Задачи к контрольной работе «Программирование»

Задача №1 Списки

Написать одну программу на языке C++, вторую на Паскале. Разбиение программ на функции является обязательным. Для создания и печати списков использовать функции, представленные в методических указаниях к работе. Задания реализовать в виде отдельных модулей.

Задание:

- Определить количество элементов списка, значения которых совпадают со значениями в первом и последнем элементах списка. (Программа 1 – на паскале)
- Создать список L3 из элементов, входящих и в список L1 и в список L2, вернуть длину нового списка. (Программа 2 – на C++)

Функции для работы к задаче №1.

```
struct List
{
int value;
List *next;
List (int val = 0,List *p = NULL) //конструктор
{
value = val;
next = p;
}
};
//включение элемента в начало
List *Ins_first(int n,List *head)
{
List *q = new List(n, head);
return q;
}
//включение элемента в конец
List *Add_last(int n, List *head)
{
List *q = new List(n), *p = head;
// если список пуст
if(head == NULL) return q;
// «идем в конец списка»
while(p -> next != NULL)
p = p -> next;
p -> next = q;
return head;
}
//включение по значению
List*Ins_Sort(int n, List *head)
{
List *q = new List(n), *p = head;
if(head == NULL) return q;
// включение в начало
if(n < head -> value)
{
q -> next = head;
```

```
return q;
}
// включение в «середину»
while(p -> next != NULL)
if (n < p -> next -> value)
{
q -> next = p -> next;
p -> next = q;
break;
}
else
p = p -> next;
// включаем последним
p -> next = q;
return head;
}
// включение по номеру
List* Ins_Number(int Num, int n, List *head)
{
List *p = head, *q = new List(n);
int i = 1;
if(head == NULL) return q;
// включаем по номеру «1»
if(Num == 1)
{
q -> next = head;
return q;
}
// ищем позицию для включения
while(p -> next != NULL)
if(Num == i+1)
{
q -> next = p -> next;
p -> next = q;
return head;
}
else
{
i++;
p = p -> next;
}
// включить последним
if(Num == i+1)
p -> next = q;
else
puts("NO THIS NUMBER");
return head;
}
// создание списка
List* Creat_list()
{
char str[N];
List *head = NULL;
puts("Creatlist.Enter numbers:");
```

```
for(;;)
{
gets(str);
if(!str[0]) break;
head=Ins_first(atoi(str), head);
}
return head;
}
// удаление из начала
List* Del_first (int &n, List *head)
{
List *p = head;
if(p == NULL)puts("LIST EMPTY!");
else
{
n = p -> value;
head = head -> next;
delete p;
}
return head;
}
// удаление из конца
List* Del_last(int &n,List *head)
{
List *p = head;
if (p == NULL)
{
puts ("LIST EMPTY!");
return NULL;
}
// если один элемент в списке
if (head -> next == NULL)
{
n = head -> value;
delete head;
return NULL;
}
// переходим на предпоследний элемент
while (p -> next -> next != NULL)
p = p -> next;
// запоминаем удаляемое значение
n = p -> next -> value;
delete p -> next;
p -> next = NULL;
return head;
}
// удаление по значению
List* Del_value(int n,List *head)
{
List *p = head, *t;
if(head == NULL)
{
puts("LIST EMPTY!");
return NULL;
```

```
}
// удаляем первый элемент
if(head -> value == n)
{
t = head;
head = head -> next;
delete t;
return head;
}
// ищем в списке значение
while(p -> next != NULL)
if (p -> next -> value == n)
{
t = p -> next;
p -> next = p ->next->next;
delete t;
return head;
}
else p = p -> next;
puts("NO VALUE!");
return head;
}
// удаление по номеру
List* Del_Number (int Num, List*head)
{
List *p = head, *t;
int i = 1;
if(head == NULL)
{
puts("LIST EMPTY!");
return NULL;
}
if(Num == 1)
{
t = head;
head = head -> next;
delete t;
return head;
}
while(p -> next != NULL)
if(Num == i+1)
{
t = p -> next;
p->next = p-> next-> next;
delete t;
return head;
}
else
{
i++;
p = p -> next;
}
puts("NO THIS NUMBER");
return head;
```

```
}
//печать списка
void Print_list(List *head)
{
List *p = head;
puts("\n PRINT LIST");
if(p==NULL) puts("List empty!");
else
while(p != NULL)
{
printf("%d ", p -> value);
p = p -> next;
}
_getch();
}
int main()
{
int i, k, n, nu;
char c, *ss[] = {"\n 0-Print list"," 1-Insert SORT"," 2-Insert first"," 3-Add last"," 4-Insert Number"," 5-Delete
first"," 6-Delete last"," 7-Delete value"," 8-Delete number"," 9-EXIT"};
List *head = NULL;
k = sizeof(ss)/sizeof(ss[0]);
for(;;)
{
for(i = 0; i < k; i++) puts(ss[i]);
c=_getch();
switch(c)
{
case '0': Print_list(head); break;
case '1': printf("n = "); scanf("%d", &num);
head = Ins_Sort (num, head); break;
case '2': printf("n = ");
scanf("%d", &num);
head = Ins_first(num, head); break;
case '3': printf("n = "); scanf("%d", &num);
head = Add_last (num, head); break;
case '4': printf("Number= "); scanf("%d", &num);
printf("n= "); scanf("%d", &n);
head = Ins_Number (num, n,head);break;
case '5': head = Del_first(num, head); break;
case '6': head = Del_last (num, head); break;
case '7': printf("value = "); scanf("%d", &num);
head = Del_value (num, head); break;
case '8': printf("Number = "); scanf("%d", &num);
head = Del_Number(num, head); break;
case '9': return;
}
}
}
```

