ )`
	`{`
	`*(p-i) *= (int)sqrt(N);`
	`}`

21–25	`int *a, n,i,*aa;  a=form(n);`
	`int **p=new int *[n];`
	`aa=a+n-1;`
	`for (i=0;aa-a>=0;aa--,i++)`
	`{ *(p+i)=aa;`
	`**(p+i)+=*(a+i);`
	`}`

26–30	`int *a, n,i,t=0,*p; a=form(n);`
	`int *aa=new int[n]; p=aa;`
	`for (i=n-1;i>=0;i--)`
	`{*p++=*(a+i); t+=*(a+i);}`
	`t=t/n;`
	`for (i=0;i<n;i++)`
	`*(a+i)=*(aa+i)+t;`
	`}`

Задание 2.2. Составить программу, которая заполняет динамический одномерный массив с помощью датчика случайных чисел, потом изменяет его содержимое согласно заданию, которое приведено в таблице 2.2 и выполнить ее в NetBeans IDE.

Примечание:

a. Вывести массив до и после модификации.

b. При обращении к элементам массива использовать указатели.

Таблица 2.2 – Варианты задания 2.2

№	Задание
1	Удалить все отрицательные двузначные элементы массива
2	Удалить элемент с заданным ключом (значением) и элемент с заданным номером
3	Удалить все элементы, расположенные между первым четным элементом и элементом *p*, где *p* – это целая часть от среднего арифметического значения элементов массива
4	Удалить N элементов, начиная с номера K, где N и K вводятся с клавиатуры
5	Удалить все четные элементы
6	Удалить все элементы с четными индексами
7	Удалить все нечетные элементы
8	Удалить все элементы с нечетными индексами
9	Добавить K простых чисел в начало массива, где K вводится с клавиатуры
10	Добавить K простых чисел в конец массива, где K вводится с клавиатуры
11	Добавить K чисел Фибоначчи в начало массива, где K вводится с клавиатуры
12	Добавить K чисел Фибоначчи в конец массива, где K вводится с клавиатуры
13	Добавить K элементов, начиная с номера N, где N и K вводятся с клавиатуры
14	Добавить после каждого отрицательного элемента его абсолютное значение
15	Добавить после каждого четного элемента, элемент со значением 0

Краткие теоретические сведения

Указатель – это переменная, содержащая адрес другой переменной. Указатели очень широко используются в языке C++. Иногда они дают единственную возможность выразить нужное действие, но обычно ведут к более компактным и эффективным программам.

Так как указатель содержит адрес объекта, это дает возможность "косвенного" доступа к этому объекту через указатель. Предположим, что `x` – переменная, например, типа `int`, а `px` – указатель, созданный неким еще не указанным способом. Унарная операция `&` возвращает адрес объекта, так что оператор `px = &x;` присваивает адрес переменной `x` указателю `px`; говорят, что `px` "указывает" на `x`. Операция `&` применима только к переменным и элементам массива. Нельзя также получить адрес регистровой переменной.

Унарная операция `*` рассматривает свой операнд как адрес конечной цели и обращается по этому адресу, чтобы извлечь содержимое. Следовательно, если `y` 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;`

Описание указателя `int *px;` является новым и должно рассматриваться как мнемоническое; оно говорит, что комбинация `*px` имеет тип `int`. Это означает, что если `px` появляется в контексте `*px`, то это эквивалентно переменной типа `int`.

Указатели могут входить в выражения. Например, если `px` указывает на целое `x`, то `*px` может появляться в любом контексте, где может встретиться `x`. В выражениях вида `y = *px + 1;` унарные операции `*` и `&` связаны со своим операндом более крепко, чем арифметические операции, так что такое выражение берет то значение, на которое указывает `px`, прибавляет 1 и присваивает результат переменной `y`.

Указатели являются переменными, с ними можно обращаться, как и с остальными переменными. Если `py` – другой указатель на переменную типа `int`, то `py = px`; копирует содержимое `px` в `py`, в результате чего `py` указывает на то же, что и `px`.

Использование указателей в качестве альтернативного способа доступа к переменным таит в себе опасность – если был изменен адрес, хранящийся в указателе, то этот указатель больше не ссылается на нужное значение.

Язык C++ предлагает альтернативу для более безопасного доступа к переменным через указатели. Объявив ссылочную переменную, можно создать объект, который, как указатель, ссылается на другое значение, но, в отличие от указателя, постоянно привязан к этому значению. Таким образом, ссылка на значение всегда ссылается на это значение.

Синтаксис объявления ссылки:

```
<имя типа> & <имя ссылки> = <выражение>;
```

или

```
<имя типа> & <имя ссылки>(<выражение>);
```

Однажды инициализировав ссылку, ей нельзя присвоить другое значение. В отличие от указателей, которые могут быть объявлены неинициализированными или установлены в нуль (NULL), ссылки всегда ссылаются на объект. Для ссылок обязательна инициализация при создании и не существует аналога нулевого указателя.

Ссылки нельзя инициализировать в следующих случаях:

– при использовании в качестве параметров функции;

– при использовании в качестве типа возвращаемого значения функции;

– в объявлениях классов.

Не существует операторов, непосредственно производящих действия над ссылками.

Ссылочные переменные используются достаточно редко, значительно удобнее использовать саму переменную, чем ссылку на нее. Более широкое применение ссылки имеют в качестве параметров функции. Ссылки особенно полезны в функциях, возвращающих несколько объектов (значений). Ссылки и указатели в качестве параметров функций тесно связаны.

В С ++ передача аргументов функциям осуществляется "по значению", т. е. переменные, передаваемые в функцию, не меняют свое значение. Если надо изменить значение аргументов функции, тогда в качестве надо использовать ссылки: `void func (int &a, int &b);`

В языке С++ существует взаимосвязь между указателями и массивами. Любую операцию, которую можно выполнить с помощью индексов массива, можно выполнить и с помощью указателей.

Описание `int a[10];` определяет массив размера `10`. Запись `a[i]` соответствует элементу массива через `i` позиций от начала. Если `pa` – указатель целого, описанный как `int *pa;` то присваивание `pa = &a[0]` приводит к тому, что `pa` указывает на нулевой элемент массива `a`. Это означает, что `pa` содержит адрес элемента `a[0]`. Теперь присваивание `x = *pa` будет копировать содержимое `a[0]` в `x`.

Если `pa` указывает на некоторый определенный элемент массива `a`, то по определению `pa+1` указывает на следующий элемент, и вообще `pa–i` указывает на элемент, стоящий на `i` позиций до элемента, указываемого `pa`, а `pa+i` на элемент, стоящий на `i` позиций после. Таким образом, если `pa` указывает на `a[0]`, то `*(pa+1)` ссылается на содержимое `a[1]`, `pa+i` – адрес `a[i]`, а `*(pa+i)` – содержимое `a[i]`.

Эти замечания справедливы независимо от типа переменных в массиве `a`. Суть определения "добавления 1 к указателю", а также его распространения на всю арифметику

указателей, состоит в том, что приращение масштабируется размером памяти, занимаемой объектом, на который указывает указатель. Таким образом, i в pa+i перед прибавлением умножается на размер объектов, на которые указывает pa.

Ссылку на a[i] можно записать в виде *(a+i). Эти две формы эквивалентны. Если применить операцию & к обеим частям такого соотношения эквивалентности, то мы получим, что &a[i] и a+i идентичны: a+i – адрес i-го элемента от начала a. С другой стороны, если pa является указателем, то в выражениях его можно использовать с индексом: pa[i] идентично *(pa+i).

Имеется одно различие между именем массива и указателем: указатель является переменной, так что операции pa=a и pa++ имеют смысл. Но имя массива является константой, а не переменной: конструкции типа a=pa или a++, или p=&a будут ошибочными.

Динамическая память или память свободного хранения отличается от статической тем, что программа должна явным образом запросить память для элементов, хранимых в этой области, а затем освободить память, если она больше не нужна.

С помощью операции выделения памяти можно выделять память динамически, т. е. на этапе выполнения программы:

```
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 два объекта: динамический безымянный объект и указатель на него с именем ip, значением которого является адрес динамического объекта. Если указателю ip присвоить другое значение, то можно потерять доступ к динамическо-

му объекту. В результате динамический объект по-прежнему будет существовать, но обратиться 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];` Далее с этой динамически выделенной памятью можно работать как с обычным массивом.

В случае успешного завершения операция `new` возвращает указатель со значением, отличным от нуля. Результат операции равный `0`, т. е. нулевому указателю `NULL`, говорит о том, что не найден непрерывный свободный фрагмент памяти нужного размера.

Операция освобождения памяти `delete` освобождает для дальнейшего использования в программе участок памяти, ранее выделенной операцией `new: delete ip;` – удаляет динамический объект типа `int`, который был создан с помощью команды `ip = new int;` или `delete [] mas;` – удаляет динамический массив который был создан с помощью команды `double *mas = new double[50];`

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, следует выполнять проверку `if (ip)` перед применением операции `delete`.

Пример выполнения работы

Задание 2.1. Пояснить, что выполняет следующий фрагмент программы, где `N=32`.

```
int *a, n,*aa,t; a=form(n);
for (aa=a; a+n-1-aa>=0; aa+=2){
t=*aa; *aa=*(aa+1); *(aa+1)=t; }
```

Решение

Программа реализует сначала формирование линейного массива

$\{a_i\}$, где $i = 1, \overline{12}$

```
32 33 34 35 36 37 38 39 40 41 42 43
```

Представленный фрагмент программы переставляет местами элементы с четными и нечетными индексами.

```
33 32 35 34 37 36 39 38 41 40 43 42
```

Задание 2.2. Удалить из массива все элементы, которые совпадают с первым элементом. При составлении программы, использовать динамическое распределение памяти и указатели, выполнить ее в NetBeans IDE.

Решение

1. Постановка задачи

Составить программу удаления из массива всех элементов, которые совпадают с первым элементом на языке C++.

2. Алгоритм решения задачи

Алгоритм решения задачи можно представить в виде такой последовательности действий:

Действие 1. Выделить динамическую память для элементов массива.

Действие 2. Заполнить массив случайными числами.

Действие 3. Вывести элементы массива на экран.

Действие 4. Определить количество элементов, которые отличаются от первого.

Действие 5. Выделить динамическую память для нового массива.

Действие 6. В новый массив записать только те элементы, которые отличаются от первого.

Действие 7. Освободить память, которая была выделена под старый массив.

Действие 8. Заменить значение указателя

3. Текст программы

```
#include <iostream>
#include <stdlib.h>
#include <windows.h>
using namespace std;
int* form(int&n){
 cout<<"\nВведите n=";
 cin>>n;
 int*a=new int[n];
 for(int i=0;i<n;i++)
     *(a+i)=rand()%100;
 return a;
}
void print(int*a,int n){
 for(int i=0;i<n;i++)
     cout<<*(a+i)<<" ";
 cout<<"\n";
}
int count0(int*a,int n){
int k=0,i=0;
for(;i<n;i++)
     if(*(a+i)!=*(a+0)) k++;
return k;
}
int*dell(int *a,int&n){
 int newsize=count0(a,n);
int*b=new int [newsize];
for(int j=0, i=0;i<n;i++)
     if(*(a+i)!=*(a+0))
     {
         *(b+j)=*(a+i);j++;
     }
    n=newsize;
```

```
    delete [] a;
    return b;
}

int main(){SetConsoleCP(1251);
 SetConsoleOutputCP(1251);
 int *a, n; a=form(n);
     cout<<"До удаления:\n";print(a,n);
 a=dell(a,n);
 cout<<"После удаления:\n";print(a,n);
    delete [] a;
 return 0;
}
```

4. Результаты работы программы

```
Введите n=10
До удаления:
41 67 34 0 69 24 78 58 62 64
После удаления:
67 34 0 69 24 78 58 62 64
```

Контрольные вопросы

1. Что такое указатель?
2. Как работает унарная операция `&`?
3. Как работает унарная операция `*`?
4. Что такое ссылочная переменная?
5. Где наиболее часто используются ссылочные переменные?
6. Как можно обратиться к определенному элементу массива с помощью индекса?
7. Как можно обратиться к определенному элементу массива с помощью указателя?
8. Что такое динамическая память?
9. Как работает операция `new`?
10. Как работает операция `delete`?

РАБОТА № 3
Разработка и реализация программ с использованием многомерных динамических массивов

Цель работы: овладение навыками разработки программ с использованием многомерных динамических массивов и выполнение их в NetBeans IDE.

Задания

Задание 3.1. Пусть дан следующий фрагмент программы, представленный в вариантах таблицы 3.1. Объясните, какую задачу реализует указанный фрагмент при следующем начале программы. Указание: вместо N подставить номер варианта по списку группы.

```
#include <iostream>
#include <stdlib.h>
#include <iomanip>
const int N=32;
using namespace std;
int** form_matr(int n){
   int **matr=new int*[n];
   for(int i=0;i<n;i++) matr[i]=new int [n];
   return matr;}
void zapolnen (int ** matr,int n){
    int p=N;
    for(int i=0;i<n;i++)
```

```
    for(int j=0;j<n;j++){     matr[i][j]=p++ ;
}}
void printArray (int ** matr,int n){
    for(int  i=0;i<n;i++){
      for(int  j=0;j<n;j++)
        cout<<setw(5)<<matr[i][j];
      cout<<"\n";   }}
int  main(){
    int  n;
    n=10+N%10;
    int  **matr=form_matr(n);
    zapolnen(matr,n);
    printArray(matr,n);
```

Таблица 3.1 – Варианты задания 3.1

№	Фрагмент программы

1–5	`int x=0;`
	`for(int i=0;i<n;i++)`
	`for(int j=i;j<=n-i-1; j++){`
	`x+=matr[i][j] ;`
	`}`

6–10	`int t;`
	`for(int i=0;i<n;i++)`
	`for(int j=i;j<=n-i-1; j++) {`
	`t=matr[i][j] ;`
	`matr[i][j]=matr[n-i-1][j];`
	`matr[n-i-1][j]=t;}`

11–15	`int x=matr[1][n-1], m=0;`
	`for(int i=0;i<n;i++){`
	`for(int j=n-1-i;j<n;j++)`
	`if (matr[i][j]<x){`
	`x=matr[i][j]; m=i;}}`

16–20	`int x=matr[1][n-1], m=0;`
	`for(int i=0;i<n;i++){`
	`for(int j=0;j<=n-1-i; j++)`
	`if (matr[i][j]>x){`
	`x=matr[i][j]; m=i;}}`

Продолж. табл. 3.1

№	Фрагмент программы

21–25	`int t;`
	`for(int j=0;j<n/2;j++)`
	`for(int i=j;i<=n-j-1; i++){`
	`t=matr[i][n-1-j] ;`
	`matr[i][n-1-j]= matr[i][j];`
	`matr[i][j]=t;}`

26–30	`double x=1;`
	`for(int i=0;i<n;i++){`
	`for(int j=0;j<=i;j++)`
	`x*=matr[i][j];`
	`}`

Задание 3.2. Составить программу, которая выполняет вычисления над двумерными динамическими массивами, при написании программы пользоваться принципами структурного программирования (описать такие функции: создание массива, заполнение его случайными числами, печать элементов массива на экран, реализовывает задание из таблицы 3.2).

Таблица 3.2 – Варианты задания 3.2

№	Задание
1	Дана действительная матрица размером `7*8`. Найти максимальный элемент матрицы. Поменять строку, которая содержит наибольший элемент с первой строкой матрицы
2	Дана действительная матрица размером `7*8`. Найти самый большой элемент матрицы. Поменять столбец, который содержит наибольший элемент с первым столбцом матрицы
3	Составить программу замены всех отрицательных элементов матрицы **A**`(9,9)` на `0`, если сумма минимального и максимального элементов этой матрицы окажется меньше `P`, где `P` вводится с клавиатуры.
4	Составить программу нахождения максимального элемента в каждом столбце матрицы **A**`(15,15)`

Продолж. табл. 3.2

№	Задание
5	Составить программу нахождения минимального положительного элемента в каждом столбце матрицы **A**(11,11)
6	Составить программу нахождения количества строк матрицы **A**(10,10), сумма элементов которых отрицательная
7	Составить программу нахождения количества строк матрицы **A**(8, 8), у которых количество отрицательных элементов больше чем P, где *P* вводится с клавиатуры
8	Составить программу формирования вектора **B**(16), если b_i – сумма минимального и максимального элементов i-*й* строки матрицы **A**(16,5)
9	Составить программу замены всех отрицательных элементов матрицы **A**(12,12) на элемент этой матрицы, который имеет максимальное значение
10	Дан двумерный массив целых чисел размерности 6*15. Найти номер строки, для которой среднеарифметическое значение элементов максимально
11	В двумерном массиве целых чисел размерности 9*9 поменять местами строку и столбец, номер которого вводится пользователем
12	Дан массив **C**(14,14). Определить количество "особых" элементов массива, считая элемент "особым", если он больше суммы других элементов своего столбца. Напечатать индексы "особых" элементов
13	Дан массив **C**(11,11). Определить количество "особых" элементов массива, считая элемент "особым", если в строке слева от него находятся элементы меньшие, чем он, а по правую сторону – большие
14	Дан массив целых чисел **C**(11,11). Найти минимальный элемент среди максимальных элементов строк этого массива. Определить номер строки и столбца такого элемента
15	Дан массив целых чисел **C**(11,11). Удалить столбец двумерного массива, в котором находится максимальный элемент
16	Дан массив целых чисел **C**(10,10). Найти все неповторяющиеся элементы двумерного массива

Продолж. табл. 3.2

№	Задание
17	Дан массив **C**(7,10). Удалить столбцы, расположенные между столбцами с минимальным и максимальным элементами
18	Дан массив **C**(9,11). Четные строки массива сдвинуть циклически на *K* элементов вправо, где *K* вводится с клавиатуры
19	Дан массив **C**(9,12). Нечетные строки массива сдвинуть циклически на *K* элементов влево, где *K* вводится с клавиатуры
20	Создайте двумерный массив целых чисел **C**(11,9). Удалите из него строку и столбец, на пересечении которых расположен минимальный элемент

Задание 3.3. Составить программу, которая модифицирует двумерные динамические массивы. При написании программы пользоваться принципами структурного программирования (описать такие функции: создать массив, сформировать массив, печатать на экране, модифицировать массив, согласно заданию из таблицы 3.3, освободить память).

Таблица 3.3 – Варианты задания 3.3

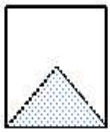
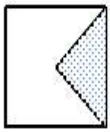
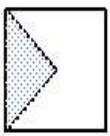
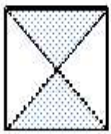
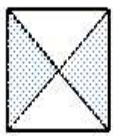
№	Задание
1	Вставить строку с указанным номером *K*
2	Вставить строку в начало матрицы
3	Вставить столбец в начало матрицы
4	Вставить *K* строк в начало матрицы
5	Вставить *K* столбцов в начало матрицы
6	Удалить строку с номером *K*
7	Удалить столбец с номером *K*
8	Удалить строки, начиная со строки K_1 и до строки K_2
9	Удалить столбцы, начиная со столбца K_1 и до столбца K_2
10	Удалить все четные строки
11	Удалить все четные столбцы
12	Удалить все строки, в которых есть хотя бы один нулевой элемент
13	Удалить все столбцы, в которых есть хотя бы один нулевой элемент

Продолж. табл. 3.3

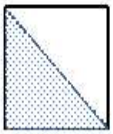
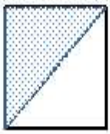
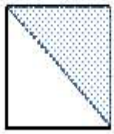
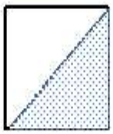
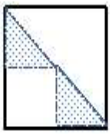
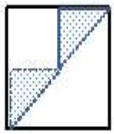
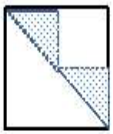
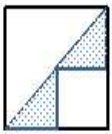
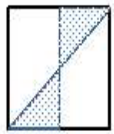
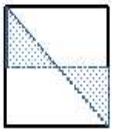
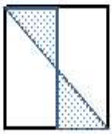
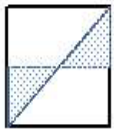
№	Задание
14	Удалить строку, в которой находится самый большой элемент матрицы
15	Добавить строки после каждой четной строки матрицы
16	Добавить столбцы после каждого четного столбца матрицы
17	Добавить *K* строк, начиная со строки с номером *N*
18	Добавить *K* столбцов, начиная со столбца с номером *N*
19	Добавить строку после строки, которая содержит самый большой элемент
20	Добавить столбец после столбца, который содержит самый большой элемент

Задание 3.4. Составить программу, которая заполняет двумерный массив следующим образом: элементы, которые принадлежат заштрихованной области, генерируются случайным образом; остальные равняются `N`, где `N` – номер варианта. Упорядочить все случайные элементы массива, которые принадлежат заштрихованной области. Четные варианты сортируют по возрастанию, нечетные – по убыванию. При написании программы пользоваться принципами структурного программирования (табл. 3.4).

Таблица 3.4 – Варианты задания 3.4

№	Рисунок	№	Рисунок	№	Рисунок
1		2		3	
4		5		6	

Продолж. табл. 3.4

№	Рисунок	№	Рисунок	№	Рисунок
7		8		9	
10		11		12	
13		14		15	
16		17		18	

Краткие теоретические сведения

Частным случаем многомерного массива является двумерный массив, или матрица. Двумерный массив представляет собой совокупность строк и столбцов, на пересечении которых находится конкретное значение. Для объявления двумерного массива необходимо указать количество строк и столбцов. При этом действуют те же правила, что и при объявлении одномерного массива:

