МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ФАКУЛЬТЕТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК
ИМЕНИ ПРОФЕССОРА Н.И.ЧЕРВЯКОВА

ЛАБОРАТОРНАЯ РАБОТА №15

Алгоритмизация и программирование

Файлы

Выполнил студент:
Сивко Иван Андреевич
студент 2 курса
группа ПМИ-б-о-23-2,
направление подготовки 01.03.02

Проверил:
Ассистент кафедры вычислительной
математики и кибернетики, к.ф.-м.н.,
Черкашина Анастасия Андреевна

2024 г.

Вариант 9

Цель: Совершенствование навыков в программировании с использованием указателей.

Задание 1

Работа с неструктурированными данными

1 Условие

Для исследования различных методов доступа к файлам данных необходимо выполнить следующие подготовительные действия:

1. **Создайте текстовый файл.**

   Содержимое файла:

   > У меня спросили: сколько будет x Опер y?
   > А я не знаю! A n Опер k? Тоже!
   > Помогите!

   Например:

   > У меня спросили: сколько будет 7 * 2?
   > А я не знаю! A 9 / 4? Тоже!
   > Помогите!

   Создайте файл с указанным содержимым в текстовом редакторе (например, в Блокноте).

2. **Обработайте данные.**

   Вам известна структура файла. Необходимо:

   - Вывести содержимое файла на экран.
   - Записать в выходной файл результаты в формате:

     x Опер y = Рез1
     n Опер k = Рез2

   Например:

   > 7 * 2 = 14
   > 9 / 4 = 2.25

3. Исходные данные берутся из таблицы согласно варианта:

Вар.	x	Опер	y	n	Опер	k	Вар.	x	Опер	y	n	Опер	k
1	15	+	4	7	*	8	***9***	23	+	37	13	*	5
2	18	-	19	18	/	4	***10***	7	-	42	37	/	6
3	9	*	6	56	-	37	***11***	34	*	3	14	-	53
4	23	/	5	31	+	29	***12***	21	/	5	11	+	77
5	7	+	23	14	/	4	***13***	12	+	25	20	/	6
6	34	-	67	11	*	3	***14***	15	-	72	54	*	2
7	21	*	2	20	+	11	***15***	18	*	4	7	+	55
8	12	/	5	54	-	32	***16***	9	/	18	18	-	81

2 Код:

```
#include <stdio.h>

int main(void) {
    FILE *file;
    int x, y, n, k;
    char op1, op2;

    file = fopen("input.txt", "r+");
    fscanf(file, "У меня спросили: сколько будет %d %c %d ?\n", &x, &op1, &y);
    fscanf(file, "А я не знаю! А %d %c %d Тоже!", &n, &op2, &k);
    fprintf(file, "%d %c %d = %d\n"
           "%d %c %d = %d\n",
           x, op1, y, op1=='+'?x+y:op1=='-'?x-y:op1=='*'?x*y:op1=='/'?x/y:0,
           n, op2, k, op2=='+'?k+n:op2=='-'?k-n:op2=='*'?k*n:op2=='/'?k/n:0);
    fclose(file);
    return 0;
}
```

3 Результат работы программы

```
[john@arch cpp]$ cat input.txt
У меня спросили: сколько будет 23 + 37 ?
А я не знаю! А 13 * 5 Тоже!
[john@arch cpp]$ ./a.out
[john@arch cpp]$ cat input.txt
У меня спросили: сколько будет 23 + 37 ?
А я не знаю! А 13 * 5 Тоже!23 + 37 = 60
13 * 5 = 65
[john@arch cpp]$
```

Задание 2

Работа со структурированными данными

1 Условие

Используя полученную при выполнении лабораторной работы 14 программу, реализовать возможность сохранения данных в файл с последующим чтением из файла введенных данных.