ПАСКАЛЬ:

```
{Вводим новый тип данных}
Type BT = LongInt;
Type U = ^Zveno;
Zveno = Record Inf : BT; Next: U End;
```

```
{Проверка на пустой элемент - нулевой адрес}
Function Pust(First : U) : Boolean;
Begin
Pust := First = Nil
End;
{Процедура добавления элемента в начало списка; в x содержится добавляемая информация}
Procedure V_Nachalo(Var First : U; X : BT);
Var Vsp : U;
Begin
New(Vsp);
Vsp^.Inf := X;
{Тот элемент, что был заглавным, становится вторым по счѐту}
Vsp^.Next := First;
First := Vsp; {Новый элемент становится заглавным}
End;
{Процедура удаления элемента из начала списка; в x содержится информация из удалѐнного звена}
Procedure Iz_Nachala(Var First : U; Var X : BT);
Var Vsp : U;
Begin
Vsp := First; {Забираем ссылку на текущее заглавный элемент}
First := First^.Next; {Второй элемент становится заглавным}
X := Vsp^.Inf; {Забираем информацию из удаляемого элемента}
Dispose(Vsp); {Уничтожаем элемент}
End;
{Процедура добавления элемента в конец списка}
Procedure V_Konez(Var First : U; X : BT);
Var Vsp, Tmp : U;
Begin
New(Vsp); {Создаем пустый элемент}
Vsp^.Inf := X; {Заносим информацию}
Vsp^.Next:= Nil; {Обнуляем ссылку}
{Если список был пустой - включаем первый элемент}
If First = Nil Then First := Vsp
Else
Begin
Tmp := First; {Проходим до конца списка}
While Not Pust(Tmp^.Next) Do Tmp := Tmp^.Next;
Tmp^.Next := Vsp; {Вставляем элемент последним}
End;
End;
{Процедура удаления элемента из конца списка}
Procedure Iz_Konza(Var First : U; Var X : BT);
Var Vsp, Tmp : U;
Begin
Tmp := First;
Vsp := First;
{Если список состоит из одного элемента}
If First^.Next = Nil Then First := Nil
Else
Begin
{Проходим до предпоследнего элемента}
While Not Pust(Tmp^.Next^.Next) Do Tmp := Tmp^.Next;
Vsp := Tmp^.Next; {Запоминаем адрес удаляемого элемента}
Tmp^.Next := Nil; {Определяем новый конец списка}
```

```
End;
X := Vsp^.Inf; {Забираем информацию из удаляемого звена}
Dispose(Vsp); {Уничтожаем элемент}
End;
{Процедура печати списка}
Procedure Print(First : U);
Begin
If First = Nil Then writeln('Пустой список!')
Else
Begin
While First <> Nil Do
Begin
Write(First^.Inf : 6);
First := First^.Next
End;
End;
WriteLn
End;
{основная программа - вызовы функций}
var chislo : BT;
Var nachalo : U;
Begin
nachalo := Nil;
writeln('Работа со списком.');
writeln('вводи числа(0 - конец ввода)');
read(chislo);
While chislo <> 0 Do
Begin
V_nachalo(nachalo, chislo);
read(chislo);
End;
writeln('Печать списка:');
Print(nachalo);
While Not Pust(nachalo) Do
Begin
Iz_Konza(nachalo, chislo);
writeln('Печать после удаления:');
Print(nachalo);
End;
End.
```