```
тип_данных имя_массива [число_строк][число_столбцов];
```

Несмотря на то, что мы представляем двумерный массив в виде матрицы, в памяти любой двумерный массив располагается построчно: сначала нулевая строка, затем первая

и так далее. Об этом следует помнить, т. к. выход за пределы массива может повлечь за собой некорректную работу программы, при этом компилятор не сообщает об ошибке.

Обращение к конкретному элементу массива осуществляется по номеру строки и номеру столбца, например: `array[2][1]`.

Многомерный массив в C++ по своей сути одномерен. Операции `new` и `delete` позволяют создавать и удалять динамические массивы, поддерживая при этом иллюзию произвольной размерности. Деятельность по организации динамического массива требует дополнительного внимания, однако характеристики массива (операнды операции `new`) могут не быть константными выражениями. Это позволяет создавать многомерные динамические массивы произвольной конфигурации. Более подробно работа с операциями `new` и `delete` была рассмотрена в работе № 2.

Организация двумерного динамического массива производится в два этапа: сначала создается одномерный массив указателей, а затем каждому элементу этого массива присваивается адрес одномерного массива:

```
nt size_row = 5, s ze_col = 5;
nt **pArr = new int*[ size_row];
for (int i = 0; i < size_row; i++)
   pArr[i] = new int[s ze_col];
```

Уничтожение двумерного массива происходит в обратной последовательности:

```
for (int i = 0; i < size_row; i++)
     delete[]pArr[i];
  delete[]pArr;
```

Пример выполнения работы

Задание 3.1. Объяснить, что выполняет следующий фрагмент программы, где `N=32`.

```
 nt  x=0;
for(int   =0;i<n;i++){
     for(int  j=0;j<=n-1-  ;j++)
      x+=matr[i][j];
```

Решение

Этот фрагмент программы реализует вычисление суммы элементов, которые расположены на и над второй диагональю матрицы. После выполнения этого фрагмента `x 208`.

Задание 3.2. Составить программу. Удалить из двумерного динамического массива строку, которая содержит максимальный элемент.

Решение

1. Постановка задачи

Составить программу, которая удаляет из двумерного динамического массива строку, содержащую максимальный элемент, на языке С++.

2. Алгоритм решения задачи

Алгоритм решения задачи можно представить в виде такой последовательности действий:

Действие 1. Выделить динамическую память для элементов массива.

Действие 2. Заполнить массив случайными числами.

Действие 3. Вывести элементы массива на экран.

Действие 4. Определить номер (индекс) строки, в которой расположен максимальный элемент.

Действие 5. Выделить динамическую память для нового массива.

Действие 6. В новый массив записать только те строки, которые отличаются от найденного индекса.

Действие 7. Освободить память, которая была выделена под старый массив.

Действие 8. Заменить значение указателя.

3. Текст программы

```
#include <iostream>
#include <stdlib.h>
#include <iomanip>
using namespace std;
int** form_matr(int n,int m){
   int **matr=new int*[n];
   for(int i=0;i<n;i++)
    matr[i]=new int [m];
    return matr;
    //возвращаем указатель на масив указателей
   }

void zapolnen (int ** matr,int n,int m){
   //заполнение массива
    for(int i=0;i<n;i++)
        for(int j=0;j<m;j++)
            matr[i][j]=rand()%100;
   }

void printArray (int ** matr,int n,int m){
    for(int i=0;i<n;i++){
      for(int j=0;j<m;j++)
        cout<<setw(5)<<matr[i][j];
      cout<<"\n";}
   }
void dell(int ** matr,int n,int m){
//освобождение памяти
    for(int i=0;i<n;i++)
        delete [] matr[i];
    delete[]matr;
   }
```

```
   //поиск номера строки с максмальным эле-
   //ментом
   int maxIndex (int ** matr,int n,int m){
   int  max=matr[0][0],imax=0;
    for(int  i=0;i<n;i++)
    for(int  j=0;j<m;j++)
      if (max<matr[i][j]) {  max=matr[i][j];
                              imax=i;}
    return  imax;
    }

int ** deleteMax (int ** matr,int &n,int m){
    int**temp=form_matr(n-1,m);
        //заполнение новой матрицы
    int  t=0,k=maxIndex(matr,n,m);
    for(int  i=0;i<n;i++)
        if(i!=k)
        {
            for(int  j=0;j<m;j++)
                temp[t][j]=matr[i][j];
            t++;
        }

    dell(matr,  n,  m);
       n--;
       return temp;
   }

   int  main()
   {
   int  n,m;//размер  матрицы
   cout<<"\nEnter  n=";
```

```
cin>>n;
cout<<"\nEnter  m=";
cin>>m;
int  **matr=form_matr(n,m);
zapolnen(matr,n,m);
cout<<"old  array:\n";
printArray(matr,n,m);
matr=deleteMax(matr,n,m);
cout<<"new  array:\n";
 printArray(matr,n,m);
 dell(matr,  n,  m);
return  0;
}
```

4. Результаты работы программы

```
Enter  n=5
Enter  m=5
old  array:
  41    67    34     0    69
  24    78    58    62    64
  5    45    81    27    61
  91    95    42    27    36
  91     4     2    53    92
new  array:
  41    67    34     0    69
  24    78    58    62    64
  5    45    81    27    61
  91     4     2    53    92
```

Контрольные вопросы

1. Как объявить двумерный массив?

2. Какие существуют варианты инициализации двумерного массива?

3. Продемонстрируйте взаимосвязь между ссылками и массивами.

4. Что такое динамическая память?

5. Какая существует особенность динамического выделения памяти в случае двумерного массива?

6. Какая существует особенность освобождения памяти в случае двумерного массива?

РАБОТА № 4
Разработка и реализация программ для работы со структурами

Цель работы: овладение навыками составления программ для работы со структурами и выполнение их в NetBeans IDE.

Задания

Задание 4.1. Пусть дан следующий фрагмент программы, представленный в вариантах таблицы 4.1. Объясните, какую задачу реализует указанный фрагмент, если программа начинается следующим образом:

```
   #include <iostream>
   #include <cstring>
   #include<cctype>
   using namespace std;
   const int N=4;
   struct myType
   {    char fff[20];     char aaa[20];
        int b;     double c;};
void printStruct(myType st[],int n){
   for (int i=0;i<n;i++){
     cout << " fff: " << st[i].fff <<
          "  aaa: "<< st[i].aaa ;
```

```
    cout << "  b : " << st[i].b<<"   c : "
         << st[i].c << endl;}}
int  main(){
  myType  st[]={{"assdss","ssdhshf",4,4.5},
            {"jherhhgb","xczvfdg",9,2.4},
           {"xcvfd","hhgeryt",5,4.1},
            {"jrhjdfd","tyyyyy",7,3.5} };
printStruct(st,N);
```

Таблица 4.1 – Варианты задания 4.1

№	Фрагмент программы

1–5	`double k=0; char c='d';`
	`for (inti=0; i<N; i++){`
	`if (strchr(st[i].fff,c)) k++;`
	`}`

6–10	`char t[20];`
	`for (inti=0; i< N; i++) {`
	`for (int j= N-1; j>i; j--){`
	`if (strcmp(st[j].aaa ,st[j 1].aaa)<0)`
	`{`
	`strcpy(t, st[j].aaa);`
	`strcpy(st[j].aaa, st[j-1].aaa);`
	`strcpy(st[j-1].aaa ,t);}}}`

11–15	`double a=0;`
	`for (inti=0; i< N; i++) a+=st[i].c;`
	`a=a/N;`
	`for (inti=0; i< N; i++)`
	`if (st[i].c>a) cout<<st[i].fff<<"\n";`

16–20	`for (inti=0; i< N; i++)`
	`if ((*(st+i)).b%2)`
	`cout<<(*(st+i)).fff<<"\n";`

21–25	`char c='j';char *l;`
	`for (inti=0; i<N; i++){`
	`if (l=strchr(st[i].fff,c))`
	`for (int j=0; j<strlen(l); j++)`
	`st[i].fff[j]-=32;`
	`}`

Продолж. табл. 4.1

№	Фрагмент программы

26–30	`char c='f';char *l;`
	`for (inti=0; i<N; i++){`
	`if (l=strchr(st[i].aaa,c))`
	`{int k=strlen(st[i].aaa);`
	`for (int j=0;j<k/2;j++) {char c=st[i].aaa[j];`
	`st[i].aaa[j]=st[i].aaa[k-j-1];`
	`st[i].aaa[k-j-1]=c;}`
	`}`
	`}`

Задание 4.2. Создать структуру, спецификация которой приведена в таблице 4.2. Определить функции, которые создают массив структур. Задать критерии отбора.

Таблица 4.2 – Варианты задания 4.2

№	Задание
1	Структура "Преподаватель": – фамилия, имя, отчество; – номер телефона; – кафедра; – дисциплины, которые преподает (5 названий). Создать массив структур. Вывести: *a*) список преподавателей указанной кафедры; *b*) список преподавателей, которые преподают заданную дисциплину; *c*) список преподавателей в алфавитном порядке
2	Структура "Поезд": – номер; – пункт назначения; – время отправления; – количество купейных мест, – количество плацкартных мест; – количество общих мест; Создать массив структур. Вывести: *a*) список поездов, которые отправляются из заданного пункта; *b*) список поездов, которые направляются к заданному пункту назначения и отправляются после указанного часа; *c*) список поездов, которые отправляются к заданному пункту назначения и имеют плацкартные места

Продолж. табл. 4.2

№	Задание
3	Структура "Абитуриент": – фамилия, имя, отчество; – год рождения; – оценки ВНО (3); – средний балл аттестата. Создать массив структур. Вывести: *a*) список абитуриентов, которые имеют оценки по ВНО ниже указанного минимума; *b*) список абитуриентов, у которых средний балл аттестата выше заданного значения; *c*) выбрать n абитуриентов, которые имеют наибольшие суммарные баллы в рейтинге
4	Структура "Сотрудник": – фамилия, имя, отчество; – должность; – год рождения; – заработная плата. Создать массив структур. Вывести: *a*) список сотрудников, у которых заработная плата ниже указанного значения; *b*) список сотрудников, которые занимают определенную должность; *c*) список сотрудников в алфавитном порядке
5	Структура "Государство": – название; – столица; – численность населения; – занимаемая площадь. Создать массив структур. Вывести: *a*) список государств, которые имеют заданную численность населения; *b*) список государств, которые имеют площадь большую, чем указанная и численность населения в указанном диапазоне; *c*) список государств в алфавитном порядке
6	Структура "Человек": – фамилия, имя, отчество; – домашний адрес; – номер телефона; – возраст.

Продолж. табл. 4.2

№	Задание
6	Создать массив структур. Вывести: *a*) сведения о людях, у которых номер телефона принадлежит определенному оператору сотовой связи; *b*) сведения о людях, которые живут на одной улице; *c*) сведения о людях в алфавитном порядке
7	Структура "Покупатель": – фамилия, имя, отчество; – домашний адрес; – номер телефона; – номер дисконтной карточки; – сумма затрат. Создать массив структур. Вывести: *a*) список покупателей в алфавитном порядке; *b*) список покупателей, у которых номер дисконтной карточки находится в заданном интервале; *c*) список покупателей, у которых сумма расходов больше, чем указанное значение
8	Структура "Человек": – фамилия, имя, отчество; – год рождения; – рост; – вес; – пол. Создать массив структур. Вывести: *a*) сведения о человеке, который имеет вес больше заданного и его возраст задается определенным диапазоном; *b*) сведения о людяхженского пола и определенного роста; *c*) сведения о людях в алфавитном порядке
9	Структура "Пациент": – фамилия, имя, отчество; – домашний адрес; – номер медицинской карты; – диагноз. Создать массив структур. Вывести: *a*) список пациентов, которые имеют указанный диагноз; *b*) список пациентов, чей номер медицинской карточки имеет последние цифры 34; *c*) сведения о пациентах в алфавитном порядке

Продолж. табл. 4.2

№	Задание
10	Структура "Футбольная команда": – название; – город; – количество игроков; – количество набранных очков. Создать массив структур. Вывести: *a*) список команд из одного города; *b*) список команд, которые набрали меньше указанного количества очков; *c*) отсортировать команды по названию
11	Структура "Владелец автомобиля": – фамилия, имя, отчество; – номер автомобиля; – телефон; – цвет; – номер техпаспорта. Создать массив структур. Вывести: *a*) список владельцев автомобилей заданного цвета; *b*) список владельцев автомобилей, которые в номере имеют определенные две цифры; *c*) сведения о владельцах автомобилей в алфавитном порядке
12	Структура "Студент": – фамилия, имя, отчество; – домашний адрес; – группа; – телефон; – рейтинг. Создать массив структур. Вывести: *a*) список студентов указанной группы; *b*) список N студентов, которые имеют наивысший рейтинг; *c*) список учебной группы в алфавитном порядке
13	Структура " DVD-Диск": – название фильма; – год выпуска; – жанр; – продолжительность; – цена. Создать массив структур. Вывести: *a*) сведения о DVD-дисках определенного жанра за последние пять лет; *b*) сведения о DVD-диске с наименьшей ценой; *c*) сведения о фильмах в алфавитном порядке

Продолж. табл. 4.2

№	Задание
14	Структура "Автомобиль": – марка; – год выпуска; – цена; – цвет. Создать массив структур. Вывести: *a*) список автомобилей заданной марки; *b*) список автомобилей заданной марки, которые эксплуатируются больше *n* лет; *c*) список автомобилей указанного года выпуска, цена которых больше указанной
15	Структура "Государство": – название; – количество государственных языков; – государственный язык (вариативная часть); – денежная единица; – курс валюты относительно $. Создать массив структур. Вывести: *a*) список государств, которые имеют больше, чем один государственный язык, языки тоже выводить; *b*) список государств, которые имеют низкий курс относительно $; *c*) список государств в алфавитном порядке
16	Структура "Владелец автомобиля": – фамилия, имя, отчество; – номер автомобиля; – номер техпаспорта; – отделение регистрации ГАИ. Создать массив структур. Вывести: *a*) список автомобилей, которые регистрировались в определенном отделении ГАИ; *b*) список автомобилей, которые принадлежат определенному человеку; *c*) список автомобилей в алфавитном порядке
17	Структура "Книга": – название; – автор; – жанр; – год издания; – количество страниц.

Продолж. табл. 4.2

№	Задание
17	Создать массив структур. Вывести: *a*) список книг заданного автора; *b*) список книгопределенного жанра; *c*) список книг, которые изданы после заданного года
18	Структура "Фильм": – название; – режиссер; – страна; – принесенная прибыль. Создать массив структур. Вывести: *a*) список фильмов определенного режиссера; *b*) список фильмов, которые принесли прибыль больше *N*и снимались в Голливуде; *c*) список фильмов в алфавитном порядке
19	Структура "Автомобиль": – марка; – серийный номер; – количество оборотов двигателя; – литраж двигателя; – год выпуска. Создать массив структур. Вывести: *a*) список автомобилей заданной марки; *b*) список автомобилей, которые имеют объем двигателя больше, чем 3 ли эксплуатируются меньше *n* лет; *c*) список автомобилей указанного года выпуска, которые имеют определенную цифру в серийном номере
20	Структура "Бассейн": – название; – адрес; – город; – дни и время работы; – телефон администратора. Создать массив структур. Вывести: *a*) список бассейнов, которые расположены в определенном городе; *b*) список бассейнов, которые не работают позднее 20.00; *c*) отсортировать бассейны по названию

Краткие теоретические сведения

Структура – это сложный тип, состоящий из одного или более полей, которые могут иметь различные типы, объединенные под одним именем. Назначение структур – создание новых типов данных. Тип данных `struct` – один из основных строительных блоков данных в языке C++, который предоставляет удобный способ объединения различных элементов, имеющих логическую связь. Все данные, которые относятся к одному объекту, собраны вместе.

Рассмотрим синтаксис объявления структуры на примере создания пользовательского типа данных (структуры) для хранения даты:

