Министерство образования Республики Беларусь
Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет непрерывного и дистанционного обучения

Кафедра экономической информатики

Контрольная работа №2
По курсу «Основы алгоритмизации и программирования» часть 1

Выполнил — Студент
Ф.И.О. Юруш Е.В.
№ зач. кн. (нет)

Проверил — Т. М. Унучек

Минск-2017

Содержание

Введение

Выполнение данной контрольной работы заключается в доработке консольного приложения разработанного при выполнении контрольной работы №1 таким образом, чтобы оно предусматривало использование многомерных массивов, динамическое использование памяти, указателей на функции, а также разделение проекта на несколько компонентов, которые используются через директиву #include. Предназначение данного приложения - автоматизации учёта студентов в общежитии. Цель данной контрольной работы - развитие практических навыков программирования на языке С и работы в среде программирования Visual C++, а также формирование компьютерной грамотности.

1. Теоретические вопросы

Вложенные циклы.

Вложенным циклом называется цикл который находится внутри другого цикла. Механизм действия в данном случае будет такой, при каждой итерации внешнего цикла, будет выполняться вложенный цикл. Операторы break и continue во вложенных циклах относятся ко вложенному циклу, а не внешнему. Так, в случае выполнения оператора break во вложенном цикле, управление будет передано следующему оператору внешнего цикла, а при выполнении оператора continue во вложенном цикле, управление будет передано на начало следующей операции вложенного цикла. Приведём пример вложенного цикла:

```
do {
последовательность операторов;
    do {
    последовательность_операторов_вложенного_цикла;
    } while (условие_вложенного цикла);
последовательность операторов;
} while (условие_внешнего_цикла);
```

Порядок вычисления выражений.

В языке С каждая операция выполняются в определённом для неё порядке. Порядок вычисления определяется рангом операции и правилом ассоциативности для определённого ранга. Ранг операции и правило ассоциативности описаны в таблице приоритетов. Если к одному рангу относится несколько операций то порядок вычисления между такими операциями определяется правилом ассоциативности для данного ранга. Ассоциативность операций - это направление вычисления выражения: с права на лево либо наоборот. Если знак операции встречается в таблице дважды то

его меньший ранг соответствует унарной операции, а больший бинарной. Приведём данную таблицу:

Ранг	Операции	Ассоциативность
1	( ) [ ] -> .	→
2	! ~ + - ++ -- & * (тип)sizeof	←
3	* / %	→
4	+ -	→
5	<< >>	→
6	< <= >= >	→
7	== !=	→
8	&	→
9	^	→
10	|	→
11	&&	→
12	||	→
13	?:	←
14	= += -= *= /= %= &= |= ^= <<= >>=	←
15	,	→

2. Практическая часть

Задание на разработку программы в соответствии с индивидуальным заданием: необходимо изменить программную реализацию консольного приложения разработанного при выполнении контрольной №1 следующим образом:
необходимо предусмотреть использование многомерных массивов, динамическое использование памяти, использование указателей на функции, а также разделить проект на несколько компонентов, которые используются через директиву #include.

Листинг кода программы:

Создаём заголовочный файл header.h. В нём:

```
1  //объявляем символьную константу в которой храним количество символов
2  //в поле фио студента структуры Student
3  #define MAXFNAME 25
```

```c
//объявляем структуру
typedef struct tagStudent
{
    int     studNmb;          //номер в списке
    char    fname[MAXFNAME]; //фио студента
    int     roomNmb;          //номер комнаты
    int     floor;            //этаж
    int     marks[5][12];     //оценки студ-в за деж-во помесячно за 5 учебных лет #КР2

    //в этих переменных храним ссылку на следующую, предыдущую связанную структуру
    struct tagStudent *pnext, *pprev;
} Student;
```

```c
//объявляем прототипы функций используемых в программе
void print(Student *p, char (*fnc)(char *));
void getUser(Student *p);
Student* addEnd(Student *p, Student *end, int stnumb);
void printList(Student *p,int k);
void loadList(Student *p, char *file, Student **pend, Student **pbegin);
void saveList(Student *p, char *file);
int editStd(Student *p, Student **pend, Student **pbegin);
int deleteStd(Student *p, Student **pend, Student **pbegin);
int printMarks(Student *p);
int printReg(Student *p);
int addMarks(Student *p);
```