Задача №2 Бинарные деревья

Написать программу на языках: C++ и Паскаль. Разбиение программ на функции является обязательным. Для создания и печати деревьев использовать функции, представленные в методических указаниях к работе. Задание реализовать в виде отдельного модуля.

Задание:

Поменять местами максимальный и минимальный элементы дерева T, все элементы дерева различны.

Функции для работы к задаче №2.

```
struct btree
{
int value;
struct btree *left, *right;
};
// Включение вершины в дерево
void Ins_Btree (int val,
btree **q)
{
if(*q == NULL)
{//Нашли место для добавления
*q = new btree;
(*q) -> left = NULL;
(*q) -> right = NULL;
(*q) -> value = val;
return;
}
if((*q) -> value > val)
// Добавляем в левое поддерево
Ins_Btree (val, &(*q) -> left);
else
// Добавляем в правое поддерево
Ins_Btree (val, &(*q) ->right);
}
//Вывод содержимого дерева
void Print_Btree (btree *p)
{
if (p == NULL) return;
Print_Btree(p -> left);
printf("%d ", p-> value);
Print_Btree(p -> right);
}
void main()
{
int d;
btree *root = NULL;
puts("Дерево. Ввод чисел:");
while(1)
{
scanf("%d", &d);
if(d == 0)break;
Ins_Btree(d, &root);
}
```

```
Print_Btree(root);
}
```

ПАСКАЛЬ:

```
{Вводим новый тип данных}
type ref = ^node;
node = record
key, count: integer;
left, right: ref;
end;
{Печать содержимого бинарного дерева}
procedure Print_Btree (w: ref; l: integer);
var i : integer;
begin
if (w <> Nil) then
with w^ do
begin
Print_Btree(right,l+1);
for i:= 1 to l*4 do
write(' ');
// writeln('значение = ',key,' счетчик = ', count);
writeln(key);
Print_Btree(left,l+1);
end;
end;
{Построение дерева повторений}
procedure Include (x: integer; var p:ref);
begin
if p = Nil then
begin {Добавляем вершину}
new (p);
with p^ do
begin
key := x;
count := 1;
left := Nil;
right := Nil;
end;
end
else
begin
if x = p^.key then
{Если число есть в дереве, то увеличиваем счѐтчик}
p^.count := p^.count + 1
else
if x > p^.key then
{Добавляем в правое поддерево}
Include(x, p^.right)
else
{Добавляем в левое поддерево}
Include(x, p^.left);
end;
end;
{Построение идеально-сбалансированного дерева с n вершинами}
function tree (n : integer): ref;
var
```

```
newnode : ref;
x, l, r : integer;
begin
if n = 0 then tree := nil else
begin
l := n div 2; r := n - l - 1;
read(x);
{Добавляем вершину}
new (newnode);
with newnode^ do
begin
key := x;
left := tree(l);
right := tree(r);
end;
tree := newnode;
end;
end;
{Основная программа}
var
root: ref;
k: integer;
begin
root := Nil;
writeln('Дерево повторений.');
writeln('Вводи числа (0 - конец ввода)');
read(k);
while k <> 0 Do
begin
Include (k,root);
read(k);
end;
writeln('Печать дерева');
Print_Btree (root,2);
writeln('Дерево идеально - сбалансированное.');
writeln('Вводи числа (первое - количество чисел)');
read(k);
root := tree(k);
writeln('Печать дерева');
Print_Btree (root,2);
end.
```

Задача №3 текстовые файлы

Написать одну программу на языке C++, вторую на Паскале. Разбиение программ на функции является обязательным.