```
struct date
{
 int day; //день
 int month; //месяц
 int year;//год
 int weekday; //день недели
 char mon_name[15];// название месяца
};
```

Создание объекта с типом `date` и инициализация его при создании: `date my_birthday={20,7,1981,1,"July"};`

Особенности структур:

1. Описание структуры начинается со служебного слова `struct`, за которым может следовать необязательное имя, называемое именем типа структуры. Это имя типа структуры используется в дальнейшем для создания конкретного объекта.

2. За именем типа структуры идет, заключенный в фигурные скобки, список элементов структуры с описанием типа каждого элемента (элементом структуры может быть переменная, массив или структура). Элементы структуры отделяются друг от друга точкой с запятой.

3. За правой фигурной скобкой, закрывающей список элементов, может следовать список объектов. Например, оператор `struct date {...} x, y, z;` определяет переменные `x, y, z` в качестве структур описанного типа и приводит к выделению памяти.

4. Описание структуры, за которым не следует список объектов, не приводит к выделению памяти, оно только определяет шаблон (форму) структуры. Однако, если такое описание снабжено именем типа (например, `date`), то это имя типа может быть использовано позднее при определении объектов структур.

5. Структуру можно инициализировать, поместив вслед за ее определением список инициализаторов для ее компонентов, заключенный в фигурные скобки.

Структуры могут вкладываться одна в другую, но самовложение структур запрещено. Операция доступа к элементу структуры `"."` вычисляется слева направо.

Обращение к определенному члену структуры производится через "оператор точка" (.) с помощью конструкции вида:

`<имя структуры>.<имя элемента>:`

`my_birthday.day; my_birthday.month;` и т. д.

Также доступ к элементу структуры можно осуществлять по указателю с помощью операции `"->"` в виде:

`<указатель на структуру>"->"<элемент_структуры>.`

Обращения `pd->year` и `(*pd).year` эквивалентны. Круглые скобки `(*pd)` необходимы, поскольку операция `"."` доступа к элементу структуры старше, чем `"*"`. Однако оператор `"."` не может быть перегружен, в отличие от оператора `"->"`, который, как правило, предназначен для работы с указателями.

Применимо определение адреса структуры с помощью операции `"&"`, а также присваивание структуры как единого

целого. Структуры можно передавать в качестве параметра функции и возвращать в результате работы функции, а также передавать отдельные компоненты структуры в качестве аргументов функции.

Порядок выполнения некоторых наиболее распространенных операций над элементами структуры следующего описания:

```
struct {int x; int *y;}
*p; // p – указатель на структуру
```

1) `(++p)->x` – операция увеличивает `p` до доступа к `x`;

2) `(p++)->x` – операция увеличивает `p` после доступа к `x` (круглые скобки не обязательны, т. к. по старшинству раньше будет применена операция `"->"`);

3) `*p->y` – выбирается содержимое объекта, на который указывает `y`;

4) `*p->y++` – увеличивается `y` после обработки того, на что он указывает;

5) `*p++->y` – увеличивает `p` после выборки того, на что указывает `y`;

6) `(*(*p).y)++` – увеличивает то, на что указывает `y`.

В языке C++ существует специальная унарная операция `sizeof()`, которая возвращает размер своего операнда в байтах. Операндом операции `sizeof()` может быть любое выражение: `sizeof`(выражение); Результат операции `sizeof()` имеет тип `int`.

Размер структуры не равен сумме размеров ее членов. Вследствие выравнивания объектов разной длины в структуре могут появляться безымянные "дыры". Так, например, если переменная типа `char` занимает один байт, а `int` – четыре байта, то для структуры: `struct Test {char c; int i;};` может потребоваться восемь байт, а не пять. Правильное значение возвращает операция `sizeof()`.

Пример выполнения работы

Задание 4.1. Определить действие фрагмента программы

```
char  c='j';
   for (int i=0; i<N; i++){
         if (st[i].fff[0]==c){
              strcat(st[i].fff,"123");
         }
}
```

Решение

Этот фрагмент программы добавляет к символьному полю структуры число 123, если строка начинается с символа 'j':

```
fff: assdss  aaa: ssdhshf  b : 4   c : 4.5
fff: jherhhgb123  aaa: xczvfdg  b : 9   c : 2.4
fff: xcvfd  aaa: hhgeryt  b : 5   c : 4.1
fff: jrhjdfd123  aaa: tyyyyy  b : 7   c : 3.5
```

Задание 4.2. Составить программу, которая определяет структуру "Студент": ФИО, адрес, возраст. Вывести на экран данные об определенном студенте; упорядочить в порядке возрастания данные (по ФИО).

Решение

1. Постановка задачи

Составить программу, которая определяет структуру "Студент": ФИО, адрес, возраст. Вывести на экран данные об определенном студенте; упорядочить в порядке возрастания данные (по ФИО).

2. Алгоритм решения задачи

Алгоритм решения задачи можно представить в виде такой последовательности действий:

Действие 1. Определить массив структур st1153.

Действие 2. Ввести значение полей структуры.

Действие 3. Ввести ФИО определенного студента.

Действие 4. Для каждого элемента массива выполнить проверку условия (поле ФИО массива структур совпадает с полем ФИО определенного студента).

Действие 5. Отсортировать массив структур по полю ФИО.

Действие 6. Вывести элементы массива `st1153` после сортировки.

3. Текст программы

```
   #include <iostream>
   #include <windows.h>
   using namespace std;
   struct student{
    char *fio;
    char *adress;
    int age;};
student enterStruct(){
   student st;
   cout << "ФИО: " ;
   st.fio=new char [20];
   cin.getline(st.fio,20);
   cout << "Адрес: " ;
   st.adress=new char [20];
   cin.getline(st.adress,20);
   cout << "Возраст: " ;
   cin>>st.age;
   return st;
   }
void printStruct(student st1153[],int n){
   for (int i=0;i<n;i++){
    cout << "ФИО: " << st1153[i].fio <<
        endl;
    cout << "Адрес: "    << st1153[i].adress
```

```
          << endl;
     cout << " Возраст: " << st1153[i].age
          << endl;
   }
   }
void sortStudent(student st1153[],int n){
   student t;
   for (int i=0; i< n; i++) {
   for (int j= n-1; j>i; j--){
    if (stricmp(st1153[j].fio ,st1153[j-
1].fio)<0) {
      t=st1153[j];
      st1153[j]= st1153[j-1];
      st1153[j-1]=t;}}}
   }
void printFilter(student st1153[],int
n,char * s){
   for (int i=0;i<n;i++)
    if (stricmp(st1153[i].fio,s)==0){
      cout << "ФИО: " << st1153[i].fio <<
           endl;
      cout << "Адрес: " << st1153[i].adress
          << endl;
      cout << " Возраст: " << st1153[i].age
           << endl;
   }
   }
   int main(){
      SetConsoleCP(1251);
      SetConsoleOutputCP(1251);
    student st1153[26];
    int n=3;char fio[20];
```

```
for (int i=0;i<n;i++){
   st1153[i]=enterStruct();
   cin.ignore();
}
cout<<"\nВведите ФИО:";
cin.getline(fio,20);
printFilter (st1153,n,fio);
cout<<"\n Упорядоченные данные \n";
sortStudent(st1153,n);
printStruct(st1153,n);
  return 0;
}
```

4. Результаты работы программы

```
ФИО: Петренко Иван
Адрес: 1 Линия, д.7
Возраст: 21
ФИО: Веселов Максим
Адрес: 7 Продольная, д.6
Возраст: 22
ФИО: Быков Андрей
Адрес: пр. Мира, д.32
Возраст: 20

Введите ФИО: Веселов Максим
ФИО: Веселов Максим
Адрес: 7 Продольная, д.6
Возраст:  22

Упорядоченные данные
ФИО: Быков Андрей
Адрес: пр. Мира, д.32
Возраст:  20
```

```
ФИО: Веселов Максим
Адрес: 7 Продольная, д.6
Возраст:  22
ФИО: Петренко Иван
Адрес: 1 Линия, д.7
Возраст:  21
```

Контрольные вопросы

1. Что такое структура?
2. Какой синтаксис объявления структуры?
3. Назовите особенности структур.
4. Назовите операторов обращения к определенному полю структуры и их отличия.
5. Назовите наиболее распространенные операции над элементами структуры.
6. Что делает унарная операция `sizeof()`?

РАБОТА № 5
Разработка и реализация программ с использованием форматированного ввода/вывода на консоль и текстовый файл

Цель работы: овладение навыками составления программ с использованием форматированного ввода/вывода на консоль и текстовый файл, и выполнение их в NetBeans IDE.

Задания

Задание 5.1. Определить действие фрагмента программы по вариантам, приведенные в таблице 5.1. Определить содержимое файла `file.txt`, если

```
#include <cstdio>
int main()
{   FILE * fl;
   fl=fopen("d:\\file.txt","w");
   double d, x; float f; long lng;
   int i, y; short s;
```

Указание: `int N` равняется номеру варианта; если формат не соответствует типу переменной, объясните где и как это влияет на результат.

Таблица 5.1 – Варианты задания 5.1

№	Фрагмент программы

1–5	`s = i = lng = f = d = N*100/3;`
	`fprintf(fl,"s = %hd i = %d lng = %ld f = %f d = %f\n", s, i, lng, f, d);`
	`d = 3.2; i = 2;x = ( y = d / i ) * N;`
	`fprintf(fl,"x = %f ;y = %d\n", x, y);`

6–10	`d = f = lng = i = s =100/3*N;`
	`fprintf(fl,"s = %hd i = %d lng = %ld f = %f d = %f\n", s, i, lng, f, d);`
	`d = 3.2; i = 2;x = ( y = d *N / i )*2;`
	`fprintf(fl,"x = %d ;y = %f\n", x, y);`

11–15	`lng = s = f = i = d =N*100/3;`
	`fprintf(fl,"s = %hd i = %d lng = %ld f = %f d = %f\n", s, i, lng, f, d);`
	`y = ( x = d / i *N) *2; fprintf(fl,"x = %f ;y = %d\n", x, y);`

16–20	`f = s = d = lng = i = double(100*N)/3;`
	`fprintf(fl,"s = %hd i = %d lng = %ld f = %f d = %f\n", s, i, lng, f, d);`
	`y = d * ( x = 2.5 / d)*N;fprintf(fl,"x = %f; y = %d\n", x, y);`

21–25	`s = i = lng = f = d = 100/(double)3;`
	`fprintf(fl,"s = %hd i = %d lng = %ld f = %f d = %f\n", s, i, lng, f, d);`
	`y = d*N*( x = 2.5 / (d*N)); fprintf(fl,"x = %f; y = %d\n", x, y);`

26–30	`i = s = lng = d = f = (double)(100/3)*N;`
	`fprintf(fl,"s = %hd i = %d lng = %ld f = %f d = %f\n", s, i, lng, f, d);`
	`d = 3.2; i = 2;`
	`x = d / N*( y = ( (int)2.9 + 1.1) / d *N);`
	`fprintf(fl,"x = %f y = %d\n", x, y);`

Задание 5.2. Составить и выполнить программу, которая обрабатывает информацию из текстового файла согласно заданию, варианты приведены в таблице 5.2. Результат записать в файл в форме отчета, обязательно использовать форматирование текста.

Указание: файл с исходными данными можно создать с помощью текстового редактора.

Таблица 5.2 – Варианты задания 5.2

№	Задание
1–2	Директор школы набирает группу школьников для обучения по факультативной программе. Обучение платное, общая стоимость курса *K* грн. Сколько должен платить каждый ученик? Очевидно, эта сумма зависит от значения *K* и от количества учеников. Вычислить и записать в файл таблицу сумм, которую должен внести один ученик, если группа будет состоять из трех, четырех, и т. д., до 20-ти учеников
3–4	На день рождения ребенка бабушка открыла счет в банке и положила на него 30 долларов. Каждый год она добавляет 50 долларов. Годовой процент по банковскому счету равняется 7 %. Какая сумма накопится к совершеннолетию ребенка (к 16-ти годам), включая последний взнос. Вывести в файл таблицу ежегодного состояния счета
5–6	Процент по банковским вкладам равняется 9 %. Если положить в банк сумму *N* грн, то эта сумма будет ежегодно увеличиваться. Как будет изменяться сумма на протяжении ближайших 10-ти лет? Если годовая инфляция составляет 4,5 %, сколько же на самом деле будут стоить эти деньги? Вывести в файл таблицу ежегодного состояния счета
7–8	Пусть в файле хранится информация о багаже пассажира: ФИО пассажира, количество вещей и общий вес вещей. Написать функции для чтения и вывода в/из файла общей информации о багаже. Найти пассажиров, которые имеют не больше двух вещей. Найти количество пассажиров, количество вещей которых превосходит среднее количество вещей. Результаты занести в файл
9–10	Оплата труда няни осуществляется почасово. За время до 6-ти часов работы она получает по 25 грн за час. Начиная с 6-го часа работы, стоимость каждого следующего часа увеличивается на 20 %, то есть 30 грн, потом 36 грн и т. д. Родители, отправляясь на вечеринку, хотят знать сумму, которую они заплатят няне, но не знают, насколько задержатся. Вычислить и вывести в файл таблицу оплат услуг няни, начиная с одного часа и до 24-х часов
11–12	Женившись, молодые супруги решили откладывать деньги на покупку автомобиля. Мужчина может положить ежемесячно *M* $, жена *V* $. Если положить деньги в банк, то по срочному вкладу годовой процент равняется 8 %. Автомобиль мечты имеет стоимость *N* $. Через какой срок супруги поедут на собственном авто? Выведите в файл таблицу ежемесячных накоплений с учетом процента по банковским вкладам

Продолж. табл. 5.2

№	Задание
13–14	После окончания сессии всегда есть некоторое количество "должников". Деканат решил провести курсы для отстающих в объеме 20 часови установил стоимость оплаты часа, которая равняется 100 грн. Из суммы, уплаченной студентами, преподавателю принадлежит 20%. Найти, сколько денег получит преподаватель, если будет заниматься с одним, двумя, тремяи т. д., до *M* студентов. Может ли он получить за эту работу 5000 грн? Скольких бездельников ему придется учить? Для убедительности выведите таблицу в файл
15–16	Незнайка учит английский язык. В первый день он выучил два слова, а каждый последующий день собирался изучать на два слова больше, чем в предыдущий. Найдите и выведите в файл в виде таблицы, на который день Незнайка выучит 100 слов, 200, 300, 400 и т.д., до 1000? В английском языке около 50 тыс. слов, а срок жизни Незнаек приблизительно 30 лет. Успеет ли к своей кончине Незнайка выучить английский язык? Если нет, то сколько жизней Незнайке понадобится, чтобы выучить английский язык?
17–18	Бабушка решила купить телевизор, когда внук подарил ей 500 грн. Она положила их в банк под 19% годовых. Ежемесячно на этот же счет бабушка вносит 50 грн. Самый дешевый телевизор стоит *K* грн. Через сколько месяцев бабушка купит телевизор? Вычислить и вывести в файл состояние счета ежемесячно
19–20	В файле имеется таблица "Успешность студентов за год". Одна строка таблицы соответствует одному студенту и содержит такие сведения: ФИО студента, номер группы, экзаменационные оценки за первый семестр, экзаменационные оценки за второй семестр. Упорядочить строки таблицы по номеру группы и найти студентов, у которых средний балл за второй семестр ниже, чем за первый. Результаты записать в файл
21–22	Улитка ползет вверх по дереву с начальной скоростью *V* м/с. При этом она устаети ее скорость движения падает по прямолинейному закону на 0,1**V* каждый час. Вычислить высоту, на которой улитка будет находиться каждый час, вывести таблицу в файл. Узнать через сколько часов она остановится
23–24	Фабрика по производству тапочек ежегодно увеличивает объем продаж на 5 %. Себестоимость продукции при этом уменьшается на 1 %. В текущем году объем продаж составил *N* тыс. грн, а себестоимость пары тапочек – *C* грн. Вычислить и вывести в файл таблицу увеличения объема продаж и снижение себестоимости на ближайшие *K* лет

Продолж. табл. 5.2

№	Задание
25–26	Ученик мастера начинает с изготовления в день одной табуретки. Совершенствуясь, он каждый день изготовляет на одну табуретку больше, чем в предыдущий. Больше, чем 20 табуреток в день изготовить нельзя. Найти, в какой день ученик достигнет вершин мастерства и сколько табуреток он смастерит за все время обучения. Для убедительности вывести в файл таблицу ежедневной производительности
27–28	Фабрика изготовляет туфли, тапочки и боты. Данные об ежемесячных объемах сбыта продукции каждого вида за прошлый год хранятся в файле. При подведении итогов года дирекция требует найти суммы накопительных итогов для каждого вида продукции и сохранить эту информацию в файле, а также выяснить, в каком месяце был самый большой сбыт по каждому виду продукции. Использовать функции для накопления итогов и для поиска максимума
29–30	В файле хранится информация о количестве студентов в той или другой группе каждого курса института с первого по пятый (в первом столбце – информация о группах первого курса, во втором – второго и т. д.). На каждом курсе есть 8 групп. Определить среднее число студентов на каждом курсе, определить на каком курсе меньше всего студентов

Краткие теоретические сведения

Файл – это поименованная совокупность данных, которая сохраняется на носителе информации.

В C++ отсутствуют операторы для работы с файлами. Все необходимые действия выполняются с помощью функций, включенных в стандартные библиотеки. Они позволяют работать с различными устройствами, при этом файловая система преобразует их в единое абстрактное логическое устройство – поток.

Существует несколько разновидностей файлов. В Windows таких разновидностей две: текстовый файл и бинарный (двоичный) файл. Данная классификация в качестве критерия

использует способ хранения информации. Сравнительная характеристика текстового и бинарного файлов приведена в приложении Е.

Файл можно открыть в текстовом или в двоичном режиме. Не следует отождествлять режим открытия файла с текстовым и двоичным форматами. Разница между режимами открытия состоит в том, что при открытии файла в двоичном режиме информация будет в том же виде, в котором она хранится на диске или в памяти. При текстовом режиме информация преобразовывается в символы и имеет разделители, например, табуляция `(\t)` и окончания строки `(\n)`.

Организация работы с файлами средствами языка С. Структура `FILE`

Структура `FILE` – это сущность языка С. Она определена в заголовочном файле стандартной библиотеки `<stdio.h>`. Размер этой структуры и ее поля зависят от ОС и от версии компилятора, поэтому обычно пользуются указателем на эту структуру `FILE*`, например, `FILE *f`;

Файловый указатель – специальная переменная, которая автоматически присваивается открытому файлу и хранит текущую позицию в файле. Она перемещается по файлу в момент процессов записи и чтения.

Дескриптор файла – уникальный номер, который операционная система присваивает любому открытому файлу, чтобы отличать его от других. Когда файл закрывается, система "отбирает" у него дескриптор. Именно это уникальное число используется для работы с конкретным файлом в программах. Узнать дескриптор конкретного файла можно с помощью соответствующей функции.

Перед началом работы с файлом его необходимо открыть с помощью функции `fopen()` (приложение А). Типы доступа для этой функции приведены в приложении Б. Если

функция отработала успешно, из нее возвращается указатель на открытый файл, в противном случае – нуль. Указатель на открытый файл принято хранить в типе данных `FILE*`.

Неформатированный файловый ввод/вывод осуществляют с помощью функций `fwrite()` (запись в файл) и `fread()` (чтение из файла).

Для форматированного файлового ввода/вывода можно использовать следующие функции:

- `fgetc()` и `fputc()` позволяют соответственно осуществить ввод/вывод символа;
- `fgets()` и `fputs()` позволяют соответственно осуществить ввод/вывод строки;
- `fscanf()` и `fprintf()` позволяют соответственно осуществить форматированный ввод/вывод и аналогичны соответствующим функциям форматированного ввода/вывода, приведенным в приложении В, только делают это применительно к файлу.

Позиционирование по файлу осуществляется с помощью функций `fseek()` и `ftell()`.

После окончания работы с файлом его необходимо закрыть с помощью функции `fclose()`. Отсутствие этой команды может привести к потере данных в файле, который был открыт для записи (дозаписи). Если не закрывать файлы, которые открыты для чтения, то это может привести к ограничению доступа к файлу для других программ. Кроме того, в любой ОС есть ограничение на количество одновременно открытых файлов.

Для форматного вывода в различные потоки значений различных типов, отформатированных согласно заданному шаблону, используется функция `printf()` для вывода в консоль и `fprintf()` для вывода в файл.

Использование этой функции в программах на C++ в некоторых случаях бывает более удобным, чем примене-

ние стандартных потоков cout и cin. Прототип этой функции находится в заголовочном файле stdio.h.

Формат, указываемый при обращении к функции printf(), выглядит следующим образом:

