**Условие задачи**

Составить программу для обработки информации о некоторых объектах - студентах какого-то факультета. Каждый объект характеризуется совокупностью признаков.

Выбор типа исходных и выходных данных зависит от языка программирования, программы обучения и желания обучаемого. Возможно, рассмотреть два варианта, соответствующих объекту: строка и запись (структура). Кроме того, для каждого из этих вариантов можно рассмотреть три способа ввода (вывода): с клавиатуры монитора (на экран); из текстового файла на диске; из типизированного файла.

23. Для заданных трех груни подсчитать число студентов, которым меньше 16 лет, и вывести все данные о них.

**Решении задачи**

Необходимо разработать информационную систему по обработке информации о студентах. Система должна обладать возможность хранения, поиска и обработки информации о студентах.

**Листинг программы, написанной на языке object pascal и скомпилированной с помощью freepascal**

**program** lab\_5\_2\_1\_and\_5\_2\_2;

**uses** sysutils, strutils, crt, DateUtils;

**type**

student\_type =

**record**

group: string;

fio: string;

date: tdatetime;

sex: boolean;

pmark: byte;

mmark: byte;

imark: byte;

scholarship: real;

**end**;

node = ^node\_type;

node\_type =

**record**

value: student\_type;

next: node

**end**;

list\_type =

**record**

head: node;

tail: node;

count: integer;

**end**;

**procedure** insert(**var** list: list\_type; item: student\_type);

**var**

tmp: node;

**begin**

**new**(tmp);

tmp^.value := item;

tmp^.next := nil;

**if** list.head = nil

**then** list.head := tmp

**else** list.tail^.next := tmp;

list.tail := tmp;

**end**;

**function** student\_type\_to\_str(item: student\_type): string;

**begin**

student\_type\_to\_str := (

format('%s %s %s %s %d %d %d %f',

[item.group,

item.fio,

datetostr(item.date),

ifthen(item.sex, 'm', 'w'),

item.pmark,

item.mmark,

item.imark,

item.scholarship]))

**end**;

**function** filter\_students\_by\_age(list: list\_type; filter: integer): list\_type;

**var**

tmp\_list: list\_type;

tmp: node;

**begin**

tmp := list.head;

**while** tmp <> nil **do**

**begin**

**if** YearsBetween(date, tmp^.value.date) <= filter **then begin**

insert(tmp\_list, tmp^.value);

**end**;

tmp := tmp^.next

**end**;

filter\_students\_by\_age := tmp\_list;

**end**;

**procedure** display(list: list\_type; table\_name: string);

**var**

tmp: node;

**begin**

ClrScr();

tmp := list.head;

writeln('------------------------------------------------');

writeln(table\_name, ' students');

writeln('------------------------------------------------');

writeln('group fio date sex pmark mmark imark scholarship');

writeln('------------------------------------------------');

**if** tmp = nil **then begin**

writeln('<empty>');

**end**;

**while** tmp <> nil **do**

**begin**

writeln(student\_type\_to\_str(tmp^.value));

tmp := tmp^.next

**end**;

writeln('------------------------------------------------');

**end**;

**procedure** init\_student(

**var** student: student\_type;

input\_group: string;

input\_fio: string;

input\_date: tdatetime;

input\_sex: boolean;

input\_pmark: byte;

input\_mmark: byte;

input\_imark: byte;

input\_scholarship: real);

**begin**

student.group := input\_group;

student.fio := input\_fio;

student.date := input\_date;

student.sex := input\_sex;

student.pmark := input\_pmark;

student.mmark := input\_mmark;

student.imark := input\_imark;

student.scholarship := input\_scholarship;

**end**;

**label** point;

**var**

// linked list vars

students: list\_type;

// tui

command: string;

age\_filter: integer;

// tmp record

tmp\_student\_object: student\_type;

// tmp vars

tmp\_group\_str: string;

tmp\_fio\_str: string;

tmp\_sex\_str: string;

tmp\_day: word;

tmp\_month: word;

tmp\_year: word;

tmp\_pmark: integer;

tmp\_mmark: integer;

tmp\_imark: integer;

tmp\_scholarship: real;

**begin**

point:

write('→ ');

readln(command);

**case** command **of**

'add':

**begin**

writeln('create student');

write('group ');

readln(tmp\_group\_str);

write('fio ');

readln(tmp\_fio\_str);

write('day ');

readln(tmp\_day);

write('month ');

readln(tmp\_month);

write('year ');

readln(tmp\_year);

write('sex m/w ');

readln(tmp\_sex\_str);

write('pmark ');

readln(tmp\_pmark);

write('mmark ');

readln(tmp\_mmark);

write('imark ');

readln(tmp\_imark);

write('scholarship ');

readln(tmp\_scholarship);

writeln('readed values');

init\_student(

tmp\_student\_object,

tmp\_group\_str,

tmp\_fio\_str,

encodedate(tmp\_year, tmp\_month, tmp\_day),

ansimatchstr(tmp\_sex\_str, ['m']),

tmp\_pmark,

tmp\_mmark,

tmp\_imark,

tmp\_scholarship);

writeln('created student: ', student\_type\_to\_str(tmp\_student\_object));

insert(students, tmp\_student\_object);

display(students, 'non filtered');

**end**;

'age':

**begin**

write('age\_filter: ');

read(age\_filter);

display(filter\_students\_by\_age(students, age\_filter), 'filtered');

**end**;

'display':

**begin**

display(students, 'non filtered');

**end**;

**end**;

**goto** point;

**end**.