Задание:

- Определить, сколько символов букв содержит файл. (Программа 1)
- Скопировать из файла F1 в файл F2 все символы кроме символов-«цифр» (Программа 2)

Функции для работы к задаче №3.

Библиотечные функции для работы с файлом:
FILE *fopen (char *Name, char *Mode); - открывает файл с именем Name в заданном режиме Mode, например,
“w” – создает файл для записи,
“r” – открывает файл для чтения,
«w+» – создает файл для записи и чтения,
“r+” – открывает файл для чтения и записи.
Добавление литеры b в модификатор режима открытия файла явно указывает на то, что файл открывается как бинарный, например «wb+» или “rb+”. Функция возвращает указатель на файловый поток.
int **fclose** (FILE *Stream); – закрывает файловый поток, открытый функцией fopen().
int **feof** (FILE *stream); – обнаруживает конец файла в потоке.
int **fseek** (FILE *Stream, long offset, int whence); – позиционирует указатель в файле, на который ссылается поток Stream, на число байт, равное значению offset относительно начала файла (whence = 0), конца файла (whence = 2) или текущей позиции в файле (whence = 1). Возвращает 0 при удачном смещении и 1 при ошибке.
int **rewind** (FILE *stream); – устанавливает указатель файла на начало потока.
long int **ftell** (FILE *stream); – возвращает положение указателя текущей позиции файла, связанного с потоком stream.
int **fgetc** (FILE *stream); – возвращает следующий символ из потока.
int **fputc** (int c, FILE *stream); – выводит символ в поток.
char ***fgets** (char *s, int n, FILE *stream); – получает строку символов из потока.
int **fputs** (char *string, FILE *stream); – выводит строку символов в поток.
size_t **fread** (void *ptr, size_t size, size_t n, FILE *stream); – считывает n блоков размером size в память по адресу ptr из потока stream. Возвращает количество прочитанных блоков.
size_t **fwrite** (const void *ptr, size_t size, size_t n, FILE *stream); – записывает n блоков размером size из памяти по адресу ptr в поток stream. Возвращает количество записанных блоков.
int **fflush** (FILE *stream); – очищает поток.
int **remove** (const char *filename); – удаляет файл.
int **rename** (const char *oldname, const char *newname); – изменяет имя файла (можно использовать для пересылки).

Определить размер файла (в байтах) можно следующей последовательностью операторов:
FILE * file_pnt = **fopen** (filename, “rt”);
fseek (file_pnt, 0L, SEEK_END);
printf (“ Размер файла %d байтов”, **ftell** (file_pnt));

Примеры программных реализаций

Пример 1. Функции для работы с текстовым файлом. Чтение информации из файла построчно.

```
// Открытие файла в заданном режиме
FILE *Open_file (char *mode)
{
char f_name[L];
FILE *fp;
printf("Enter file name:");
scanf("%s",f_name);
if((fp=fopen(f_name,mode))==NULL)
{
puts("Error creat!");
getch();
exit(0);
}
return fp;
}
// Вывод строк файла на экран. Перед строкой отображается еѐ номер в
файле
void Print_File(FILE * f)
{
char str[N];
int i = 1;
rewind(f);
while(feof(f)== 0)
{
fgets(str, N, f);
printf("%d\t %s",i,str);
i++;
}
}
// Поиск максимальной длины строк текстового файла
int Find_Max_Length(FILE * f)
{
char str[N];
int l, len = 0;
rewind(f);
while(feof(f)== 0)
{
fgets(str, N, f);
l = strlen(str);
if (l > len) len = l;
}
return len;
}
// Напечатать первую из самых коротких строк файла.
int Print_First_Short(FILE * f)
{
char str[N];
int n, i = 2, l, len;
rewind(f);
len = strlen(fgets(str, N, f));
while(feof(f) == 0)
{
fgets(str, N, f);
l = strlen(str);
if (l < len)
{ len = l; n = i;}
```

```
i++;
}
rewind(f);
while(feof(f) == 0)
{
fgets(str, N, f);
l = strlen(str);
if (l == len)
{
printf("\n%d\t%s\n",l, str);
break;
}
}
return n;
}
//Подсчитать количество пустых строк файла
int Count_Empty(FILE * f)
{
char str[N];
int l, count = 0;
rewind(f);
while(feof(f) == 0)
{
fgets(str, N, f);
l = strlen(str);
if (str[0] == '\n') count++;
}
return count;
}
int main()
{
FILE *fstream = Open_file("rt");
Print_File(fstream);
printf("\n Max len = %d", Find_Max_Length(fstream));
printf("\n Min_len number = %d", Print_First_Short(fstream));
printf("\n Kol-vo pustyx = %d", Count_Empty(fstream));
_getch();
}
```