```
printf (Управляющая_строка,Аргумент1,-
Аргумент2,...);
```

Управляющая_строка – строка символов, задающая формат вывода данных. Помимо обычного текста она содержит команды преобразования, которые начинаются с символа "%", за которым следуют символы и цифры, задающие правила вывода аргументов. Формат определяется следующим образом:

```
%[флажки][ширина][.точность][F|N|h|l|L]
преобразование
```

Формат (команда преобразования) начинается с обязательного знака процента. Чтобы вставить в выводимый текст сам знак процента, его нужно указать дважды: %%.

Конструкции в квадратных скобках могут отсутствовать, а символ "|" (вертикальная черта) свидетельствует о том, что здесь может использоваться одна из перечисленных возможностей.

Аргумент1, Аргумент 2,... – печатаемые параметры, которые могут быть переменными, константами или выражениями, вычисляемыми перед выводом на печать.

Функции scanf() и fscanf() предназначены для ввода данных. Так же, как и для функций printf()/fprintf(), для функций scanf()/fscanf() указываются управляющая строка и следующий за ней список аргументов. Обращение к этой функции имеет вид:

```
scanf (Управляющая_строка,&Имя1,&Имя2,...,&ИмяN);
```

Управляющая_строка – это строка символов, которая задает количество и типы вводимых переменных. Делается это так: в формате указывается символ %, за которым следует буква, определяющая тип вводимой переменной.

Сочетание %буква является спецификацией преобразования. При вводе используются следующие спецификации:

%d – ввести целое число;

%c – ввести один символ;

%s – ввести строку символов.

Спецификации преобразования должны соответствовать количеству и типу вводимых переменных.

Имя1,Имя2,...,ИмяN – это имена переменных, значения которых надо ввести.

Правила применения функций scanf()/fscanf():

1. Если нужно ввести некоторое значение и присвоить его переменной одного из основных типов, то перед именем переменной требуется писать символ &, т. е. передавать переменную в функцию по ссылке.

2. Если требуется ввести значение строковой переменной, то использовать символ & не нужно.

При обращении к функциям scanf()/fscanf() выполнение программы приостанавливается до ввода значений указанных переменных, после чего работа программы продолжится.

Управляющая строка содержит спецификации преобразования, которые используются для непосредственной интерпретации входной последовательности. Управляющая строка может содержать:

– пробелы ' ', символы табуляции '\t' и перевода строки '\n', которые являются разделителями;

– обычные символы (кроме символа "%"), которые предполагаются совпадающими с очередными (отличными от символов, перечисленных в предыдущем пункте) символами входного потока;

– спецификации преобразования, состоящие из %, необязательного символа "подавления" присваивания "*", не-

обязательного числа, задающего максимальную ширину поля и символа преобразования.

Спецификация преобразования управляет преобразованием очередного входного поля. Результат обычно помещается в переменную, на которую ссылается соответствующий параметр. Однако, если присутствует символ `"*"`, то входное поле просто пропускается и никакого присваивания не производится.

Основные символы преобразования:

`d` – десятичное число;

`x` – шестнадцатеричное число;

`o` – восьмеричное число;

`c` – символ;

`s` – строка символов.

Пример выполнения работы

Задание 5.1. Определить действие фрагмента программы

```
i = s = lng =  f = (double)(1000000/3)*N;
fprintf(fl,"s = %hd i = %d lng = %ld f =
%f\n", s, i, lng, f);
d = 3.2;x = d / N*( y = ( 1 + (int)1.1) / d );
fprintf(fl,"x = %f y = %d\n", x, y);
```

Решение

Этот фрагмент программы записывает в текстовый файл `file.txt`, следующие данные

```
s = -15712 i = -15712 lng = 10666656
f = 10666656.000000
x = 0.000000 y = 0
```

Поскольку переменная `s` имеет тип `short` (диапазон – 32768…32767), возникает переполнение разрядной сетки, и как следствие недостоверный результат.

Задание 5.2. Составить и выполнить программу, которая обрабатывает информацию. В текстовом файле хранится

информация о зарплате 18 лиц за каждый месяц (в первом столбике – зарплата за январь, во втором – за февраль и т. д.). Составить программу для расчетов средней зарплаты за любой месяц. Результат записать в файл в форме отчета, обязательно использовать форматирование текста.

Решение

1. Постановка задачи

Составить программу для расчетов средней заработной платы за любой месяц.

2. Алгоритм решения задачи

Алгоритм решения задачи можно представить в виде такой последовательности действий:

Действие 1. Напишем программу, которая создает и заполняет текстовый файл `file.txt`.

Действие 2. Напишем еще одну программу, которая выполняет следующие действия:

Действие 2.1. Открыть файл `file.txt` для чтения и считать из него информацию, закрыть файл;

Действие 2.2. Создадим функцию, которая находит среднее арифметическое заданного столбца;

Действие 2.3. Создадим новый файл `filenew.txt`, и запишем в него названия месяцев, зарплату за каждый месяц и среднюю зарплату за заданный месяц.

3. Текст программ

```
//программа, которая создает текстовый файл
#include <cstdio>
#include <cstdlib>
#include <windows.h>
int main()
{FILE *f;double k;   SetConsoleCP(1251);
     SetConsoleOutputCP(1251);
```

```
   char month[12][10]={"январь","февраль","март","апрель",
   "май","июнь","июль","август","сентябрь",
"октябрь","ноябрь","декабрь"};
     f=fopen("file.txt","w");
      for (int j=0;j<12;j++)
             fprintf(f,"%10s",month[j]);
      fprintf(f,"\n");
      for(int i=0;i<18;i++){
         for (int j=0;j<12;j++){
               k=rand()%2000+2000+1/
               double (rand()%10+1);
        fprintf(f,"%10.2f",k);}
        fprintf(f,"\n");
      }fclose(f);
        return 0;
   }
   //программа, которая обрабатывает тек-
//стовый файл
   #include <cstdio>
   #include <cstdlib>
   #include <windows.h>
void readFile(float salary[][12],int
&n,char s[][10]){
   FILE *f=fopen("file.txt","r");
   if(f==0){ printf("error read from file");
       exit(1); }
   for (int j=0;j<12;j++){
        fscanf(f,"%s",&s[j]);
   }
   int i=0,j=0;
   while (fscanf(f,"%f",&salary[i][j++])!=EOF){
```

```
        if (j==12) {i++;j=0;}
   }
   n=i;  fclose(f);
   }
double average(float salary[][12],int n,int
m){
       double s=0;
     for (int i=0;i<n;i++)
           s+=salary[i][m];
     return s/n;
   }
void writeFile(float salary[][12],int
n,char month[][10]){
   FILE *f2=fopen("fileNew.txt","w");
     for (int j=0;j<12;j++)
         fprintf(f2,"%10s",month[j]);
      fprintf(f2,"\n");
      for(int i=0;i<n;i++){
         for (int j=0;j<12;j++)
         fprintf(f2,"%10.2f",salary[i][j]);
         fprintf(f2,"\n"); }
   printf("введите месяц (число): ");
    int m;
   scanf("%d",&m);
   fprintf(f2,"\nСредняя зарплата за %s
месяц \n",month[m-1]);
   for (int j=0;j<12;j++)
   if (j==m-1)
   fprintf(f2,"%10.2f",average(salary,n,m-1));
   else fprintf(f2,"          ");
      fclose(f2);
   }
```

```
int main()
{char s[12][10];
     SetConsoleCP(1251);
     SetConsoleOutputCP(1251);
     float salary[30][12];int n;
readFile(salary,n,s);
writeFile(salary,n,s);
    return 0;
}
```

4. Результат работы программы

Из-за экономии печатных страниц результаты работы программы в методических указаниях не приводятся. При оформлении работы результаты работы программы указать обязательно.

Контрольные вопросы

1. Что такое файл, дескриптор файла, файловый указатель?

2. Какие разновидности файлов вы знаете?

3. В каких режимах можно открыть файл?

4. Что такое структура `FILE` и где она описана?

5. Как работают функции `fopen()` и `fclose()`?

6. С помощью каких функций можно осуществить неформатированный ввод/вывод при использовании структуры `FILE`?

7. С помощью каких функций возможно осуществить форматированный ввод/вывод при использовании структуры `FILE`?

РАБОТА № 6
Разработка и реализация программ для работы с бинарными файлами

Цель работы: овладение навыками составления программ для работы с бинарными файлами и выполнение их в NetBeans IDE.

Задания

Задание 6.1. Запишите строки, которые будут выведены на экран вследствие выполнения фрагментов, представленных в вариантах таблицы 6.1, если программа имеет следующее начало. Указание: вместо N подставить номер варианта по списку группы.

```
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;
int main(){
struct rB{
char name[10];
char ph[5];}; int i,j,nr;
ofstream f; char s[10];
struct {
bool b;
```

```
   union {
         short c;
         short d;};
   }a;
   f.open("LR9_1.txt",ios::binary);
   rB rBk;
   s t r c p y ( r B k . n a m e , " A A " ) ;
strcpy(rBk.ph,"2222"); f.write((char*)&rBk,
sizeof rBk);
   s t r c p y ( r B k . n a m e , " B B B " ) ;
strcpy(rBk.ph,"333"); f.write((char*)&rBk,
sizeof rBk);
   s t r c p y ( r B k . n a m e , " C C " ) ;
strcpy(rBk.ph,"444"); f.write((char*)&rBk,
sizeof rBk);
   f.close();
   ifstream ff ("LR6_1.txt",ios::binary);
   nr=sizeof(rB);
   a.c=N; a.d=10*N;
   cout<<a.c<<"   "<<a.d<<"\n";
```

Таблица 6.1 – Варианты задания 6.1

№	Фрагмент программы

1–5	`for(int i=1; i<3;i++){`
	`  ff.read((char*)&rBk, sizeof rBk);`
	`  cout<<rBk.name<<"  "<<rBk.ph;`
	`}`
	`ff.read((char*)&rBk, sizeof rBk);`
	`  cout<<rBk.ph<<"\n";`
	`  cout<<"n="<<nr;`
	`ff.close();`
	`  cout<<"  "<<rBk.name<<"\n";`

Продолж. табл. 6.1

№	Фрагмент программы

6–10	`for(int i=0; i<3;i++){`
	`  ff.read((char*)&rBk, sizeof rBk);`
	`  cout<<rBk.name<<"  "<<rBk.ph;`
	`}`
	`cout<<"n="<<nr;`
	`cout<<"  "<<rBk.name<<"\n";`
	`ff.read((char*)&rBk, sizeof rBk);`
	`cout<<rBk.ph<<"\n";`
	`ff.close();`

11–15	`for(int i=2; i>1;i--){`
	`  ff.read((char*)&rBk, sizeof rBk);`
	`  cout<<rBk.name<<"  "<<rBk.ph<<"\n";`
	`}`
	`ff.read((char*)&rBk, sizeof rBk);`
	`cout<<rBk.ph;`
	`cout<<"n="<<nr;ff.close();`
	`cout<<"  "<<rBk.name<<"\n";`

16–20	`ff.read((char*)&rBk, sizeof rBk);`
	`cout<<rBk.ph<<" n="<<nr;`
	`for(int i=0; i<2;i++){`
	`  ff.read((char*)&rBk, sizeof rBk);`
	`  cout<<rBk.name<<"  "<<rBk.ph<<"\n";`
	`}`
	`ff.close();`
	`cout<<"  "<<rBk.name<<"\n";`

21–25	`for(int i=0; i<2;i++){`
	`  ff.read((char*)&rBk, sizeof rBk);`
	`  cout<<rBk.name<<"  "<<rBk.ph<<"\n";`
	`}`
	`cout<<" n="<<nr;ff.close();`
	`cout<<"  "<<rBk.name<<"\n";`

26–30	`ff.read((char*)&rBk, sizeof rBk);`
	`for(int i=1; i<3;i++){`
	`  ff.read((char*)&rBk, sizeof rBk);`
	`  cout<<rBk.name<<"  "<<rBk.ph<<"\n";`
	`}`
	`cout<<rBk.ph;`
	`cout<<" n="<<nr;ff.close();`
	`cout<<"  "<<rBk.name<<"\n";`

Задание 6.2. Составить и выполнить программу, которая обрабатывает информацию из файла. Файл надо создать на основе задания 4.2. Выполнить запросы, описанные в задании 4.2.

Указание: реализовать меню с опциями (создать файл, добавить одну запись в файл, добавить несколько записей в файл, удалить файл, удалить одну запись в файле, удалить несколько записей в файле, вывести содержимое файла на экран, вывести содержимое файла на экран согласно заданным критериям).

Краткие теоретические сведения

В языке C++ объекты для работы с файлами называются потоками (streams). В данном случае слово "поток" означает то же самое, что и "файл" в языке C. Классами для работы с файлами в языке C++ являются `fstream`, `ifstream` и `ofstream`.

Для работы с файловыми потоками необходимо подключить библиотеку `<fstream.h>`, которая используется для работы с двунаправленными потоками, эта библиотека определяет классы `ifstream` для работы с входными потоками и `ofstream` для работы с выходными потоками.

В библиотечных файлах потоки описаны как классы, т. е. представляют собой пользовательские типы данных (аналогично структурам данных). В описание класса, кроме данных, добавляются описания функций, обрабатывающих эти данные (соответственно компонентные данные и компонентные функции (методы)). Обращаться к компонентным функциям (методам) можно также, как и к компонентным данным, с помощью операции `"."` (точка) или с помощью операции `"->"`. Основные функции (методы) для работы с файлами приведены в приложении Г.

Использование файлов в программе предполагает следующие операции:

– создание потока;

– открытие файла и связывание его с потоком;

– обмен информацией (ввод/вывод);

– закрытие файла.

Для связи файла с потоком ввода/вывода необходимо объявить объект класса `fstream`, `ifstream` или `ofstream`, передавая конструктору этого объекта имя требуемого файла. Далее следует открыть файл. Например:

```
//создать файловый поток f
ifstream f;
//открыть файл, который необходимо свя-
//зать с потоком
f.open("..\\f.dat",ios::in);
```

Также можно использовать конструктор с параметром:

```
//создать и открыть для чтения файловый
//поток f
ifstream f ("..\\f.dat",ios::in);
```

Вторым параметром в конструкторе является режим открытия файла, перечень которых приведен в приложении Д.

После того как файл открыт, можно считывать/записывать данные в файл, используя операторы ">>" (чтение из потока) и "<<" (вывод в поток), аналогично стандартным потокам `cin` и `cout`.

Ввод/вывод символов в файл/из файла возможно выполнить, используя функции `get()` и `put()`. Считать из файла строку возможно с помощью функции `getline()`. Если необходимо вводить/выводить такие данные, как структуры или массивы, можно использовать функции `read()` и `write()`.

При чтении данных из входного файла необходимо контролировать, был ли достигнут конец файла. Это можно сделать с помощью метода `eof()`. Если в процессе работы возникнет ошибочная ситуация, то потоковый объект при-

нимает значение равное 0. Для проверки ошибок файловых операций можно использовать функцию `fail()`.

После окончания работы с файлом его следует закрыть с помощью метода `close()`. По завершению программы в деструкторе класса этот метод вызовется автоматически, однако, если программе файл больше не нужен, лучше его закрыть вручную, поскольку при этом все данные, которые программа записала в файл, сбрасываются на диск, и обновляется запись каталога для этого файла.

При работе с файлами не следует смешивать различные техники. Например, не нужно в одной и той же программе использовать функции из стандартной библиотеки C и классы-потоки из языка C++.

Пример выполнения работы

Задание 6.1. Пояснить, что выполняет следующий фрагмент программы, где `N=32`.