Создаём главный файл main.c. В нём:

```c
//подключаем необходимые заголовочные файлы стандартной библиотеки
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
//подключаем созданный заголовочный файл
#include "header.h"
```

```c
//пишем главную функцию программы
void main()
{
    //объявляем переменные
    //переменная для выбранного пользователем номера команды
    int n=0;
    //перем. для временного хранения стр-ры, для передачи её из одной функции в другую
    Student std;
    //указатели на первый и последний элемент связанных структур
    Student *begin=NULL, *end=NULL;
```

Организуем пользовательское меню.

```
    //пользовательское меню
    //1.добавить запись в список студентов 2.отредактировать запись в списке
    //3.сохр. список в файл 4.загруз. список из файла 5.вывести список на экран
    //6.выход
L:  printf("\n1.add    3.save   5.print  7.print marks\n2.edit   4.load   6.exit   8.print H/l reg\n");
    printf("Input the command number (from 1 to 6): ");
    //сохраняем введённое пользователем значение в переменной n
    scanf("%d", &n);

    //в зав-ти от введённого пользователем значения (1-5) запуск. соотв. функции,
    //при вводе '6' (6й команды) выходим без ошибки
    switch(n)
    {
    case 1: //принимаем значения от пользователя во временную структуру std
            getUser(&std);
            //связываем заполненную польз-ем стр-ру std с имеющимися структурами
            end=addEnd(&std,end,0);
            //при первом заполнении стр-ры указатели на 1 и последн. структуру равны
            if (begin==NULL) begin=end;
            break;
    case 2: //редактируем выбранную пользователем строку
            editStd(begin, &end, &begin);
            break;
    case 3: //сохраняем список в файл
            saveList(begin,"list.dat"); break;
    case 4: //загружаем список из файла
            loadList(begin,"list.dat",&end, &begin); break;
    case 5: //печатаем список на экран
            printList (begin,0); break;
    case 6: //завершаем программу без ошибки
            exit(0); break;
    case 7: //печатаем оценки студента
            printMarks (begin); break;
    case 8: //печатаем список студентов в верхнем или нижнем регистре
            printReg(begin); break;
    }
    //при вводе пользователем значения не равного 1,2,3,4,5,6,7,8
    //просим пользователя ввести команду ещё раз
    goto L;
}
```

Добавим ещё один модуль list.c. В нём опишем все остальные функции которые используются в главной функции main.

Опишем функции которые запускаются при вводе команды добавления записи в список ‘1’.

```c
//функция для приёма значений от пользователя
//вход параметр: указатель на структуру
void getUser(Student *p)
{
    Student tmp;
    int i,k;
    //поочерёдно запрашиваем и принимаем информацию от пользователя
    printf("\nInput name :");
    fflush(stdin);
    fgets(tmp.fname,MAXFNAME,stdin);
    //удаляем символ завершения строки кот. добавился функцией fgets
    tmp.fname[strlen(tmp.fname)-1]='\0';
    printf("Input floor: ");
    scanf("%d", &tmp.floor);
    fflush(stdin);
    printf("Input room number: ");
    scanf("%d", &tmp.roomNmb);
    //инициалзируем поля массива с оценками нулями
    for(i=0;i<5;i++)
    {
        for(k=0;k<12;k++) tmp.marks[i][k]=0;
    }
    //указатели на пред. след. структуру укажем позже
    tmp.pnext = tmp.pprev = NULL;
    //копируем структуру tmp в структуру по указателю p (вход. пар-р)
    *p = tmp;
}
```

```
//функция (Ф) для связывания заполненной польз-ем структуры с имеющимся списком
//структур. Ф принимает: заполненную структуру, указатель на последнюю
//структуру в списке, номер в списке который необходимо заполнить для
//добавляемого студента (т.к. данная ф-я еще используется при загрузке
//списка из файла). Ф возвращает: указатель на структуру который записывается в
//переменную end - указатель на последнюю структуру
Student* addEnd(Student *p, Student *end, int stnumb)
{
    //выделяем память для структуры, помещаем указатель на неё в pAdd
    Student *pAdd=(Student*)malloc(sizeof(Student));
    //заполняем память по адресу pAdd переданной, через параметр указатель p, структурой
    *pAdd=*p;
    if (end==NULL) //если добавляем первый элемент
    {
        //в указатель end помещаем указатель на заполненную структуру
        end=pAdd;
        //если переданный параметр stnumb равен нулю, тогда номеру в списке присваиваем 1
        //иначе значение равное переданному параметру stnumb
        if (stnumb==0) pAdd->studNmb=1;
        else pAdd->studNmb=stnumb;
    }
    else //если добавляем не первый элемент
    {
        end->pnext = pAdd;//связываем последний элемент с добавляемым
        pAdd->pprev=end;  //связываем добавляемый элемент с последним
        //если переданный параметр stnumb равен нулю, тогда номер в списке присваиваем
        //номер последней строки в списке плюс один
        if (stnumb==0) pAdd->studNmb=end->studNmb+1;
        else  pAdd->studNmb=stnumb;
        end= pAdd;
    }
    //возвращаем указатель на последний элемент
    return end;
}
```