Пример 2. Написать программу – интерпретатор текста, включающего фрагменты вида: `# repeat 3 текст #end` .При просмотре файла программа выводит его текст, текст фрагментов `# repeat - #end` выводится указанное количество раз. Фрагменты текста могут быть вложенными. Рекурсивная функция `Scan ()` запоминает значение файлового указателя начала очередного фрагмента в переменной `file_ptr` и вызывается определенное количество раз.

```
#define N 40
void Scan( int counter,
FILE *f_inp, *f_out)
{
char buf[N];
int i, num;
long file_ptr = ftell(f_inp);
for( i = 0; i < counter; i ++)
{
fseek(f_inp,file_ptr,SEEK_SET);
while(fgets(buf,N,f_inp)!=NULL)
{
if (buf[0] != '#')
{ fputs( buf, f_out);
continue;
```

```
}
if(strstr( buf, "end"))break;
if(strstr(buf,"repeat")==NULL)
{
fputs( buf, f_out);
continue;
}
num = atoi(strstr(buf,"repeat")+7);
Scan (num, f_inp, f_out);
}
}
}
void main()
{
FILE *f_inp, *f_out;
char name_inp[N],name_out[N];
puts("\nEnter inp_file name:");
gets(name_inp);
if(!f_inp=fopen(name_inp,"rt")) {
puts("Error1!");
return;
}
puts("\nEnter out_file name:");
gets(name_out);
if(!f_out=fopen(name_out,"wt"))
{
puts("Error2!");
return;
}
Scan(1, f_inp, f_out);
fcloseall();
}
```

Процедуры и функции для работы с файлами
через файловые переменные в Паскале

procedure Assign(f: FileType; name: string); Связывает файловую переменную f с именем файла name

procedure AssignFile(f: FileType; name: string); Связывает файловую переменную f с именем файла name

procedure Close(f: FileType); Закрывает файл f

procedure CloseFile(f: FileType);Закрывает файл f

procedure Reset(f: Text); Открывает текстовый файл f на чтение

procedure Reset(f: file of T); Открывает типизированный файл f на чтение и запись

procedure Reset(f: file); Открывает нетипизированный файл f на чтение и запись

procedure Rewrite(f: Text); Открывает текстовый файл f на запись, обнуляя его содержимое. Если файл существовал, он обнуляется

procedure Rewrite(f: file of T); Открывает типизированный файл f на чтение и запись, обнуляя его содержимое. Если файл существовал, он обнуляется

procedure Rewrite(f: file); Открывает нетипизированный файл f на чтение и запись, обнуляя его содержимое. Если файл существовал, он обнуляется

procedure Append(f: Text); Открывает текстовый f файл на дополнение

function Eof(f: FileType): boolean; Возвращает True, если достигнут конец файла f

procedure Flush(f: FileType); Записывает содержимое буфера файла на диск

procedure Erase(f: FileType); Удаляет файл, связанный с файловой переменной f

procedure Rename(f: FileType; newname: string); Переименовывает файл, связаный с файловой переменной f, давая ему имя newname