```
ff.read((char*)&rBk,  sizeof  rBk);
cout<<rBk.name<<"   "<<rBk.ph<<"\n";
ff.read((char*)&rBk,  sizeof  rBk);

  cout<<rBk.name;
  ff.close();
  cout<<"   "<<rBk.ph<<"\n";
```

Решение

Этот фрагмент программы считывает информацию из бинарного файла и выводит на дисплей.

```
320   320
AA   2222
BBB   333
```

Задание 6.2. Составить и выполнить программу, которая сохраняет данные о студентах (фамилия, имя, отчество,

номер группы) в файл, считывает данные из файла и выводит на экран, упорядочивает данные в алфавитном порядке.

Решение

1. Постановка задачи

Составить программу, которая сохраняет данные о студентах (фамилия, имя, отчество, номер группы) в файл, считывает данные из файла и выводит на экран, упорядочивает данные в алфавитном порядке.

2. Алгоритм решения задачи

Действие 1.Выполнять действие 2, действие 3, действие 4, действие 5 в бесконечном цикле.

Действие 2.Если избранное действие 2, тогда:

Действие 2.1. Определить массив структур `sd`;

Действие 2.2. Ввести значение полей структуры;

Действие 2.3. Ввести имя файла;

Действие 2.4. Добавить данные к существующему файлу или создать новый файл;

Действие 3. Если избранное действие 3, тогда:

Действие 3.1. Ввести имя файла;

Действие 3.2. Если файл открывается, перейти к действию 3.3, иначе к главному меню;

Действие 3.3. Определить размер массива структур `sd`;

Действие 3.4. Считать данные из файла в массив структур `sd`;

Действие 3.5. Ввести значение полей массива структур `sd`;

Действие 4. Если избранное действие 4, тогда:

Действие 4.1. Ввести имя файла;

Действие 4.2. Если файл открывается, перейти к действию 4.3, иначе к главному меню;

Действие 4.3. Определить размер массива структур `sd`;

Действие 4.4. Считать данные из файла в массив структур `sd`;

Действие 4.5. Отсортировать массив структур по фамилии;

Действие 4.6. Вывести элементы массива после сортировки;

Действие 4.7. Спросить у пользователя необходимость сохранения упорядоченных данных, если так, тогда перейти к действию 4.8, иначе к главному меню;

Действие 4.8. Перезаписать файл.

Действие 5. Если избранное действие 5, тогда конец работы программы.

3. Текст программы

```
#include <iostream>
#include <cstring>
#include <fstream>
#include <windows.h>
using namespace std;
struct stud{
  char surname[30];
  char name[30];
  char name2[30];
  char grupa[10];};
void enterStruct(stud  sd[], int kol){
    for(int i=0; i < kol; i++){
       cout << "\nФамилия: ";cin >>
sd[i].surname;
       cout << "Имя: "; cin >> sd[i].name;
       cout << "Отчество: "; cin >>
sd[i].name2;
       cout << "Группа: "; cin >>
sd[i].grupa; }
  }
void fileWrite(char* txt,stud  sd[], int
kol) {
    ofstream t(txt,ios::binary | ios::app);
    for(int i=0;i<kol;i++){
```

```
   t.write((char*)&sd[i], sizeof sd[i]); }
    t.close(); }
void printStruct(stud* sd,int kol){
    for(int i = 0; i < kol; i++){
        cout << "\nФамилия: "<<
        sd[i].surname;
        cout << "\nИмя:   "<< sd[i].name;
        cout << "\nОтчество:
        "<<sd[i].name2;
        cout << "\nГруппа:   "<<
        sd[i].grupa<<"\n";}}

int fileOfSize (char *txt) {
    ifstream t(txt,ios::binary);
    if (!t.is_open()){
    cout << "Файл не может быть открытым
    !\n";return 0; }
    t.seekg(0,ios_base::end);
    int k = t.tellg()/sizeof (stud);
    t.close();
    return k;}

void readFile(char *txt,stud* sd,int kol) {
    ifstream t(txt,ios::binary );
    for(int i=0;i<kol;i++){
   t.read((char*)&sd[i], sizeof sd[i]); }
    t.close();}
void rewriteFile(char* txt,stud* sd,int kol)
{
    ofstream t(txt,ios::binary);
    for(int i=0;i<kol;i++){
```

```
      t.write((char*)&sd[i], sizeof
      sd[i]);}
    t.close();}
void sortStud(stud* sd,int kol){
    stud s;
    for(int i=1;i<kol;i++)
    for(int j=kol-1;j>=i;j--)
    if(stricmp(sd[j].surname, sd[j-
1].surname) < 0){
        s = sd[j];
        sd[j]=sd[j-1];
        sd[j-1]=s;
    }
    else if(stricmp(sd[j].surname, sd[j-
1].surname) ==0){
       if(stricmp(sd[j].name, sd[j-
1].name) < 0){
       s = sd[j];
       sd[j]=sd[j-1];
       sd[j-1]=s;
       }
  }}

int main(){
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    char fName[80];
    int k = 0;stud  * sd;
   while(1){
       cout<<"0: Exit\n";
       cout<<"1: Добавить записи в файл\n";
```

```
        cout<<"2: Вывести содержимое файла
             на экран\n";
        cout<<"3:  Сортировка  данных по
             фамилии\n ";
        cin >> k;
        switch(k){
        case 1:
          cout<<"\nВведите количество сту-
             дентов:  ";
   int zn;
        cin>>zn; sd = new stud[zn];
        enterStruct(sd, zn);
        cin.ignore();
        cout << "\nВведите имя файла: ";
        cin.getline(fName,80);
        fileWrite(fName,sd,zn);
        break;
    case 2:
        cout << "\nВведите имя файла: ";
        cin.ignore();
           cin.getline(fName,80);
        zn=fileOfSize(fName);
        if (zn!=0){sd = new stud[zn];
        readFile(fName,sd,zn);
        cout << "\n Содержимое файла: \n";
        printStruct(sd,zn); cin.ignore();}
        break;
    case 3:
        cout << "\n Сортировка данных по
фамилии: \n";
        cout << "\nВведите имя файла: ";
        cin.ignore();
        cin.getline(fName,80);
```

```
        zn=fileOfSize(fName);
        if (zn!=0){sd = new stud[zn];
        readFile(fName,sd,zn);
        sortStud(sd,zn);
        printStruct(sd,zn);
        cout << "\nСохранить отсортирован-
              ные данные? (Y/N): ";
        char tak;cin>>tak;
        if (tak=='Y' || tak =='y')
        rewriteFile(fName,sd,zn);}
        break;
      case 0:
        exit(0);
        break;
           }}
    return 0;
}
```

4. Результаты работы программы

Из-за экономии печатных страниц результаты работы программы в методических указаниях не приводятся. При оформлении работы результаты работы программы указать обязательно.

Контрольные вопросы

1. Какие классы используются для работы с файлами?

2. Как можно обращаться к компонентным данным и компонентным функциям?

3. Какие операции предполагают использование файлов в программе?

4. Как можно связать файл с потоком ввода/вывода?

5. С помощью каких функций можно выполнить файловые операции чтения/записи при использовании потоков?

6. Какие функции существуют для проверки корректности файловых операций при использовании потоков?

РАБОТА № 7
Разработка и реализация программ с использованием директив препроцессора в приложениях

Цель работы: овладение навыками составления программ с использованием директив препроцессора в приложениях и выполнение их в NetBeans IDE.

Задание

Задание 7.1. Определить действие фрагмента программы по вариантам, которые приведены в таблице 7.1.

Таблица 7.1 – Варианты задания 7.1

№	Фрагмент программы

1–5	`#include <iostream>`
	`using namespace std;`
	`# define mkstr(s) #s`
	`int  main()`
	`{  cout<<mkstr(I love C);`
	`return 0;`
	`}`

6–10	`#include <iostream>`
	`using namespace std;`
	`# define concat(a,b) a##b`
	`int main()`
	`{  int ij=30;`
	`cout<<concat(i,j);`
	`return 0;`
	`}`

Продолж. табл. 7.1

№	Фрагмент программы

11–15	`#include <iostream>`
	`using namespace std;`
	`#define ArrFlg 1`
	`int  main ()`
	`{`
	`  #ifdef ArrFlg`
	`  int Arr[30];`
	`  cout << "Array created!";`
	`  #else`
	`  cout << "Array is not defined!";`
	`#endif`
	`  return 0;`
	`}`

16–20	`#include <iostream>`
	`using namespace std;`
	`#if 1`
	`  int Arr[2];`
	`  #else`
	`  float Arr[2];`
	`  #endif`
	`int main ()`
	`  {`
	`  #if  1`
	`  cout<<"int";`
	`  #else`
	`  cout<<"float";`
	`  #endif`
	
	`return 0;`
	`}`

21–25	`#include <iostream>`
	`using namespace std;`
	`#define fl 3`
	`#if fl==1`
	`  int Arr[2];`
	`  #elif fl==2`
	`  float Arr[2];`
	`  #elif fl==3`
	`  double Arr[2];`
	`  #endif`

Продолж. табл. 7.1

№	Фрагмент программы

21–25	`int main (){`
	
	`#if fl==1`
	`cout<<"int";`
	`#elif fl==2`
	`cout<<"float";`
	`#elif fl==3`
	`cout<<"double";`
	`#endif`
	`return 0;`
	`}`

26–30	`#include <iostream>`
	`using namespace std;`
	`int main ()`
	`{`
	`#ifdef ArrFlg`
	`int Arr[10];`
	`cout << "Array created!";`
	`#else`
	`cout << "Array is not defined!";`
	`#endif`
	`return 0;`
	`}`

Задание 7.2. На основе задания 6.2 разработать и выполнить проект, который состоит из нескольких файлов: главного файла, файлов-заголовков и дополнительных модулей. В каждом модуле реализовать отдельные функции. Использовать условную компиляцию.

Краткие теоретические сведения

Препроцессор – это программа, которая производит некоторые манипуляции с первоначальным текстом программы перед тем, как он подвергается компиляции. Препроцессоры создают входной текст для компиляторов и могут выполнять следующие функции:

– обработку макроопределений;
– включение файлов;
– "рациональную" предобработку;
– расширение языка.

Оператор (директива) препроцессора – это одна строка исходного текста, начинающаяся с символа #, за которым следуют название оператора и операнды. Операторы препроцессора могут появляться в любом месте программы, и их действие распространяется на весь исходный файл.

Одно из часто используемых действий препроцесора – включение в исходный текст программы содержимого других файлов. Для включения текста из файла используется команда `#include`, которая имеет две формы записи:

```
#include <имя_файла>// имя в угловых скобках
#include "имя_файла"// имя в кавычках
```

Если имя_файла дается в угловых скобках, то препроцессор ищет файл в стандартных системных каталогах. Если имя_файла заключено в кавычки, то вначале препроцессор просматривает текущий каталог пользователя, а затем обращается к стандартным системным каталогам.

Когда исходный текст программы обрабатывается препроцессором, на место этой инструкции ставится содержимое соответствующего файла.

В практике программирования обычна ситуация, при которой в программе используются функции, тексты которых хранятся в отдельном заголовочном файле `Header File` с расширением `.h`. Препроцессор добавляет тексты всех функций в программу из файла `.h` и как единое целое передает на компиляцию.

Оператор `#define` часто используют для определения символических констант. Он может появиться в любом месте исходного файла, а даваемое им определение имеет силу,

начиная с места появления и до конца файла. В конце определения символической константы (в конце оператора #define) точка с запятой не ставится. Например:

```
# define min 1
# define max 100
```

В тексте программы вместо констант 1 и 100 можно использовать соответственно min и max. Однако, текст внутри строк, символьные константы и комментарии не подлежат замене, т. к. строки и символьные константы являются неделимыми лексемами языка C++. Так что после макроопределения #define YES 1 в операторе cout << "YES"; не будет сделано никакой макроподстановки.

Препроцессор языка C++ позволяет переопределять не только константы, но и целиком программные конструкции. Например, можно написать определение: #define forever for(;;) и затем всюду писать бесконечные циклы в виде forever { <тело цикла> }

С помощью директивы #define также можно создавать макросы, которые используются при встраивании.

Директивы условной компиляции, позволяют генерировать программный код в зависимости от выполнимости определенных условий. Условная компиляция обеспечивается в языке C++ набором команд, которые, по существу, управляют не компиляцией, а препроцессорной обработкой. Общая структура применения директив условной компиляции такова:

```
#if/#ifdef/#ifndef <константное_выражение или идентификатор>
<текст_1>
#else //необязательная директива
<текст_2>
#endif
```

текст_1 включается в компилируемый текст только при истинности проверяемого условия. Если условие ложно, то при наличии директивы #else на компиляцию передается текст_2. Если директива #else отсутствует, то весь текст от #if до #endif при ложном условии опускается.

Директива #if проверяется значение константного целочисленного выражения. Если оно отлично от нуля, то считается, что проверяемое условие истинно. В директиве #ifndef проверяется, определен ли с помощью команды #define к текущему моменту идентификатор, помещенный после #ifndef. Если идентификатор определен, то текст_1 используется компилятором. В директиве #ifndef проверяется обратное условие – истинным считается неопределенность идентификатора, т. е. тот случай, когда идентификатор не был использован в команде #define или его определение было отменено командой #undef.

Директиву #undef удобно использовать при разработке больших программ, когда они собираются из отдельных "кусков текста", написанных в разное время или разными программистами. В этом случае могут встретиться одинаковые обозначения разных объектов. Чтобы не изменять исходных файлов, включаемый текст можно "обрамлять" подходящими директивами #define – #undef и тем самым устранять возможные ошибки.

Для организации мультиветвлений во время обработки препроцессором исходного текста программы введена директива #elif <константное_выражение>, которая является сокращением конструкции #else#if. Препроцессор обрабатывает всегда только один из участков текста, выделенных командами условной компиляции. Перечень директив препроцессора приведен в приложении Ж.

Пример выполнения программы

Задание 7.1. Определить действие фрагмента программы

```
#include <stdio.h>
#define print(x) printf(#x"=%10.5f\n",x);
using namespace std;

int main()
{float a=5.34564, d=235.64456345;
     print(a);print(d);
     return 0;
}
```

Решение

В теле макроса перед параметрами можно задавать символ #, это означает, что при расширении макроса параметры превращаются в строку символов.

Этот фрагмент программы выводит на экран:

```
a=    5.34564
d=  235.64456
```

Задание 7.2. На основе задания 6.2 разработать и выполнить проект, который состоит из нескольких файлов: главного файла, файлов-заголовков и дополнительных модулей. В каждом модуле реализовать отдельные функции. Использовать условную компиляцию.

Решение

1. Постановка задачи

Составить программу, которая определяет структуру "Студент": ФИО, адрес, возраст. Вывести на экран данные об определенном студенте; упорядочить по алфавиту в порядке возрастания.

2. Алгоритм решения задачи

Алгоритм решения задачи совпадает с алгоритмом решения задачи 6.2 и из-за экономии печатных страниц в ме-

тодических указаниях не приводится. При оформлении работы алгоритм решения задания приводить обязательно.

3. Текст программ

main.cpp

```
#include <iostream>
#include <windows.h>
#include "student.h"
using namespace std;
int main(){
 SetConsoleCP(1251);
 SetConsoleOutputCP(1251);
 char fName[80];
 int k = 0;stud  * sd;
 while(1){
   cout<<"0: Exit\n";
   cout<<"1: Добавить записи в файл\n";
   cout<<"2: Вывести содержимое файла
        на экран\n";
   cout<<"3: Сортировка данных по
        фамилии\n ";
   cin >> k;
   switch(k){
   case 1:
     cout<<"\nВведите количество студентов: ";
int zn;
     cin>>zn; sd = new stud[zn];
     enterStruct(sd, zn);
     cout << "\nВведите имя файла:";
     cin.ignore();
     cin.getline(fName,80);
     fileWrite(fName,sd,zn);
     break;
```

```
      case 2:
        cout << "\nВведите имя файла:";
        cin.ignore();
        cin.getline(fName,80);
        zn=fileOfSize(fName);
        if (zn!=0){sd = new stud[zn];
        readFile(fName,sd,zn);
        cout << "\n Содержимое файла: \n";
        printStruct(sd,zn);getchar();}
        break;
      case 3:
        cout << "\n Сортировка данных по
фамилии: \n";
        cout << "\nВведите имя файла:";
        cin.ignore();
        cin.getline(fName,80);
        zn=fileOfSize(fName);
        if (zn!=0){sd = new stud[zn];
        readFile(fName,sd,zn);
        sortStud(sd,zn);
        printStruct(sd,zn);
        cout << "\nСохранить отсортирован-
ные данные? (Y/N): ";
        char tak;cin>>tak;
        if (tak=='Y' || tak =='y')
        rewriteFile(fName,sd,zn);}
        break;
      case 0:
        exit(0);
        break;
      }}
   return 0;
   }
```

student.h

```
#ifndef STUDENT_H_INCLUDED
#define STUDENT_H_INCLUDED
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;
struct stud{
   char surname[30];
   char name[30];
   char name2[30];
   char grupa[10];};
void enterStruct(stud *, int );
void fileWrite(char* ,stud *, int );
void printStruct(stud* ,int );
int fileOfSize (char *);
void readFile(char *,stud* ,int );
void rewriteFile(char* ,stud* ,int );
void sortStud(stud* ,int );
#endif // STUDENT_H_INCLUDED
```

student.cpp

```
# include "student.h"
void enterStruct(stud  sd[], int kol){
     for(int i=0; i < kol; i++){
       cout << "\nФамилия: ";cin >>
sd[i].surname;
       cout << "Имя: "; cin >> sd[i].name;
       cout << "Отчество: "; cin >>
sd[i].name2;
       cout << "Группа: "; cin >>
sd[i].grupa; }
   }
void fileWrite(char* txt,stud  sd[], int
kol) {
```