Опишем функции, которые запускаются при вводе команды редактирования записи в списке ‘2’.

```
71  //функция для редактирования выбранных пользователем строк
72  //вход. пар-ры: указатель на первую структуру,
73  //указатель на указатель последней в списке структуры,
74  //указатель на указатель первой в списке структуры,
75  int editStd(Student *p, Student **pend, Student **pbegin)
76  {   //объявляем переменные
77      int n, k;
78      Student *s, std;
79      //выводим список на экран и предлагаем выбрать номер строки
80      printList (p,0);
81      printf("Input number of student: ");
82      scanf("%d", &n);
83      //находим структуру номер в списке кот. равен выбранному
84      while(p!=NULL) {
85      if (p->studNmb==n){ s=p; break;}
86      p=p->pnext; }
87      //предлагаем выбрать номер команды "что сделать со строкой"
88  L:  printf("\nWhat do you want to do with it? \n1.edit \n2.delete\
89   \n3.add marks \n4.exit to the main menu \nInput the command number : ");
90      scanf("%d", &k);
91      switch(k)
92      {
93      case 1: //при редактировании строки запрашиваем переввести данные
94              getUser(&std);//и сохраняем их в структуру std
95              //в выбранную пользователем строку вносим изменения
96              strncpy(s->fname,std.fname,sizeof(std.fname));
97              s->roomNmb=std.roomNmb; s->floor=std.floor;
98              goto M; //выход из цикла
99      case 2: //удаляем строку функцией deleteStd, передаём в неё
100             //указатель на удаляемую стр-ру, на первый и последний эл-ты списка
101             deleteStd(s, pend, pbegin); goto M; //и выходим из цикла
102     case 3: //запускаем функцию добавления оценок
103             addMarks(s); goto M;
104     case 4: goto M; //выход из цикла в главное меню
105     }
106     goto L;  //переспрашиваем номер команды если введённой значение не 1,2,3
107 M:  return 0;//выходим без кода об ошибке
108 }
```

```
307 //функция добавления оценок студентам
308 //параметр указатель на структуру типа Student
309 int addMarks(Student *p)//#КР2
310 {
311     int y,m,r,k;
312     //запрашиваем год,месяц за который необходмо добавить оценку
313     //и саму оценку
314 L:  printf("\nInput Year (1-5):");
315     scanf("%d", &y);
316     printf("\nInput Month (1-12):");
317     scanf("%d", &m);
318     printf("\nInput Mark (1-5):");
319     scanf("%d", &r);
320     //поправка на то что в массиве индексы начинаются с нуля
321     --y;
322     --m;
323     //вводим оценку
324     p->marks[y][m]=r;
325     //предоставляем возможность продолжить вводить оценки
326     printf("\nMark entered. \n1.add another \n2.exit to the main menu");
327     printf("\nInput the command number :");
328     scanf("%d", &k);
329     //выходим либо продолжаем вводить оценки
330     switch(k)
331     {
332     case 1: goto L;
333     case 2: goto M;
334     }
335 M:  return 0;//выходим без кода об ошибке
336 }
```