2 Код:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#define DELIM ";"
#define MAXLINE 1024

enum {
    EXIT,
    DISPLAY, ADD, SEARCH1, SEARCH2
};

void writerows(FILE *file, char ***rows, int nrow, int ncol);
char ***readrows(FILE *file, int *nrow, int *ncol);
```

```
char **strparce(const char *line, const char *delim, int *ncol);
void printrows(char ***rows, const long *col_len, int nrow, int ncol, int flag);
char ***search_in_rows(char ***rows, const char *smsg, int *snrow, int (*sfunc)(char**, const char
long *get_col_len(char ***rows, int nrow, int ncol);
static long rtrimlen(const char *s);

int is_valid(const char *line, const char *delim);
static int is_valid_4num(const char *line);

int match_by_num(char **row, const char *smsg);
int match_by_mileage(char **row, const char *smsg);

int
main(void) {
    char buffer[MAXLINE], ***rows, ***srows;
    int is_continue, nrow, snrow, ncol;
    long *col_len;
    FILE *file;

    file = fopen("table1.csv", "r+");
    rows = readrows(file, &nrow, &ncol);
    /* init a header if it doesn't exist */
    if (rows==NULL) {
        rows = (char***)realloc(rows, (nrow+1)*sizeof(char**));
        rows[nrow++] = strparce(strdup(
                    "car model;"
                    "registration number;"
                    "owner's last name;"
                    "owner's initials;"
                    "quarterly mileage (array of four elements)"), ";", &ncol);
    }
    for (is_continue=1; is_continue; ) {
        printf("Enter:\n"
                "1: to display rows,\n"
                "2: to add a 1 row,\n"
                "3: to search by model,\n"
                "4: to search by mileage,\n"
                "anything else: to record and exit\n");
        fgets(buffer, sizeof(buffer), stdin);
        switch (feof(stdin)
                ? EXIT
                : atoi(buffer))
        {
        case DISPLAY:
            col_len = get_col_len(rows, nrow, ncol);
            if (nrow>1)
                printrows(rows, col_len, nrow, ncol, 1);
            else
                printf("\nempty\n\n");
            break;
        case ADD:
            fgets(buffer, sizeof(buffer), stdin);
            while (!is_valid(strdup(buffer), ";") && !feof(stdin)) {
                printf("try again\n");
                fgets(buffer, sizeof(buffer), stdin);
            }
            if (!feof(stdin)) {
                rows = (char***)realloc(rows, (nrow+1)*sizeof(char**));
                rows[nrow++] = strparce(buffer, ";", &ncol);
            }
            break;
```

```
        case SEARCH1:
            snrow = nrow;
            fgets(buffer, sizeof(buffer), stdin);
            srows = search_in_rows(rows, strndup(buffer, rtrimlen(buffer)), &snrow, match_by_num);
            if (srows!=NULL) {
                col_len = get_col_len(rows, nrow, ncol);
                printrows(rows, col_len, 1, ncol, 1);
                printrows(srows, col_len, snrow, ncol, 0);
            } else
                printf("not found\n");
            break;
        case SEARCH2:
            snrow = nrow;
            fgets(buffer, sizeof(buffer), stdin);
            srows = search_in_rows(rows, strndup(buffer, rtrimlen(buffer)), &snrow, match_by_milea
            if (srows!=NULL) {
                col_len = get_col_len(rows, nrow, ncol);
                printrows(rows, col_len, 1, ncol, 1);
                printrows(srows, col_len, snrow, ncol, 0);
            } else
                printf("not found\n");
            break;
        default:
            printf("\nsave\n");
            writerows(file, rows, nrow, ncol);
            is_continue = 0;
            break;
        }
    }

    fclose(file);
    return 0;
}

void
writerows(FILE *file, char ***rows, int nrow, int ncol) {
    int i, j;
    if (rows==NULL)
        return;
    rewind(file);
    ftruncate(fileno(file), 0);
    for (i=0; i<nrow; ++i)
        for (j=0; j<ncol; ++j)
            fprintf(file, (j<ncol-1)?"%s"DELIM:"%s\n", rows[i][j]);
}

char **
strparce(const char *line, const char *delim, int *ncol) {
    int col;
    char *token, **row=NULL;
    if (*ncol!=0)
        row = (char**)malloc(*ncol*sizeof(char*));
    for (col=0, token=strtok(line, delim);
            token!=NULL;
            ++col, token=strtok(NULL, delim))
    {
        if (*ncol==0)
            row = (char**)realloc(row, (col+1)*sizeof(char*));
        else if (col>=*ncol)
            break;
        row[col] = strndup(token, rtrimlen(token));
```