```
   ofstream t(txt,ios::binary | ios::app);
   for(int  i=0;i<kol;i++){
        t.write((char*)&sd[i], sizeof sd[i]); }
      t.close();  }
void  printStruct(stud*  sd,int  kol){
      for(int i = 0; i < kol; i++){
        cout << "\nФамилия: "<< sd[i].surname;
        cout  <<  "\nИмя:   "<<  sd[i].name;
        cout << "\nОтчество: "<<sd[i].name2;
        cout << "\nГруппа: "<< sd[i].grupa<<"\n";}}

int  fileOfSize  (char *txt)  {
      ifstream  t(txt,ios::binary);
      if  (!t.is_open()){
      cout << " Файл не может быть открытым
!\n";return  0;  }
      t.seekg(0,ios_base::end);
      int  k  =  t.tellg()/sizeof  (stud);
      t.close();
      return  k;}

void  readFile(char  *txt,stud*  sd,int  kol)  {
      ifstream  t(txt,ios::binary  );
      for(int  i=0;i<kol;i++){
    t.read((char*)&sd[i], sizeof  sd[i]);  }
      t.close();}
void rewriteFile(char* txt,stud* sd,int kol)
{
      ofstream  t(txt,ios::binary);
      for(int  i=0;i<kol;i++){
        t.write((char*)&sd[i], sizeof sd[i]);}
      t.close();}
```

```
void sortStud(stud* sd,int kol){
      stud s;
      for(int i=1;i<kol;i++)
      for(int j=kol-1;j>=i;j--)
      if(stricmp(sd[j].surname, sd[j-
1].surname) < 0){
      s = sd[j];
      sd[j]=sd[j-1];
      sd[j-1]=s;
      }
      else if(stricmp(sd[j].surname, sd[j-
1].surname) ==0){
        if(stricmp(sd[j].name, sd[j-
         1].name) < 0){
                    s = sd[j];
                    sd[j]=sd[j-1];
                    sd[j-1]=s;
        }
   }}
```

4. Результаты работы программы

Из-за экономии печатных страниц результаты работы программы в методических указаниях не приводятся. При оформлении работы результаты работы программы приводятся обязательно.

Контрольные вопросы

1. Что такое препроцессор?
2. Что такое директива препроцессора?
3. Назовите основные директивы препроцессора.
4. Что делает директива препроцессора `#define`?
5. Для чего используется директива препроцессора `#elif`?
6. Что делает директива препроцессора `#undef`?

РАБОТА № 8
Разработка и реализация программ для работы с однонаправленными списками

Цель работы: овладение навыками составления программ для работы с однонаправленными списками и выполнение их в NetBeans IDE.

Задания

Задание 8.1. Запишите строки, которые будут выведены на экран вследствие выполнения фрагментов, представленных в вариантах таблицы 8.1, при следующем начале программы (Указание: вместо N подставить номер варианта по списку группы):

```
#include <stdio.h>
#include <string.h>
 struct lnk{
     char name[10];
     int ph;
     lnk *next;
     };
int main(){
int i, nr; short int n;
short int *k, *p;
lnk *cR,*fst,*p1; char *nAr[3];
int pAr[3];
```

```
scanf("%hd",&n);k=&n;p=k;*p=*p+2;
printf("%p  %hd\n",k,*p);
nAr[0]="AAA";nAr[1]="BBBB";nAr[2]="CCCC";
pAr[0]=2222;pAr[1]=333;pAr[2]=4444;
nr=sizeof  (lnk);
fst=NULL;
for  (i=0;  i<3;i++){
       cR = new lnk;
        strcpy(cR->name,nAr[i]);
       cR->ph=pAr[i];
        cR->next=fst;  fst=cR;
}
```

Таблица 8.1 – Варианты задания 8.1

№	Фрагмент программы

1–5	`fst->next=fst->next->next;`
	`printf("%d \n",cR->ph); cR=fst;`
	`while (cR!=NULL){`
	`  printf("%s %d\n",cR->name,cR->ph);`
	`  cR=cR->next;`
	`}`
	`printf("nr=%d",nr);`
	`printf(" %s\n",fst->name);`

6–10	`fst=fst->next;`
	`printf("%d \n",cR->ph);cR=fst;`
	`while ( cR!=NULL){`
	`printf("%s  %d \n",cR->name,cR->ph);`
	`cR=cR->next;`
	`}`
	`printf("nr=%d",nr);`
	`printf("  %s\n",fst->name);`

11–15	`p1=new lnk;`
	`strcpy(p1->name,nAr[0]);p1->ph=pAr[0];`
	`p1->next=fst->next; fst->next=p1;`
	`printf("%d\n",cR->ph);cR=fst;`
	`for (i=2;i>0;i--){`
	`printf("%s  %d\n",cR->name,cR->ph);`
	`cR=cR->next;`

Продолж. табл. 8.1

№	Фрагмент программы

11–15	`}`
	`printf("nr=%d",nr);`
	`printf("  %s\n",fst->name);`

16–20	`p1=new lnk;`
	`strcpy(p1->name,nAr[1]);p1->ph=pAr[1];`
	`p1->next=fst->next; fst->next=p1;`
	`printf("%d\n",cR->ph);cR=fst;`
	`for (i=0;i<3;i++){`
	`printf("%s  %d\n",cR->name,cR->ph);`
	`cR=cR->next;`
	`}`
	`printf("nr=%d",nr);`
	`printf("  %s\n",fst->name);`

21–25	`printf("%d\n",cR->ph);cR=fst;`
	`for (i=1;i<3;i++){`
	`printf("%s  %d\n",cR->name,cR->ph);`
	`cR=cR->next;`
	`}`
	`printf("nr=%d",nr);`
	`printf("  %s\n",fst->name);`

26–30	`printf("%d\n",cR->ph);cR=fst;`
	`for (i=2;i>=0;i--){`
	`printf("%s  %d\n",cR->name,cR->ph);`
	`cR=cR->next;`
	`}`
	`printf("nr=%d",nr);`
	`printf("  %s\n",fst->name);`

Задание 8.2. На основе задания 6.2 разработать и выполнить программу, которая будет считывать информацию из файла в динамическую связанную структуру – связанный список. Массивы не использовать!

Краткие теоретические сведения

Список – это структура (составная переменная, класс), которая содержит обычные переменные (поля данных) и адрес следующей такой структуры (указатель на нее) в случае однонаправленного списка. В случае двунаправленного

списка структура содержит два указателя – на следующую и предыдущую структуры. Каждый элемент списка содержит информационное поле (или поля) и ссылочное поле. В первом элементе списка ссылочное поле с указателем на предыдущую структуру и в последнем элементе списка ссылочное поле с указателем на следующую структуру содержат нули (NULL). В список легко вставить элемент или исключить элемент из списка, откорректировав значения ссылочного поля предыдущего (или последующего) элемента. Для работы со списком обычно заводят несколько указателей: на первый элемент, на текущий элемент, на последний элемент.

Динамическая реализация однонаправленного списка основана на динамическом выделении и освобождении памяти для элементов списка. Логическая последовательность элементов списка создается ссылочными полями с адресами последующих элементов (последний элемент имеет пустую ссылку NULL).

Для удобства реализации считается, что список всегда содержит хотя бы один элемент-заголовок с адресом первого реального элемента списка. Это позволяет унифицировать процедуры добавления и удаления крайних элементов и устранить некоторые проверки. Адрес элемента-заголовка задается переменной-указателем Head. Эта переменная устанавливается при первоначальном создании списка и в дальнейшем не изменяется. Для реализации основных действий используются вспомогательные ссылочные переменные.

Необходимые объявления:

```
struct ListItem {
int Info;
ListItem *Next;
};
ListItem *Head;
```

Создание пустого списка включает:

– выделение памяти под заголовок:

```
Head = new ListItem;
```

– установку пустой ссылочной части заголовка:

```
Head->Next = NULL;
```

Поиск заданного элемента включает:

– установку вспомогательного указателя на адрес первого элемента списка;

– организацию цикла прохода по списку с завершением, либо по совпадению информационной части элемента с заданным значением, либо по достижению конца списка. После завершения цикла необходимо проверить значение вспомогательного указателя и сделать вывод об успешности поиска.

```
ListItem *Current = Head->Next;
while (Current != NULL && Current->Info
!= Info)
  Current=Current->Next;
if(Current==NULL)
{…
//ничего не найдено;
}
else
{…
//элемент найден;
}
```

Удаление заданного элемента включает:

– поиск удаляемого элемента с определением адреса элемента-предшественника. Для этого вводится еще одна вспомогательная ссылочная переменная `Previous`, инициированная адресом заголовка и изменяющая свое значение внутри цикла. Если удаляемый элемент найден, то изменяется ссылочная часть его предшественника:

```
Previous->Next = Current->Next;
```

– удаляемый элемент обрабатывается необходимым образом, т. е. либо освобождается занимаемая им память, либо он включается во вспомогательный список.

Добавление нового элемента после заданного включает:

– поиск заданного элемента с помощью вспомогательного указателя `Current`;

– выделение памяти для нового элемента с помощью еще одного указателя `Tmp`;

– формирование полей нового элемента, в частности – настройка ссылочной части:

```
Tmp->Next = Current->Next;
```

– изменение ссылочной части текущего элемента на адрес нового элемента:

```
Current->Next = Tmp;
```

Добавление нового элемента перед заданным включает в себя:

– поиск заданного элемента с одновременным определением элемента-предшественника (используются указатели `Current и Previous`);

– выделение памяти для нового элемента с помощью еще одного указателя `Tmp`;

– формирование полей нового элемента, в частности – настройка ссылочной части:

```
Tmp->Next = Current;
```

– изменение ссылочной части элемента-предшественника на адрес нового элемента:

```
Previous->Next = Tmp;
```

Если при использовании списка часто приходится добавлять или удалять элементы в конце списка, то для уменьшения расходов на просмотр всего списка можно ввести второй основной указатель – на последний элемент списка. Это

потребует изменения процедур удаления и добавления для отслеживания этого указателя.

Часто используется упорядоченная разновидность однонаправленного списка, в котором элементы выстраиваются в соответствии с заданным порядком, например: целые числа по возрастанию или текстовые строки по алфавиту. Для таких списков изменяется процедура добавления: новый элемент должен вставляться в соответствующее место для сохранения порядка элементов. Например, если порядок элементов определяется целыми числами по возрастанию, то при поиске подходящего места надо найти первый элемент, больший заданного и выполнить вставку перед этим элементом.

Пример выполнения работы

Задание 8.1. Определить действие фрагмента программы

```
printf("%s\n",cR->name);p1=fst->next;
while  (p1->next){
  printf("%s   %d\n",p1->name,p1->ph);
  p1=p1->next;
}
printf("nr=%d",nr);
printf("  %s   %d\n",p1->name,p1->ph);
```

Решение

Этот фрагмент программы выводит на экран в первой строке значение переменной k, а именно адрес и значение по указателю p, который указывает на n. Во второй строке выводится значение информационного поля последнего элемента списка, дальше выводятся значения двух информационных полей предыдущего элемента. В последней строке приведен размер элемента списка и значение полей первого элемента списка.

```
0028FF02  6
```

```
CCCC
BBBB   333
nr=20 AAA   2222
```

Задание 8.2. Разработать программу, которая обрабатывает линейный список: удалить все элементы, которые больше, чем среднее арифметическое значение.

Решение

1. Постановка задачи

Разработать программу, которая обрабатывает линейный список: удалить все элементы, которые больше чем среднее арифметическое значение.

2. Алгоритм решения задачи

Алгоритм решения задачи можно представить в виде следующей последовательности действий:

Действие 1. Создать список.

Действие 2. Вывести список на экран.

Действие 3. Найти среднее арифметическое значение элементов списка.

Действие 4. Повторять до конца списка:

Действие 4.1. Найти адрес элемента списка, у которого значение информационного поля больше, чем среднее арифметическое значение;

Действие 4.2. Если адрес не нуль, тогда удалить элемент списка по этому адресу;

Действие 5. Вывести измененный список на экран.

Действие 6. Удалить весь список.

3. Текст программы

```
#include <iostream>
#include <windows.h>
using namespace std;
struct element {
    int x;
```

```
      element *next;
   };
void add(int x, element *head){
       element * t = new element;
       t->x = x;
       t->next = head->next;
       head->next = t;
   }
void print(element *head) {
        cout << "-----------------------"
        << endl;
       element * t = head->next;
       while (t != NULL) {
           cout << t->x << " ";
           t = t->next;
       }
       cout << endl << "------------------
       ------" << endl;
   }
element * findAbove(element *head, float
value) {
       element *p;
       p = head;
       while(p && (p->x <= value)) {
            p = p->next;
       }
       return p;
   }
float average(element *head) {
      element *p;
      int sum=0;int k=0;
      p = head->next;
```

```
   while (p) {
        sum+=p->x;k++;
        p=p->next;
   }
   return  float(sum)/k;
   }
void deleteList(element *head) {
   while (head->next) {
        element * t = head->next;
        head->next = t->next;
        delete t;
   }
   }
void deleteElement(element *p) {
        p->x = p->next->x;
        element *t = p->next;
        p->next = p->next->next;
        delete t;}

int main() {
        SetConsoleCP(1251);
        SetConsoleOutputCP(1251);
   element *head = new element;
   int k;
   for (int i=0;i<10;i++){
        k=rand()%10;
        add(k, head);
        }
   print(head);
   float av=average(head);
   cout<<"\n average="<<av<< endl;
   element *temp=head->next;
```

```
while  (temp!=NULL){
     temp  =  findAbove(temp,  av);
    if  (temp)  deleteElement(temp);
   }
print(head);
deleteList(head);
     return 0;
}
```

4. Результат работы программы

```
------------------------
4  2  8  8  4  9  0  4  7  1
------------------------
  average=4.7
------------------------
4  2  4  0  4  1
------------------------
```

Контрольные вопросы

1. Что такое список и элемент списка?
2. В чем отличие между однонаправленным и двунаправленным списком?
3. Как создать пустой список?
4. Как происходит поиск заданного элемента списка?
5. Как происходит удаление заданного элемента списка?
6. Как происходит добавление нового элемента списка после заданного элемента?
7. Как происходит добавление нового элемента списка перед заданным элементом?

РАБОТА № 9
Разработка и реализация программ с использованием рекурсивных алгоритмов

Цель работы: овладение навыками составления программ с использованием рекурсивных алгоритмов и выполнение их в NetBeans IDE.

Задания

Задание 9.1. Определить действие фрагмента программы согласно вариантам, которые приведены в таблице 9.1. Указание: вместо N подставить номер варианта по списку группы.

Таблица 9.1 – Варианты задания 9.1

№	Фрагмент программы

1–5	`#include <iostream>`
	`using namespace std;`
	`int f(int n) {`
	`  if (n < 10)  return n;`
	`  else  return n % 10 + f(n / 10);`
	`  }`
	`int main() {`
	`  intch=1235+N;`
	`  cout<<"f="<<f(ch);`
	`return 0;`
	`}`

6–10	`#include <iostream>`
	`using namespace std;`
	`int f(int n, inti) {`
	`  if (n <= 1)  return false;`

Продолж. табл. 9.1

№	Фрагмент программы

6–10	`else if (n == 2)  return true;`
	`     else if (n % i == 0)   return false;`
	`else`
	`              if (i< n / 2)return f(n, i + 1);`
	`              else    return true;`
	`}`
	`int main() {`
	`cout<<"f="<<f(10+N,2)<<"\n";`
	`cout<<"f="<<f(11,2)<<"\n";`
	`cout<<"f="<<f(12,2)<<"\n";`
	`return 0;`
	`}`

11–15	`#include <iostream>`
	`using namespace std;`
	`void f(int n, int k) {`
	`   if (k > n / 2) {`
	`cout<<n<<"  ";`
	`              return;`
	`         }`
	`    if (n % k == 0) {`
	`cout<<k<<"  ";`
	`           f(n / k, k);`
	`         }`
	`         else f(n, ++k);`
	
	`}`
	`int main() {`
	`      f(125+N,2);return 0;`
	`}`

16–20	`#include <iostream>`
	`using namespace std;`
	`int f(int n, int k) {`
	`   if (n==0)  return k;`
	`   else    f(n/10, k*10+n%10);`
	`}`
	`int main() {`
	`cout<<"f="<<f(N,0)<<"\n";`
	`cout<<"f="<<f(125,0)<<"\n";`
	`cout<<"f="<<f(4568,0)<<"\n";`
	`return 0;`
	`}`

Продолж. табл. 9.1

№	Фрагмент программы

21–25	`#include <iostream>`
	`#include <cmath>`
	`using namespace std;`
	`float f(int n) {`
	`if (n <=1) return 1;`
	`else return (1/sqrt(n)+f(--n));`
	`}`
	`int main() {int n=10-N%10;`
	`cout<<"f="<<f(n)<<"\n";`
	`return 0;`
	`}`

26–30	`#include <iostream>`
	`#include <limits.h>`
	`using namespace std;`
	`int f() {int a;`
	`cin >>a;`
	`if (a ==0) return INT_MIN;`
	`else { int t=f();`
	`return (a>t)? a : t;}`
	`}`
	`int main() {`
	`cout<<"f="<<f()<<"\n"; return 0;}`

Задание 9.2. Составить программу нахождения следующих величин согласно вариантам, которые приведены в таблице 9.2. Вместо циклических операторов использовать рекурсию!

Таблица 9.2 – Варианты задания 9.2

№	Варианты задания
1–2	Найти наибольший общий делитель (НОД) чисел x и y. $\text{НОД}(x,y) = \begin{cases} x, \text{если } x = y \\ \text{НОД}(x-y,y), \text{если } x > y \\ \text{НОД}(x,y-x), \text{если } x < y \end{cases}$
3–4	Используя рекурсивную функцию, найти n-ый элемент последовательности $x_n = 5 * x_{n-1}$, $x_0 = 1$
5–6	Используя рекурсивную функцию, найти сумму первых n элементов последовательности $x_n = 3 * x_{n-1}+1$, $x_0 = 0$

Продолж. табл. 9.2

№	Варианты задания
7–8	Используя рекурсивную функцию, найти произведение первых n элементов последовательности $x_n = 3 * x_{n-1}+1$, $x_0 = 1$
9–10	Вводится с клавиатуры последовательность ненулевых действительных чисел, которая заканчивается `0`. Используя рекурсивную функцию, найти количество отрицательных чисел
11–12	Пусть с клавиатуры вводится некоторая последовательность символов (учитывая пробел). Используя рекурсивную функцию, найти количество латинских букв
13–14	Пусть с клавиатуры вводится некоторая последовательность символов (учитывая пробел). Используя рекурсивную функцию, найти количество цифр
15–16	Пусть с клавиатуры вводится некоторая последовательность символов (учитывая пробел). Используя рекурсивную функцию, найти сумму кодов символов
17–18	Пусть с клавиатуры вводится некоторая последовательность символов. Описать рекурсивную функцию, которая превращает последовательность цифр, которые находятся среди символов в целое число. Например, вводится последовательность `апр456ро 78`, получим число `45678`
19–20	Вводится с клавиатуры последовательность ненулевых действительных чисел, которая заканчивается `0`. Используя рекурсивную функцию, найти среднее арифметическое чисел
21–22	Используя рекурсивную функцию, вычислить значение по формуле, где n вводится с клавиатуры: $\sqrt{2+\sqrt{4+\cdots+\sqrt{2n}}}$
23-24	Используя рекурсивную функцию, вычислить значение по формуле, где n вводится с клавиатуры: $\sqrt{1+\sqrt{3+\cdots+\sqrt{2n+1}}}$
25–26	Используя рекурсивную функцию, вычислить значение по формуле, где n вводится с клавиатуры: $\cfrac{1}{1+\cfrac{1}{2+\cfrac{1}{3+\cdots+\cfrac{1}{n}}}}$

Продолж. табл. 9.2

№	Варианты задания
27–28	Используя рекурсивную функцию, вычислить значение по формуле, где n вводится с клавиатуры: $\cfrac{1}{1+\cfrac{1}{3+\cfrac{1}{5+\cdots+\cfrac{1}{2n+1}}}}$
29–30	Используя рекурсивную функцию, перевернуть строку (вывести в обратном порядке)

Краткие теоретические сведения

Рекурсивной называется функция, если во время ее выполнения происходит повторный ее вызов: непосредственно в теле функции (прямой вызов) или путем вызова цепочки других функций, которые содержат обращение к искомой функции (непрямой вызов). Все функции вызываемой цепочки функций также называются рекурсивными.

Когда функция рекурсивно вызывается, в стеке создается копия значений ее локальных переменных и параметров, функция выполняется сначала. При повторном вызове этот процесс повторяется. Глубина рекурсии, т. е. количество вызовов, в C++ не ограничивается. Реально она зависит от ресурсов памяти (размера стека).

В общем случае рекурсивность – это не свойство самой функции, а свойство ее описания. Рекурсивная функция, как правило, менее короткая и более наглядная, чем нерекурсивная, но при выполнении требует больше времени и памяти за счет повторных обращений к самой себе и дублированию локальных переменных. Необходимо хорошо понимать, что каждый очередной рекурсивный вызов приводит к созданию новой копии локальных объектов функции и все эти копии, которые соответствуют цепочке активизированных и незавершенных рекурсивных вызовов, существуют независимо друг от друга и хранятся в стеке. Это может привести к так называемому

"взрыву" стека. "Взрыв" стека означает, что для записи локальных переменных функции не хватает памяти, которая отводится под стек.

В рекурсивных функциях можно выделить две серии шагов:

– рекурсивное погружение функции до тех пор, пока выбранный параметр не достигнет предельного значения. Если такое предельное значение не задать, рекурсивная функция создаст бесконечную последовательность вызовов самой себя.

– рекурсивный вызов до тех пор, пока выбранный параметр не достигнет начального значения. Это обеспечивает получение промежуточных и конечного результатов.

Наиболее типичные примеры рекурсивных функций – вычисление факториала и возведение в степень.