Опишем функцию по указателю deleteStd.

```
149  //удаление строки из списка, параметры идентичные функции editStd
150  int deleteStd(Student *p, Student **pend, Student **pbegin)
151  {
152      Student *pr=NULL, *pn=NULL;//указатели для пред. след. эл. списка
153      if (p->pnext==NULL) //если удаляемая строка последняя
154      {
155          pr=p->pprev; //сохраняем в pr указатель на предыд. эл. списка
156          if(pr!=NULL) //если предыд. эл. есть
157          {
158              pr->pnext=NULL; //обнуляем указатель в пред. эл-те
159              *pend=pr;       //предыдущий элемент объявляем последним
160
161          } else //если предыд. элемента нет (т.е. элемент в списке один)
162          {
163              *pend=NULL; //последний и первый элементы списка обнуляем
164              *pbegin=NULL;
165          }
166      }
167      else if (p->pprev==NULL) //если удаляемая строка первая
168      {
169          pn=p->pnext; //сохраняем в pn указатель на след. эл. списка
170          *pbegin=pn;  //указатель на первый элемент меняем на pn
171          pn->pprev=NULL; //в след эл-те обнуляем указатель на пред. эл-т
172      }
173      else //если удаляемая стр. ни первая ни последняя
174      {
175          pn=p->pnext; //сохраняем в pn указатель на след. эл. списка
176          pr=p->pprev; //сохраняем в pr указатель на предыд. эл. списка
177     //связываем указ-ли предыдущ. и след. эл-та так как тек. эл-т удаляем
178          pn->pprev=p->pprev;
179          pr->pnext=p->pnext;
180      }
181      free(p); //освобождаем память по указателю тек. элемента
182      p=NULL;  //обнуляем указатель
183      return 0; //выходим без кода ошибки
184  }
```

Опишем функцию, которая запускается при вводе команды сохранения списка в файл ‘3’.

```
147 //функция записи списка в файл
148 //параметр: указатель на первый элемент списка и название файла
149 void saveList(Student *p, char *file)
150 {
151     int i, k;
152     //объявляем указатель pf типа FILE, открываем/создаём файл
153     //функцией fopen (режим символьн. записи), и помещаем указатель на файл в pf
154     FILE *pf=fopen(file,"w");
155     //если файл успешно открыт/создан
156     if (pf!=NULL)
157     {
158         while(p!=NULL)//проходимся по всем элементам списка
159         {
160         //и записываем элементы структур в файл
161         fprintf(pf,"%d %s %d %d ", p->studNmb, p->fname, p->roomNmb, p->floor);
162         //записываем оценки
163         for(i=0;i<5;i++)
164         {
165         for(k=0;k<12;k++) fprintf(pf,"%d ", p->marks[i][k]);
166         }
167         //переносим указатель на сл. строку
168         fprintf(pf,"\n");
169         p=p->pnext;
170         }
171         fclose(pf);//закрываем файл
172     }
173 }
```

Опишем функцию, которая запускается при вводе команды загрузки списка из файла ‘4’.

```
//функция загрузки списка из файла, параметры:
//указатель на структуру типа Student, название файла, указатель на
//указ-ль на первый эл-т списка, указ-ль на указ-ль на последний эл-т списка
void loadList(Student *p, char *file, Student **pend, Student **pbegin)
{
    int i, k;
    Student tmp, *ptmp;
    //объявляем указатель pf типа FILE, открываем/создаём файл
    //функцией fopen (режим символьн. чтения), и помещаем указатель на файл в pf
    FILE *pf=fopen(file,"r");
    //если файл успешно открыт/создан
    if (pf!=NULL)
    {
        //очищаем текущий список структур
        while(p!=NULL)
        {
            ptmp=p->pnext;
            deleteStd(p, *pend, *pbegin);
            p=ptmp;
        }
        *pend=NULL;
        *pbegin=NULL;
        //пока не достигнут конец файла читаем данные во временную структуру tmp
        while(!feof(pf))
        {
            fscanf(pf,"%d %s %d %d", &tmp.studNmb, &tmp.fname, &tmp.roomNmb, &tmp.floor);
            //
            for(i=0;i<5;i++)
            {
            for(k=0;k<12;k++) fscanf(pf,"%d ", &tmp.marks[i][k]);
            }
            fscanf(pf,"\n");
            tmp.pnext = tmp.pprev = NULL;
            //запускаем функцию addEnd, передаём в неё указатель на заполненную временную структуру,
            //указатель на указатель последнего элемента списка, и номер записываемого элемента в списке
            //функция добавит полученную из файла структуру в список
            *pend=addEnd(&tmp,*pend,tmp.studNmb);
            //если элемент первый то последний элемент равен первому элементу
            if (*pbegin==NULL) *pbegin=*pend;
        }
        fclose(pf);//закрываем файл
    }
}
```