function Eoln(f: Text): boolean; Возвращает True, если достигнут конец строки в текстовом файле f

function SeekEof(f: Text): boolean; Пропускает пробельные символы, после чего возвращает True, если достигнут конец текстового файла f
function SeekEoln(f: Text): boolean; Пропускает пробельные символы, после чего возвращает True, если достигнут конец строки в текстовом файле f
function FilePos(f: file of T): int64; Возвращает текущую позицию файлового указателя в типизированном файле f
function FilePos(f: file): int64; Возвращает текущую позицию файлового указателя в нетипизированном файле f
function FileSize(f: file of T): int64; Возвращает количество элементов в типизированном файле f
function FileSize(f: file): int64; Возвращает количество байт в нетипизированном файле f
procedure Seek(f: file of T; n: int64); Устанавливает текущую позицию файлового указателя в типизированном файле f на элемент с номером n
procedure Seek(f: file; n: int64); Устанавливает текущую позицию файлового указателя в нетипизированном файле f на байт с номером n
Примеры программных реализаций (Паскаль)
Пример 1. Функции для работы с текстовым файлом. Чтение информации из файла посимвольно.

```
// Открытие файла
procedure OpenFile(var f: file of char);
var
fileName: string;
begin
write('Введите имя файла: ');
Readln(fileName);
assign(f,fileName);
reset(f);
end;
// Вывод строк файла на экран. Перед строкой отображается еѐ номер в файле
procedure PrintFile(var f: file of char);
var
ch: char;
num,pos: integer; // num – номер строки в файле
begin
num := 1; write(num,' ');
reset(f);
while not eof(f) do
begin
Read(f,ch);
Dec(pos);
if ch = #10 then continue;
if ch = #13 then // переходим к следующей строке
begin
writeln();
Inc(num);
write(num,' ');
continue;
end;
write(ch);
end;
end;
// Поиск максимальной длины строк текстового файла
function Find_Max_Length(var f:file of char):integer;
var
ch: char;
len,rez:integer;//len-длина строки,rez-максимальная длина
begin
reset(f);
```

```
while not eof(f) do
begin
Read(f, ch);
if ch = #10 then continue;
if ch = #13 then // переходим к следующей строке
begin
if len > rez then rez := len;
len:=0;
continue;
end;
Inc(len);
end;
Find_Max_length := rez;
end;
//Подсчитать количество пустых строк файла
function Count_Empty(var f: file of char):integer;
var
ch: char;
len:integer;
count:integer;
begin
reset(f);
while not eof(f) do
begin
Read(f, ch);
if ch = #10 then continue;
if ch = #13 then // переходим к следующей строке
begin
if len = 0 then Inc(count);
len := 0;
continue;
end;
Inc(len);
end;
Count_Empty := count;
end;
// Напечатать первую из самых коротких строк файла.
function Print_First_Short(var f:file of char):integer;
var
ch: char;
len, l, i, n:integer;
begin
reset(f);
// читаем первую строку
while ch<>#13 do
begin
Read(f,ch);
if ch = #13 then break;
Inc(l);
end;
i:=2;
// читаем остальные строки и ищем минимальную
while not eof(f) do
begin
Read(f, ch);
if ch = #10 then continue;
if ch = #13 then // переходим к следующей строке
begin
```

```
if (l>len) and (len<>0) then
begin
n:=i;
l:=len;
end;
len:=0;
Inc(i);
continue;
end;
Inc(len);
end;
i:=1;
reset(f);
// выводим минимальную строку
while not eof(f) do
begin
if i = n then
begin
write(n,' строка минимальная: ');
while ch <> #13 do
begin
Read(f,ch);
write(ch);
end;
Print_First_Short:=n;
exit();
end;
Read(f, ch);
if ch = #13 then continue;
if ch = #10 then Inc(i);
end;
end;
var
f: file of char;
begin
OpenFile(f);
writeln('Вывод файла: ');
PrintFile(f);
writeln('Максимальная длина строки: ',Find_Max_Length(f));
writeln('Количество пустых строк файла: ', Count_Empty(f));
Print_First_Short(f);
close(f);
Readln();
end.
```