```
// вычисление факториала n
int fact(int n) {
if (n==0) return 1;
else
return n*fact(n-1);
}
```

Обычно рекурсивные функции несколько уменьшают размер кода и повышают эффективность использования памяти по сравнению со своими итеративными аналогами, однако основное преимущество рекурсивных функций состоит в упрощении и наглядности алгоритмов. Сравнительная характеристика рекурсивных и итеративных алгоритмов приведена в приложении 3.

Пример выполнения работы

Задание 9.1. Определить действие фрагмента программы

```
#include <iostream>
#include <limits.h>
using namespace std;
int  f() {int n;
      cin >>n;
```

```
        if (n ==0) return INT_MAX;
        else { int t=f();
               return (n<t)? n : t;}
 }
int main() {
cout<<"f="<<f()<<"\n";
return 0;}
```

Решение

Этот фрагмент программы находит минимальное значение последовательности чисел. Последовательность будет завершена, если введено с клавиатуры `0`.

```
6
3
5
1
9
0
f=1
```

Задание 9.2. Используя рекурсивную функцию, вычислить количество пробелов в строке, которая считывается с консоли.

Решение

1. Постановка задачи

Используя рекурсивную функцию, вычислить количество пробелов в строке, которая считывается с консоли.

2. Алгоритм решения задачи

Алгоритм решения задачи можно представить в виде такой последовательности действий:

Действие 1. Считать строку `sentence` с консоли.

Действие 2. Повторять следующие действия :

Действие 2.1. Если текущий символ строки `sentence` равняется `\0`, то конец рекурсии;

Действие 2.2. Если текущий символ строки `sentence` равняется не пробел, тогда перейти к следующему символу;

Действие 2.3. Если текущий символ строки `sentence` равняется пробел, тогда увеличить результат на 1 и перейти к следующему символу.

Действие 3. Вывести результат работы рекурсивной функции на экран.

3. Текст программы

```
#include <cstdio>
int countW(char *str){
 if (*str == '\0' )
     return 0;
 if (*str !=' ') return countW(++str);
         else  return countW(++str)+1;
}
int main(){
char sentence[80];
printf( "Enter text:\n" );
gets( sentence);
printf( "Result:\n" );
printf("%d",countW(sentence));
return 0;
}
```

4. Результат работы программы

```
Enter text:
word w o r d
Result:
4
```

Контрольные вопросы

1. Какая функция называется рекурсивной?
2. Что такое прямой и косвенный вызов рекурсии?
3. Что такое глубина рекурсии?
4. Назовите серии шагов в рекурсивной функции.
5. Назовите преимущества рекурсивного описания функций.
6. Назовите особенности описания рекурсивной функции.

РАБОТА № 10
Разработка и реализация программ с использованием битовых операций

Цель работы: овладение навыками составления программ с использованием битовых операций и выполнение их в NetBeans IDE.

Задания

Задание 10.1. Определить действие фрагмента программы согласно вариантам, которые приведены в таблице 10.1. Указание: вместо `N` подставить номер варианта по списку группы.

Таблица 10.1 – Варианты задания 10.1

№	Фрагмент программы

1–5	`#include<stdio.h>`
	`#include <limits.h>`
	`int main ()`
	`{size_t len = sizeof(int) * CHAR_BIT;`
	`unsigned int value=N;`
	`const unsigned int mask = 1 << len-1;`
	`printf ("\nThe bitwise representation is: ");`
	`for (int i = 0; i < len; i++)`
	`{`
	`putchar( mask & value ? '1' : '0');`
	`value <<=1;`
	`if ( (i+1) % 8 == 0)`
	`putchar (' ');`
	`}`
	`return 0;`
	`}`

Продолж. табл. 10.1

№	Фрагмент программы

6–10	`#include <stdio.h>`
	`int main () {`
	`unsigned int a= N & 0xFFFE;`
	`int b;`
	`printf ("The shifted number is %u\n", a);`
	`printf ("Enter the shift direction: left (0)`
	`and right (1) \n");`
	`scanf ("%d", &b);`
	`if (!b)`
	`a = a << ((N^3)&(N|3));`
	`else`
	`a = a >> 1;`
	`printf ("The shifted number is %u\n", a);`
	`return 0;`
	`}`

11–15	`#include <stdio.h>`
	`int f(int x){`
	`unsigned mask = 0x80000000;`
	`if (x &  mask) return 1;`
	`  else return 0;`
	`}`
	`int main () {`
	`printf ("The result is %d\n",f(N));`
	`printf ("The result is %d\n",f(-N));`
	`printf ("The result is %d\n",f(-2*N));`
	`printf ("The result is %d\n",f(2147483647));`
	`return 0;`
	`}`

16–20	`#include <stdio.h>`
	`int f(int x){`
	`unsigned mask = 0x1;`
	`return (x &  mask);`
	`}`
	`int main () {`
	`printf ("The result is %d\n",f(N));`
	`printf ("The result is %d\n",f(2*N+1));`
	`printf ("The result is %d\n",f(2*N));`
	`return 0;`
	`}`

Продолж. табл. 10.1

№	Фрагмент программы

21–25	`#include <stdio.h>`
	`int f(int x){`
	`unsigned int mask = x >> 31;`
	`return (x + mask)^ mask;`
	`}`
	`int main () {`
	`printf ("The result is %d\n",f(N));`
	`printf ("The result is %d\n",f(-N));`
	`printf ("The result is %d\n",f(0));`
	`return 0;`
	`}`

26–30	`#include <stdio.h>`
	`int f(int x){`
	`if (x && !(x & (x-1))) return 1;`
	`  else return 0;`
	`}`
	`int main () {`
	`printf ("The result is %d\n",f(N));`
	`printf ("The result is %d\n",f(2));`
	`printf ("The result is %d\n",f(4));`
	`printf ("The result is %d\n",f(6));`
	`printf ("The result is %d\n",f(32));`
	`return 0;`
	`}`

Задание 10.2. Составить программу нахождения следующих величин согласно вариантам, которые приведены в таблице 10.2. Задания решать с помощью бинарных операций.

Таблица 10.2 – Варианты задания 10.2

№	Величины, которые нужно найти
1–3	Дано целое число A и натуральное число n. Выведите число, которое состоит только из n последних битчисла A (замените на 0 все биты числа A, кроме последних n). Результат вывести в 2, 8, 16 и 10 системах счисления
4–6	Дано число $n<32$. Запишите число 2^n, т. е. число, у которого n-й битравняется 1, а остальные – нули. Результат вывести в 2, 8, 16 и 10 системах счисления

Продолж. табл. 10.2

№	Величины, которые нужно найти
7–9	Дано целое число A и натуральное число x. Запишите нули в последние x бит числа A. Результат вывести в 2, 8, 16 и 10 системах счисления
10–12	Дано целое число A и натуральное число x. Выведите число, в которое преобразуется число A, если установить значение x-го бита равному 1. Результат вывести в 2, 8, 16 и 10 системах счисления
13–15	Дано два неравных числа: n и m, что не превышают 31. Вычислите $2^n + 2^m$. Результат вывести в 2, 8, 16 и 10 системах счисления
16–18	Дано целое число A и натуральное число x. Выведите число, в которое преобразуется число A, если инвертировать x-й бит. Результат вывести в 2, 8, 16 и 10 системах счисления
19–21	Дано целое число A, пользуясь бинарными операциями, определить знак числа (число является отрицательным или нет). Число A вывести в 2, 8, 16 и 10 системах счисления
22–24	Дано целое число A и натуральное число x. Выведите число, в которое преобразуется число A, если установить значение x-го бита равному 0. Результат вывести в 2, 8, 16 и 10 системах счисления
25–27	Дано целое число A и натуральное число x. Необходимо получить значение x-го бита числа A. Число A вывести в 2, 8, 16 и 10 системах счисления
28–30	Дано целое число A. Необходимо подсчитать количество единичных бит, которые содержит число A. Результат и число A вывести в 2, 8, 16 и 10 системах счисления

Краткие теоретические сведения

В языке C++ существует ряд операций, выполняющихся над разрядами. Они носят название битовые операции:

– унарная операция: инверсия битов (~);

– бинарные операции: битовое "И" (&), битовое "ИЛИ" (|), битовое исключающее "ИЛИ" (^), сдвиг влево (<<), сдвиг вправо (>>).

Инверсия битов (поразрядное отрицание, дополнение до единицы) инвертирует биты, т. е. каждый бит со значением 1 получает значение 0 и наоборот.

Битовое "И" сравнивает последовательно разряд за разрядом два операнда. Для каждого разряда результат равен 1, тогда и только тогда, когда оба соответствующих разряда операндов равны 1. Так, например, `10010011 & 00111101 = 00010001`, потому, что только нулевой и четвертый разряды обоих операндов содержат 1.

Битовое "ИЛИ" сравнивает последовательно разряд за разрядом два своих операндов. Для каждого разряда результат равен 1 тогда и только тогда, когда любой из соответствующих разрядов операндов равны 1. Так, например, `10010011 | 00111101 = 10111111`, потому, что все разряды (кроме шестого) в одном из двух операндов имеют значение 1.

Битовое исключающее "ИЛИ" сравнивает последовательно разряд за разрядом два своих операндов. Для каждого разряда результат равен 1, если один из двух (но не оба) соответствующих разрядов операндов равен 1. Так, например, `10010011 ^ 00111101 = 10101110`. Поскольку нулевой разряд в обоих операндах имеет значение 1, нулевой разряд результата имеет значение 0.

Описанные выше операции часто используются для установки некоторых битов, причем другие биты остаются неизменными. Они удобны для фильтрации или маскирования битов.

Сдвиг влево сдвигает разряды левого операнда влево на число позиций, указанное правым операндом. Освобождающиеся позиции заполняются нулями, а разряды, сдвигаемые за левый предел левого операнда, теряются. Поэтому, например, `10001010 << 2 = 00101000` (каждый разряд сдвинулся на две позиции влево).

Таким образом, `x<<2` сдвигает `x` влево на 2 позиции, заполняя освобождающиеся позиции нулями (эквивалентно умножению на 4).

Сдвиг вправо сдвигает разряды левого операнда вправо на число позиций, указанное правым операндом. Освобождающиеся позиции заполняются нулями, а разряды, сдвигаемые за правый предел левого операнда, теряются. Для значений без знака имеем `10001010 >> 2 = 00100010`, каждый разряд сдвинулся на две позиции вправо.

Эти две операции выполняют сдвиг, а также эффективное умножение и деление на степени числа 2.

Битовые поля в типичных применениях служат для хранения целых данных, чаще типа `unsigned`. Описание поля битов состоит из описания типа поля, его имени и указанного после двоеточия размера поля в битах, например: `unsigned status: 6;`

Если имя поля опущено, то создается скрытое поле. Если размер поля битов представлен числом 0, то следующее поле битов начнется с границы машинного слова.

Пример выполнения работы

Задание 10.1. Определить действие фрагмента программы

```
#include <stdio.h>
int f(int a, unsigned short int x){
     return (a>>x)&1;}
int main () { int N=5;
printf ("The result is %d\n",f(N,0));
printf ("The result is %d\n",f(N,1));
printf ("The result is %d\n",f(N,2));
return 0;
}
```

Решение

Этот фрагмент программы выводит на экран значения бит, сначала нулевого, потом первого и второго.

```
The result is 1
The result is 0
The result is 1
```

Задание 10.2. Составить программу нахождения следующих величин: дано целое число a и натуральное число n, пользуясь бинарными операциями, определить a^n.

Решение

1. Постановка задачи

Дано целое число a и натуральное число n, пользуясь бинарными операциями, определить a^n.

Бинарное (двоичное) возведение в степень – это прием, который позволяет возводить любое число в n-ю степень за O(log n) умножений (вместо n умножений при обычном подходе).

2. Алгоритм решения задачи

Алгоритм решения задачи можно представить в виде такой последовательности действий:

Действие 1. Ввести значение чисел a и n.

Действие 2. Повторять пока n не будет 0:

Действие 2.1. Если n нечетное, то перейти к действию 2.2, иначе к действию 2.3;

Действие 2.2. Результату присвоить результат, умноженный на число a; степень n уменьшить на 1;

Действие 2.3. Числу a присвоить число a, умноженное на число a; степень n поделить на 2;

Действие 3. Вывести результат на экран.

3. Текст программы

```
#include <stdio.h>
int binpow (int a, int n) {
 int res = 1;
 while (n)
     if (n & 1) {
         res *= a;
         --n;
     }
     else {
```

```
            a *= a;
            n >>= 1;
        }
     return res;
    }
    int main () {
    int a;
    unsigned int n;
    printf ("Please enter an integer: ");
    scanf ("%d", &a);
    printf ("Please enter a power: ");
    scanf ("%u", &n);
    printf ("The result is :
    %d",binpow(a,n));
    return 0;
    }
```

4. Результат работы программы