Опишем функцию, которая запускается при вводе команды печати списка на экран ‘5’.

```
//функция вывода списка на экран принимает указатель на структуру
//и вещественное число-флаг
void printList(Student *p, int k)//#КР2
{
    //объявляем переменную типа указатель на функцию
    //которая возвращает указатель на строку и принимает указатель на символ
    char (*fnc)(char *);
    //выводим шапку таблицы
    printf("\n%s %s %s %s \n","# ","Name                    ","Floor ","Room ");
    printf("%s \n","----------------------------------------");

    //если переданный параметр k равен 1, тогда
    //передаём в указатель fnc адрес функции strupr
    //которая преобразует символы строки в верхний регистр
    if (k==1) fnc=&strupr;
    //если переданный параметр k равен 2, тогда
    //передаём в указатель fnc адрес функции strlwr
    //которая преобразует символы строки в нижний регистр
    else if (k==2) fnc=&strlwr;
    //иначе записываем в указатель NULL
    else fnc=NULL;
    //циклом проходимся по элементам списка и выводим их на экран
    //ниже описанной функцией print
    while(p!=NULL)
    {
        print(p,fnc);
        p=p->pnext;
    }
}
```

```
//функция печати строки списка на экран
//параметры: указатель на структуру типа Student и
//указатель на функцию кот. возвращает char, принимает
//параметр указатель на char
void print(Student *p, char (*fnc)(char *))//#КР2
{
    //объявл. массив символов размером поял имя структуры Student
    char s [MAXFNAME];
    //копируем в s имя студента
    strcpy(s, p->fname);
    //если передали указатель на функцию (не NULL),
    //используем её для преобразования строки s
    if ((*fnc)!=NULL)
    {
    (*fnc)(s);
    }
    //печатаем поля структуры,вместо имени печатаем то что храниться в s
    printf("%-2d %-25s %-6d %-3d \n", p->studNmb, s, p->floor, p->roomNmb);
}
```

Опишем функцию, которая запускается при вводе команды печати оценок студента на экран ‘7’.