```
    }
    if (*ncol==0)
        *ncol = col;
    return row;
}

char ***
readrows(FILE *file, int *nrow, int *ncol) {
    int col;
    char ***rows, *token, line[1024];
    rewind(file);
    for (*nrow=*ncol=0, rows=NULL;
            fgets(line, sizeof(line), file)!=NULL; // ‘feof‘ maybe crash
            ++*nrow)
    {
        rows = (char***)realloc(rows, (*nrow+1)*sizeof(char**));
        rows[*nrow] = strparce(line, DELIM, ncol);
    }
    if (*nrow==0)
        return NULL;
    return rows;
}

void
printrows(char ***rows, const long *col_len, int nrow, int ncol, int flag) {
    int i, j, k, n;
    for (i=n=0; i<nrow; ++i) {
        if (n<2&&flag) {
            for (j=0; j<ncol; ++j) {
                putchar('+');
                for (k=0; k<col_len[j]+2; ++k)
                    putchar('-');
            }
            printf("+\n");
            ++n;
        }
        for (j=0; j<ncol; ++j)
            printf("| %-*s", (int)col_len[j]+1, rows[i][j]);
        printf("|\n");
    }
    if (n<=2) {
        for (j=0; j<ncol; ++j) {
            putchar('+');
            for (k=0; k<col_len[j]+2; ++k)
                putchar('-');
        }
        printf("+\n");
    }
}

long
rtrimlen(const char *s) {
    long len;
    for (len=(long)strlen(s)-1; len>=0 && isspace(s[len]); --len)
        ;
    return len+1;
}

long *
get_col_len(char ***rows, int nrow, int ncol) {
    int i, j;
```

```
    long *col_len;
    col_len = (long*)malloc(ncol*sizeof(long));
    for (i=0; i<nrow; ++i)
        for (j=0; j<ncol; ++j) {
            if (i==0)
                col_len[j] = 0;
            if (strlen(rows[i][j]) > col_len[j])
                col_len[j] = strlen(rows[i][j]);
        }
    return col_len;
}

char ***
search_in_rows(char ***rows, const char *smsg, int *snrow, int (*sfunc)(char**, const char*)) {
    char ***srows;
    int i, count;
    for (i=1, count=0, srows=NULL; i<*snrow; ++i) {
        if (sfunc(rows[i], smsg)) {
            srows = (char***)realloc(srows, (count+1)*sizeof(char**));
            srows[count] = rows[i];
            count++;
        }
    }
    *snrow = count;
    return srows;
}

int
match_by_num(char **row, const char *smsg) {
    return strlen(smsg)>0 && strstr(row[0], smsg) == row[0];
}

int
match_by_mileage(char **row, const char *smsg) {
    int i, summ, snum;
    char *endptr;
    strtok(strdup(row[4]), " \t\n");
    for (i=1, summ=0; i<4; ++i)
        if (i==1 || i==2)
            summ+=atoi(strtok(NULL, " \t\n"));
        else
            strtok(NULL, " \t\n");
    snum = strtol(smsg, &endptr, 10);
    if (endptr!=smsg && (isspace(*endptr) || *endptr=='\0'))
        return summ<=snum;
    return 0;
}

int
is_valid(const char *line, const char *delim) {
    int i;
    char *token, *col2;
    for (i=0, col2=NULL, token=strtok(strdup(line), delim);
            token!=NULL;
            ++i, token=strtok(NULL, delim))
        if (i>=5) {
            return 0;
        }
        else if (i==4)
            col2=strdup(token);
    return col2==NULL?0:is_valid_4num(col2);
```

```
}

int
is_valid_4num(const char *line) {
    int i;
    char *token, *endptr;
    for (i=0, token=strtok(strdup(line), " \t\n");
            token!=NULL;
            ++i, token=strtok(NULL, " \t\n"))
    {
        strtol(token, &endptr, 10);
        if (!(i<4 && endptr!=token &&
             (*endptr==' ' ||
              *endptr=='\t' ||
              *endptr=='\n' ||
              *endptr=='\0')))
            return 0;
    }
    return 1;
}
```

3 Результат работы программы:

```
Enter:
1: to display rows,
2: to add a 1 row,
3: to search by model,
4: to search by mileage,
anything else: to record and exit
1
+-------------+-------------------+-----------------+-----------------+-----------------------------------------+
| car model   | registration number | owner's last name | owner's initials | quarterly mileage (array of four elements) |
+-------------+-------------------+-----------------+-----------------+-----------------------------------------+
| Toyota      | BNXR45            | Peterson        | PI              | 2345 3456 4567 5678                     |
| Honda       | JKZQ78            | Johnson         | JA              | 1234 2345 3456 4567                     |
| Ford        | XMWP23            | Sidorov         | SP              | 3456 4567 5678 6789                     |
| Chevrolet   | LNQY61            | Kuznetsov       | KA              | 2345 3456 4567 5678                     |
| BMW         | PZRJ92            | Smirnov         | SD              | 1234 2345 3456 4567                     |
| Audi        | FWKX34            | Popov           | PS              | 3456 4567 5678 6789                     |
| Mercede-Benz | DQTM78           | Sokolov         | SA              | 2345 3456 4567 5678                     |
| Toyota      | BNXR45            | Peterson        | PI              | 2345 3456 4567 5678                     |
| Honda       | JKZQ78            | Johnson         | JA              | 1234 2345 3456 4567                     |
| BMW         | FWKX34            | Sokolov         | SA              | 1234 2345 3456 4567                     |
| Toyota      | PZRJ92            | Kuznetsov       | KA              | 2345 0 0 4567                           |
| Chevrolet   | PZRJ92            | Johnson         | JA              | 3456 4567 5678 6789                     |
+-------------+-------------------+-----------------+-----------------+-----------------------------------------+
Enter:
1: to display rows,
2: to add a 1 row,
3: to search by model,
4: to search by mileage,
anything else: to record and exit
2
Chevrolet;LNQY61;Kuznetsov;KA;2345 3456 4567 5678
Enter:
1: to display rows,
2: to add a 1 row,
3: to search by model,
4: to search by mileage,
anything else: to record and exit
1
+-------------+-------------------+-----------------+-----------------+-----------------------------------------+
| car model   | registration number | owner's last name | owner's initials | quarterly mileage (array of four elements) |
+-------------+-------------------+-----------------+-----------------+-----------------------------------------+
| Toyota      | BNXR45            | Peterson        | PI              | 2345 3456 4567 5678                     |
| Honda       | JKZQ78            | Johnson         | JA              | 1234 2345 3456 4567                     |
| Ford        | XMWP23            | Sidorov         | SP              | 3456 4567 5678 6789                     |
| Chevrolet   | LNQY61            | Kuznetsov       | KA              | 2345 3456 4567 5678                     |
| BMW         | PZRJ92            | Smirnov         | SD              | 1234 2345 3456 4567                     |
| Audi        | FWKX34            | Popov           | PS              | 3456 4567 5678 6789                     |
| Mercede-Benz | DQTM78           | Sokolov         | SA              | 2345 3456 4567 5678                     |
| Toyota      | BNXR45            | Peterson        | PI              | 2345 3456 4567 5678                     |
| Honda       | JKZQ78            | Johnson         | JA              | 1234 2345 3456 4567                     |
| BMW         | FWKX34            | Sokolov         | SA              | 1234 2345 3456 4567                     |
| Toyota      | PZRJ92            | Kuznetsov       | KA              | 2345 0 0 4567                           |
| Chevrolet   | PZRJ92            | Johnson         | JA              | 3456 4567 5678 6789                     |
| Chevrolet   | LNQY61            | Kuznetsov       | KA              | 2345 3456 4567 5678                     |
+-------------+-------------------+-----------------+-----------------+-----------------------------------------+
Enter:
1: to display rows,
2: to add a 1 row,
3: to search by model,
4: to search by mileage,
anything else: to record and exit
```