```
Please enter an integer: 6
Please enter a power: 3
The result is : 216
```

Контрольные вопросы

1. Что такое битовые операции?
2. Как работает операция инверсия бит?
3. Как работает операция побитовое "И"?
4. Как работает операция побитовое "ИЛИ"?
5. Как работает операция побитовое исключающее "ИЛИ"?
6. Как работает операция сдвиг влево?
7. Как работает операция сдвиг вправо?
8. Что такое битовое поле?

СПИСОК РЕКОМЕНДОВАННОЙ ЛИТЕРАТУРЫ

1. **Буч, Г.** Объектно-ориентированное проектирование с примерами применения [Текст] : пер. с англ. / Г. Буч. – М. : Конкорд, 1992. – 519 с.

2. **Дьюхарт, С.** Программирование на Си++ [Текст] / С. Дьюхарт, К. Старк. – К. : НИПФ "ДиаСофт", 1993. – 272 с.

3. **Павловская, Т. А.** С/С++. Программирование на языке высокого уровня [Текст] / Т. А. Павловская. – СПб. : Питер, 2002. – 464 с.

4. **Прата, С.** Язык программирования С++. Лекции и упражнения [Текст] : 6-е изд., обновл. и расшир. / С. Прата. – М. : Диалектика. – Вильямс, 2016. – 1248 с.

5. **Страуструп, Б.** Программирование: принципы и практика с использованием С++ [Текст] : 2-е изд.; пер. с англ. / Б. Страуструп. – М. : ООО "И. Д. Вильямс", 2016. – 1328 с.

6. **Страуструп, Б.** Язык программирования С++ [Текст] / Б. Страуструп. – М. : Бином, 2004. – 369 с.

7. **Страуструп, Б.** Язык программирования С++. Специальное издание [Текст] / Б. Страуструп. – М. : Бином, 2011. – 1136 с.

8. **Шилдт, Г.** Самоучитель С++ [Текст] : 3-е изд. : пер. с англ. / Г. Шилдт. – СПб. : BHV – Санкт-Петербург, 1998. – 688 с.

9. **Шилдт, Г.** С++: базовый курс [Текст] : 3-е изд. пер. с англ. / Г. Шилдт. – М. : Издательский дом "Вильямс", 2010. – 624 с.

Приложение А

Функции для работы с файлами с использованием структуры FILE

Основные функции для работы с файлами с использованием структуры FILE приведены в таблице А.1.

Таблица А.1 – Основные функции для работы с файлами с использованием структуры FILE

Функция	Описание
`FILE *fopen (const char *fname, const char *mode)`	Открывает файл, имя которого указано аргументом `fname` и возвращает связанный с ним указатель. Тип операций, разрешенных над файлом, определяется аргументом `mode`. Разрешенные для `mode` значения приведены в приложении Б
`size_t fread (void *buffer, size_t size, size_t count, FILE *stream)`	Возвращает число прочитанных элементов. Данное значение может быть меньше, чем `count`, если был достигнут конец файла или произошла ошибка
`size_t fwrite (const void *buffer, size_t size, size_t count,FILE *stream)`	Записывает массив данных в файл. Возвращает число записанных элементов `size`. Данное значение равно `count`, если не возникла ошибка
`int fputs (const char * string, FILE *stream)`	Записывает содержимое строки, на которую указывает `string`, в заданный поток. Нуль в конце строки не записывается. В случае успеха возвращает последний записанный символ, а в случае неудачи `EOF`
`char *fgets (char *string, int n, FILE *stream)`	Считывает до `n–1` символов из файла `stream` и помещает их в массив символов, на который указывает `string`. Символы считываются до тех пор, пока не встретится символ «новая строка», `EOF` или до достижения указанного предела. По окончании считывания в массив `string` сразу после последнего считанного символа помещается нулевой символ. Символ «новая строка» при считывании будет сохранен и станет частью массива `string`

Продолж. табл. А.1

Функция	Описание
`int fclose (FILE *stream)`	Используется для закрытия потока, ранее открытого с помощью `fopen()`. Сохраняет в файл данные, находящиеся в дисковом буфере, и выполняет операцию системного уровня по закрытию файла. В случае успешного выполнения возвращает `0`, иначе – `EOF`
`int fseek (FILE *stream, long offset, int origin)`	Устанавливает указатель положения в файле, связанном со `stream`, в соответствии со значениями `offset` и `origin`. Аргумент `offset` – выраженный в байтах сдвиг от позиции, определяемой `origin`, до новой позиции. Аргумент `origin` может принимать значения: 0 – начало файла, 1 – текущую позицию, 2 – конец файла. В случае успеха возвращает 0, ненулевое значение означает неудачу
`long ftell(FILE *stream)`	Возвращает текущее значение указателя положения в файле для указанного потока. Это значение представляет собой количество байт, на которое указатель отстоит от начала файла. Возвращает `1L` в случае ошибки. Если в данном потоке невозможен поиск по произвольному адресу (в случае, например, консоли), возвращаемое значение не определяется
`int fflush(FILE *stream)`	Если `stream` связан с файлом, открытым для записи, приводит к физической записи содержимого буфера в файл. Если `stream` связан с файлом, открытым для чтения, то очищается входной буфер. В обоих случаях файл остается открытым. Возврат `0` означает успех, а возврат ненулевой величины указывает на наличие ошибки по записи
`int fileno (FILE *stream)`	Возвращает дескриптор файла указанного потока

Приложение Б

Допустимые значения аргумента `mode` функции `fopen`

В качестве типов доступа аргумента `mode` для использования в функции `fopen` могут быть указаны:

Значение	Описание
`r`	Файл открывается только для чтения. Если файл не существует, то функция генерирует ошибку (возвращает 0)
`w`	Файл открывается только для записи. Если файл не существует, то он будет создан, если существует – исходное содержимое будет перезаписано
`a`	Файл открывается для добавления в конец (дозаписи). Если файл не существует, то он будет создан
`r+`	Файл открывается для чтения и записи (файл должен существовать)
`w+`	Файл открывается для записи и чтения. Если файл существует – исходное содержимое будет перезаписано
`a+`	Файл открывается для добавления в конец (дозаписи) и чтения. Если файл не существует, то он будет создан

Все вышеописанные типы доступа предназначены для текстового режима открытия файла. Для двоичного режима открытия файла после режима достаточно добавить букву `b`. Например, `rb`.

Приложение В

Команды форматирования ввода/вывода для функций `printf()` и `fprintf()` языка С

Таблица В.1 – Преобразования, используемые в функции `printf()` и `fprintf()`

Символ преобразования	Описание
`%`	Выводится символ процента
`c`	Выводится символ, заданный соответствующим аргументом
`d`	Выводится десятичное число со знаком
`e`	Значение типа `double` будет выведено в экспоненциальной форме
`E`	Аналогично предыдущему, но в результат будет помещена прописная (заглавная) буква `E`
`f`	Значение типа `double` будет выведено в десятичном формате
`g`	Значение типа `double` будет выведено либо в экспоненциальной форме, либо в десятичном формате. Автоматически выбирается либо преобразование `f`, либо `e`, которое короче
`G`	Аналогично предыдущему, но в случае представления числа в экспоненциальной форме используется прописная (заглавная) буква `E` вместо строчной (маленькой) буквы `e`
`i`	То же, что и `d` (выводится десятичное число со знаком)
`n`	Значение аргумента интерпретируется как указатель на переменную типа `int`, в которой функция запоминает число символов, записанных до этого на устройство вывода. Это преобразование не отправляет на устройство вывода никаких символов
`o`	Выводится беззнаковое восьмиричное число
`p`	Выводится значение указателя. В моделях памяти tiny (крошечной), small (малой) и medium (средней) указатели форматируются как шестнадцатиричные значения смещения. В моделях памяти compact (компактной), large (большой) и huge (огромной) указатели форматируются как шестнадцатиричные значения сегментного компонента адреса и компонента смещения, разделенные двоеточием

Продолж. табл. В.1

Символ преобразования	Описание
`s`	Выводится содержимое завершающейся нулем строки. Задавая значение точности, можно ограничить вывод требуемым максимальным числом символов
`u`	Выводится беззнаковое десятичное число
`x`	Выводится беззнаковое шестнадцатиричное число, для представления которого используются цифры от 0 до 9 и строчные буквы `a, b, c, d, e, f`
`X`	Аналогично предыдущему, но используются прописные буквы `A, B, C, D, E, F`

Ширина – задает минимальную ширину поля вывода (в символьных позициях). Любое свободное пространство обычно заполняется пробелами. Но если значение ширины начинается с цифры 0, эти добавляемые пробелы заменяются нулями. В качестве ширины можно также указать символ * ("звездочка"), что приводит к использованию значения следующего аргумента типа `int` в качестве ширины поля.

В этом случае задается два значения: целое значение, задающее минимальную используемую ширину, и значение, выводимое в этом поле.

Значение, задающее ширину, должно располагаться перед выводимым значением. Если указана нулевая ширина, то вывод производится в поле переменной ширины и дополняется ведущими нулями. Можно задавать меньшую ширину, чем требуется, для представления значения.

Ширина поля вывода увеличивается по мере необходимости, чтобы обеспечить вывод значения без отсечения.

Точность – если в этом месте форматной строки присутствует точка, то следующее за ней значение представляет точность, используемую для представления форматируемого результата. Интерпретация точности зависит от типа форматируемого элемента. За точкой должно следовать целое зна-

чение. Это значение по умолчанию равно 0, что означает отсутствие компонента точности.

Для символов преобразования `d, i, o, u, x` и `X` по умолчанию это значение равно 1. Для символов `e, E` и `f` по умолчанию это значение равно 6. Для `g` и `G` – переменному числу значащих цифр. Для символов `s` и `c` – полному числу символов. Для символов преобразования `g` и `G` точность представляет максимальное число значащих цифр сформатированного результата. Для преобразований `e`, `E` и `f` точность равна числу используемых десятичных позиций, а последняя цифра округляется.

В случае использования `s` точность указывает максимальное число используемых символов строки.

Точность никак не влияет на результат с использованием символа преобразования `c` – всегда выводится один символ.

Для преобразований `d, i, o, u, x` и `X` выводится столько цифр, сколько указано точностью, дополненных, при необходимости, слева цифрами 0.

Один из нескольких модификаторов позволяет указать характеристики, связанные с размером выводимого значения. Требуемый тип данных выбирается соответствующим символом преобразования:

`F` – дальний (`far-`) указатель;

`N` – ближний (`near-`) указатель;

`h` – значение типа `short int`;

`l` – значение типа `long`;

`L` – значение типа `long double`.

Преобразование – задает тип соответствующего аргумента. Программист сам несет ответственность за правильность типа этого аргумента. Например, если используется символ преобразования `g`, то аргумент должен быть значением с плавающей точкой. Все допустимые символы преобразования

приведены в приложении Б, причем в каждой отдельной команде можно использовать только один из указанных символов.

Флажки задают правила выравнивания, знаки "+" и "–", десятичную точку, хвостовые нули и префиксы для восьмеричных и шестнадцатиричных значений. Флажки не являются обязательными. Если они присутствуют, то могут состоять из одного или нескольких символов (табл. В.2).

Таблица В.2 – Флаги, используемые в функциях `printf()` и `fprintf()`

Флаг	Описание
`-`	Выводимый текст выравнивается по левому краю. Все оставшееся справа пустое пространство заполняется пробелами. По умолчанию текст выравнивается по правому краю
`+`	Числовые значения предваряются знаком плюса или минуса. Обычно только отрицательные значения предваряются знаком минуса
`пробел`	Перед положительными числовыми значениями выводится пробел, а перед отрицательными – знак минуса. Этот режим действует по умолчанию, потому не употребляйте этот флаг вместе с флагом +. Не заключайте символ пробела в апострофы
`#`	В случае использования символа преобразования `x` или `X` ненулевые аргументы предваряются префиксом `0x` или `0X` соответственно. В случае символа преобразования `o` результат предваряется цифрой `0`. Если используется преобразование `e, E` или `f`, в выводимом тексте, изображающем число, будет присутствовать десятичная точка (обычно она выводится только для дробных значений). Если используется преобразование `g` или `G`, в выводимом тексте, отображающем число, будет присутствовать десятичная точка и хвостовые нули не подавляются (как это имеет место по умолчанию)

Приложение Г

Функции для работы с файлами с использованием потоков языка С++

Основные функции для работы с файлами с использованием потоков языка С++ приведены в таблице Г.1.

Таблица Г.1 – Основные функции для работы с файлами с использованием потоков

Функция	Описание
`void open (const char* filename, ios_base::openmode mode = ios_base::in | ios_base::out);`	Открывает файл, указанный в аргументе `filename`, связывая его с объектом потока. Аргумент `mode` определяет режим открытия. Разрешенные для `mode` значения приведены в приложении Д. Если поток уже связан с файлом (т. е. он уже открыт), вызов этой функции завершается с ошибкой
`int istream::get();`	Читает единственный символ из ассоциированного потока и помещает его значение в `ch`. Возвращает ссылку на поток
`ostream& ostream::put(char ch);`	Записывает `ch` в поток и возвращает ссылку на этот поток
`istream& istream::getline( char* buffer, streamsize num, char delim );`	Считывает неформатированные данные из потока в строку. Останавливается, как только найден символ, равный разделителю или исчерпан поток. Первая версия использует в качестве разделителя `delim`, вторая – `'\n'`. Символ-разделитель удаляется из потока и не помещается в строку
`istream& istream::read( char* buffer, streamsize num );`	Используется с потоками ввода. Считывает `num` байт из ассоциированного потока и посылает их в буфер `buffer`. Если достигнут конец файла `EOF`, останавливается, оставляя текущее количество байт в буфере

Продолж. табл. Г.1

Функция	Описание
`ostream& ostream::write( const char* buffer, streamsize num );`	Используется с потоками вывода. Записывает `num` байт в ассоциированный поток из буфера `buffer`
`void close();`	Используется для закрытия файла. Не имеет ни параметров, ни возвращаемого значения
`bool istream::eof();`	При вызове с действительным дескриптором файла возвращает 1, если был достигнут конец файла; в противном случае она возвращает 0. В случае ошибки она возвращает -1
`bool stream::fail();`	Возвращает истину, если обнаружена ошибка в текущем потоке, иначе возвращает ложь. Функция может использоваться для проверки успешности предыдущей операции
`int gcount();`	Возвращает число символов, прочитанных последним оператором двоичного ввода.

Приложение Д

Допустимые значения аргумента mode метода open с использованием потоков языка С++

Открыть файл в программе при использовании потоков можно с использованием либо конструкторов, либо метода open, имеющего такие же параметры, как и в соответствующем конструкторе.

Значением аргумента `mode` при использовании метода `open` является режим открытия файла. Вместо значения по умолчанию можно указать одно из следующих значений, определенных в классе `ios`: (табл. Д.1).

Таблица Д.1 – Основные режимы для работы с файлами с использованием потоков

Флаг	Описание
`ios::in`	Открыть файл для чтения (выбирается по умолчанию для `ifstream`)
`ios::out`	Открыть файл для записи (выбирается по умолчанию для `ofstream`)
`ios::ate`	Установить файловый указатель на конец файла
`ios::app`	Присоединять данные; запись в конец файла
`ios::trunc`	Уничтожить содержимое, если файл существует (выбирается по умолчанию, если флаг `out` указан, а флаги `ate` и `app` – нет)
`ios::binary`	Открыть файл в двоичном режиме
`ios::nocreate`	Если файл не существует, выдать ошибку, новый файл не открывать
`ios::noreplace`	Если файл существует, выдать ошибку, существующий файл не открывать

Приложение Е

Сравнительные характеристики текстового и бинарного файлов

Сравнительные характеристики типов файлов приведены в таблице Е.1.

Таблица Е.1 – Сравнительные характеристики текстового и бинарного файлов

Текстовый файл	Бинарный файл
Число, записанное в файл, запишется как текст, т.е. размер занятого пространства будет равен количеству цифр в этом числе	Число, записанное в файл, запишется в бинарном формате, т.е. размер занятого пространства будет равен размеру типа данных в этом числе
Прочитать файл можно с помощью любого текстового редактора	Прочитать файл можно с помощью специальной программы
При потере нескольких байт информацию можно восстановить по смыслу	Потеря нескольких байт может быть необратима
Считывание информации программным путем затруднено, т. к. файл представляет собой единый текстовый блок	Считывание информации программным путем облегчено, т. к. файл представляет собой набор блоков информации с конкретными типами данных
Используется как файл для чтения, редактирования, вывода на печать	Используется как файл для чтения, хранения, записи информации программным путем

Приложение Ж

Директивы препроцессора

Основные директивы препроцессора приведены в таблице Ж.1.

Таблица Ж.1 – Директивы препроцессора

Директива	Описание
`#include <имя_файла>`	Включение в исходный текст программы содержимого файла `имя_файла`. Препроцессор ищет файл в стандартных системных каталогах
`#include "имя_файла"`	Включение в исходный текст программы содержимого файла `имя_файла`. Вначале препроцессор просматривает текущий каталог пользователя, а затем обращается к стандартным системным каталогам
`#define`	Используется для определения символических констант и программных конструкций целиком
`#undef`	Отменяет определение, заданное с помощью `#define`
`#if`	Директива условной компиляции. Проверяется значение константного целочисленного выражения. Если оно отлично от нуля, то считается, что проверяемое условие истинно
`#ifdef`	Директива условной компиляции. Проверяется, определен ли с помощью команды `#define` к текущему моменту идентификатор, помещенный после `#ifdef`
`#ifndef`	Директива условной компиляции. Проверяется обратное условие – истинным считается неопределенность идентификатора, помещенного после `#ifndef`, т. е. тот случай, когда идентификатор не был использован в команде `#define` или его определение было отменено командой `#undef`
`#else`	Директива условной компиляции. При ее наличии директивы на компиляцию передается текст, следующий за директивой `#else`

Продолж. табл. Ж.1

Директива	Описание
`#elif`	Директива условной компиляции. Используется для организации мультиветвлений. Является сокращением конструкции `#else#if`
`#endif`	Директива условной компиляции. Определяет окончание блока команд условной компиляции
`#line<константа>`	Указывает компилятору, что следующая ниже строка текста имеет номер, определяемый целой десятичной константой. Команда может определять не только номер строки, но и имя файла
`#error<последовательность_лексем>`	Приводит к выдаче диагностического сообщения в виде последовательности лексем
`#pragma<последовательность_лексем>`	Определяет действия, зависящие от конкретной реализации компилятора, и позволяет выдавать компилятору различные инструкции
`#`	Превращает аргумент, которому предшествует, в строку, заключенную в кавычки
`##`	Используется для конкатенации (объединения) двух лексем

Приложение З

Сравнение рекурсивных и итеративных алгоритмов

Сравнительные характеристики рекурсивных и итеративных алгоритмов представлены в таблице З.1.

Таблица З.1 – Сравнительные характеристики рекурсивных и итеративных алгоритмов

Рекурсия	Итерация
Решает задачу методом "от сложного к простому"	Решает задачу методом "от простого к сложному"
Алгоритм – описание сложного объекта через более простой (простые) объект того же типа	Алгоритм – описание процесса построения сложного объекта из более простых объектов
Запись алгоритма, как правило, компактна, интуитивно понятна, красива	Запись алгоритма не всегда компактна, интуитивно понятна
Исполнение алгоритма связано с большими затратами памяти, и, как правило, более медленное, чем итеративное	Исполнение итеративного алгоритма связано с меньшими затратами памятии, как правило, более быстрое, чем рекурсивное

Рекурсивная версия алгоритма выполняется медленнее, чем ее итеративный эквивалент, из-за дополнительных затрат ресурсов, что связано с многократным вызовом. Проиллюстрируем время выполнения кода на примере решения задачи вычисления факториала. В примере 1 рассмотрим рекурсивный вариант алгоритма.

Пример 1

```
#include <stdio.h>
#include <time.h>
using namespace std;
int fact(int n) {
```

```
if (n==0) return 1;
   else
      return n*fact(n-1);
}

int main(){int k;
unsigned int start_time =  clock();
for (int i=0;i<10000000;i++)
    k=fact(15);
 unsigned int end_time =  clock();

printf("fact=%d   time=%16.12f
    \n",k,double(end_time-start_time) /
    10000000);
    return 0;
}
```

Результат работы программы:

```
fact=2004310016   time=   0.000109300000
```

В примере 2 приведен итеративный вариант алгоритма задачи вычисления факториала.

Пример 2

```
#include <stdio.h>
#include<time.h>
using namespace std;
int fact(int n) {
    int f=1;
for(int i=1;i<=n;i++)
 f=f*i;
 return f;
}
```

```
int main(){int k;
unsigned int start_time =  clock();
for (int i=0;i<10000000;i++)
k=fact(15);
 unsigned int end_time =  clock();

printf("fact=%d   time=%16.12f
   \n",k,double(end_time-start_time)/
   10000000);
     return 0;
}
```

Результат работы программы:

```
fact=2004310016   time=   0.000058700000
```

Как видно из приведенных результатов работы программ, решение задачи вычисления факториала итеративным способом дает незначительное сокращение времени выполнения программы.

СОДЕРЖАНИЕ