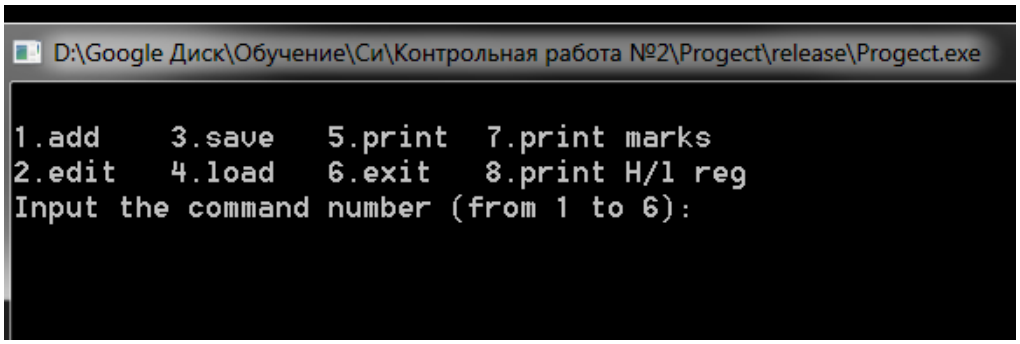
```
//функция печати оценок студентов
//параметр указатель на структуру типа Student
int printMarks(Student *p)//#КР2
{
    Student *s;
    int      i, k, n;
    //объявляем массив указателей на строки
    //инициализируем его названиями месяцев
    char *monthsArr[12]={"January","February","March","April","May","June",\
        "July","August","September","October","November","December"};
    //печатаем список студентов
    printList (p,0);
    //спрашиваем по какому студенту распечатать оценки
    printf("Input number of student: ");
    scanf("%d", &n);
    //находим структуру номер в списке кот. равен выбранному
    while(p!=NULL) {
    if (p->studNmb==n){ s=p; break;}
    p=p->pnext;}

    //выводим шапку таблицы
    printf("\nMarks of a student: #%-d, name:%-s, floor:%-d, room:%-d \n",
        s->studNmb, s->fname, s->floor, s->roomNmb);
    printf("%s \n","--------------------");
    printf("%s \n","   Months Year     ");
    printf("%s \n","          1 2 3 4 5");
    printf("%s \n","--------------------");
    //с помощью двух циклов выводим элементы многомерного массива
    //т.е. печатаем оценки студента
    for(k=0;k<12;k++)
    {
        printf("%9s", monthsArr[k]);
        for(i=0;i<5;i++) printf(" %d",s->marks[i][k]);
        printf("\n");
    }
    return 0;//выходим без кода об ошибке
}
```

Опишем функцию, которая запускается при вводе команды печати списка студентов в верхнем и нижнем регистре ‘8’.

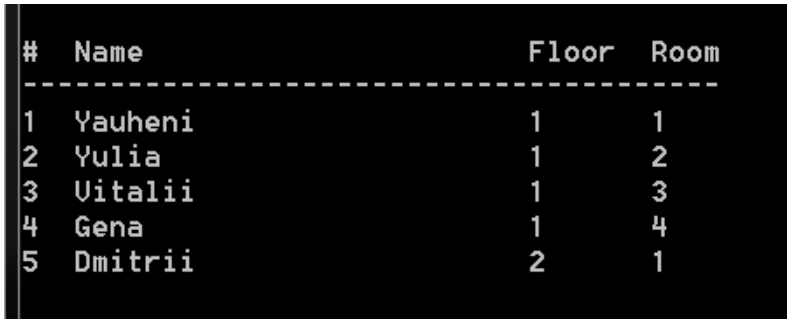
```
250 //функция печати списка студентов в верхнем и нижнем регистре
251 //параметр указатель на структуру типа Student
252 int printReg(Student *p)//#KP2
253 {
254     int      n;
255     //запрашиваем у пользователя тип регистра в котором вывести список
256     printf("\nWhat letter case do you want?");
257     printf("\n1.UPPER \n2.lower");
258     printf("\nInput the number (1-2):");
259     scanf("%d", &n);
260     //в зависимости от выбранного регистра передаём
261     //1 в функцию printList 2м параметром - если печатаем в верхнем регистре,
262     //2 - если печатаем в нижнем регистре
263     if (n==1) {printList(p, 1); return 0;}
264     else if (n==2) {printList(p, 2); return 0;}
265     //выходим в главное меню если ввод не равен 1 или 2
266     else return 0;
267 }
```

Интерфейс работы программы:

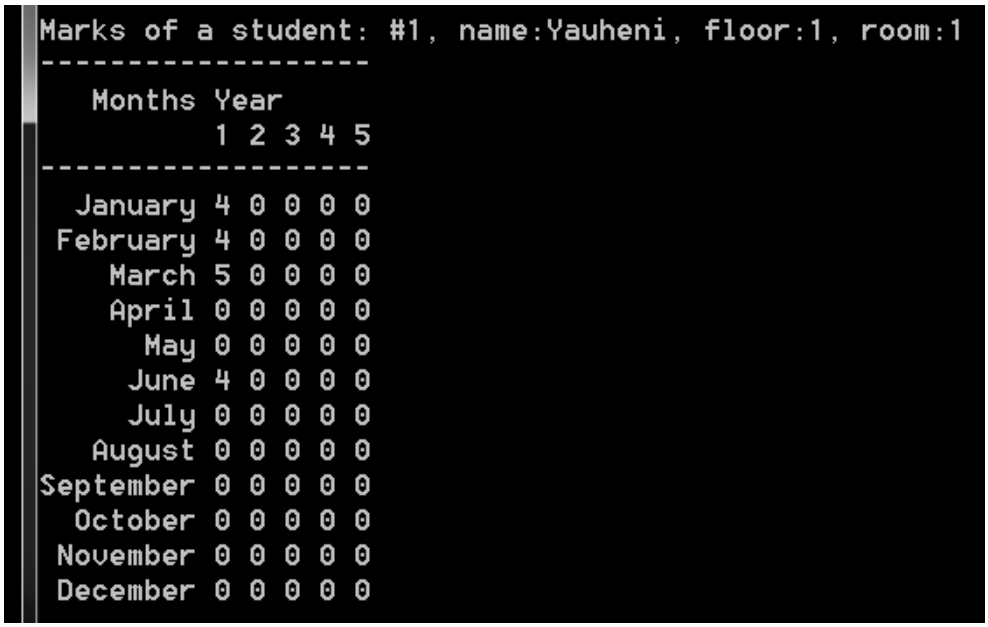
Главное меню:



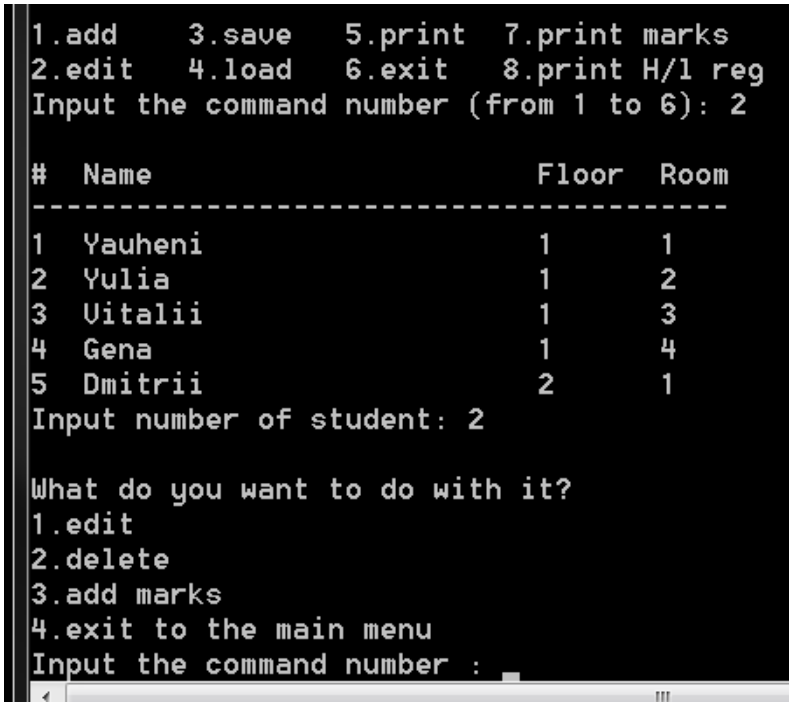
Печать списка студентов:



Печать оценок студента за дежурство:



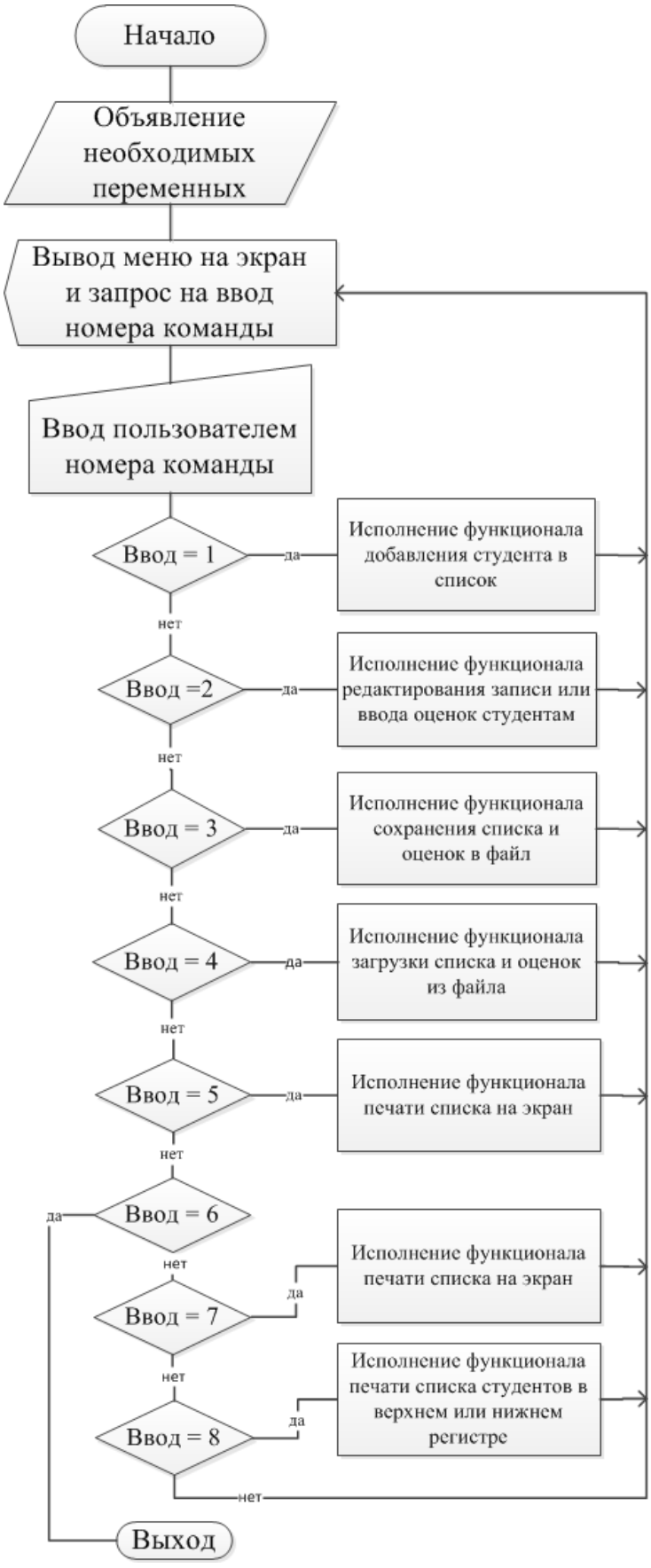
Редактирование записи о студенте:



Печать списка студентов в верхнем регистре:

```
1.add    3.save   5.print  7.print marks
2.edit   4.load   6.exit   8.print H/l reg
Input the command number (from 1 to 6): 8

What letter case do you want?
1.UPPER
2.lower
Input the number (1-2):1

#  Name                      Floor  Room
----------------------------------------
1  YAUHENI                   1      1
2  YULIA                     1      2
3  VITALII                   1      3
4  GENA                      1      4
5  DMITRII                   2      1
```

3. Блок-схема работы программы.



Заключение

В соответствии с индивидуальным заданием №2 доработана реализация консольной программы учёта студентов в общежитии, разработанной при выполнении контрольной работы №1. Отработаны навыки работы с многомерными массивами, указателями на функции, динамическим использованием памяти, разбиением проекта на несколько компонентов, которые используются через директиву #include. Предоставлены ответы на теоретические вопросы.

Литература

1. Подбельский В.В., Фомин С.С. Программирование на языке Си: Учебное пособие. 2-е доп. изд. - М.: Финансы и статистика – 600 с.: ил.
2. Березин Б.И., Березин С.Б. Начальный курс С и С++. –М.: Диалог - МИФИ, 1999 - 288 с.
3. Основы алгоритмизации и программирования. Язык Си: учеб. пособие / М. П. Батура, В. Л. Бусько, А. Г. Корбит, Т. М. Кривоносова. - Минск: БГУИР, 2007. - 240 с.: ил.