```
Enter:
1: to display rows,
2: to add a 1 row,
3: to search by model,
4: to search by mileage,
anything else: to record and exit
3
Toy
+-------------+---------------------+-------------------+-----------------+-----------------------------------------+
| car model   | registration number | owner's last name | owner's initials | quarterly mileage (array of four elements) |
+-------------+---------------------+-------------------+-----------------+-----------------------------------------+
| Toyota      | BNXR45              | Peterson          | PI              | 2345 3456 4567 5678                     |
| Toyota      | BNXR45              | Peterson          | PI              | 2345 3456 4567 5678                     |
| Toyota      | PZRJ92              | Kuznetsov         | KA              | 2345 0 0 4567                           |
+-------------+---------------------+-------------------+-----------------+-----------------------------------------+
Enter:
1: to display rows,
2: to add a 1 row,
3: to search by model,
4: to search by mileage,
anything else: to record and exit
4
1000
+-------------+---------------------+-------------------+-----------------+-----------------------------------------+
| car model   | registration number | owner's last name | owner's initials | quarterly mileage (array of four elements) |
+-------------+---------------------+-------------------+-----------------+-----------------------------------------+
| Toyota      | PZRJ92              | Kuznetsov         | KA              | 2345 0 0 4567                           |
+-------------+---------------------+-------------------+-----------------+-----------------------------------------+
Enter:
1: to display rows,
2: to add a 1 row,
3: to search by model,
4: to search by mileage,
anything else: to record and exit
```

Задание 3

1 Условие

В соответствии с вариантом написать и отладить программу:

Дана информация о пяти больных. Структура имеет вид: фамилия, возраст, пол, давление. Вывести данные о больных с повышенным давлением (более 140) и подсчитать их количество.

2 Код:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#define MAXLINE 1024
#define DELIM ";"

char ***readrows(FILE *file, int *nrow, int *ncol);
char **strparce(const char *line, const char *delim, int *ncol);
void printrows(char ***rows, const long *col_len, int nrow, int ncol, int flag);
char ***search_in_rows(char ***rows, const char *smsg, int *snrow, int (*sfunc)(char**, const char
long *get_col_len(char ***rows, int nrow, int ncol);
static long rtrimlen(const char *s);

int match_by_num(char **row, const char *smsg);

int main(void) {
    char buffer[MAXLINE], ***rows, ***srows;
    int row, nrow, snrow, ncol;
    long *col_len;
    FILE *file;

    file = fopen("table2.csv", "r");
    rows = readrows(file, &nrow, &ncol);
    fclose(file);

    snrow = nrow;
    srows = search_in_rows(rows, strdup("120"), &snrow, match_by_num);
    col_len = get_col_len(rows, nrow, ncol);
    printrows(rows, col_len, 1, ncol, 1);
    printrows(srows, col_len, snrow, ncol, 0);
    return 0;
}
```

```
char **
strparce(const char *line, const char *delim, int *ncol) {
    int col;
    char *token, **row=NULL;
    if (*ncol!=0)
        row = (char**)malloc(*ncol*sizeof(char*));
    for (col=0, token=strtok(line, delim);
            token!=NULL;
            ++col, token=strtok(NULL, delim))
    {
        if (*ncol==0)
            row = (char**)realloc(row, (col+1)*sizeof(char*));
        else if (col>=*ncol)
            break;
        row[col] = strndup(token, rtrimlen(token));
    }
    if (*ncol==0)
        *ncol = col;
    return row;
}

char ***
readrows(FILE *file, int *nrow, int *ncol) {
    int col;
    char ***rows, *token, line[1024];
    rewind(file);
    for (*nrow=*ncol=0, rows=NULL;
            fgets(line, sizeof(line), file)!=NULL; // ‘feof‘ maybe crash
            ++*nrow)
    {
        rows = (char***)realloc(rows, (*nrow+1)*sizeof(char**));
        rows[*nrow] = strparce(line, DELIM, ncol);
    }
    if (*nrow==0)
        return NULL;
    return rows;
}


void
printrows(char ***rows, const long *col_len, int nrow, int ncol, int flag) {
    int i, j, k, n;
    for (i=n=0; i<nrow; ++i) {
        if (n<2&&flag) {
            for (j=0; j<ncol; ++j) {
                putchar('+');
                for (k=0; k<col_len[j]+2; ++k)
                    putchar('-');
            }
            printf("+\n");
            ++n;
        }
        for (j=0; j<ncol; ++j)
            printf("| %-*s", (int)col_len[j]+1, rows[i][j]);
        printf("|\n");
    }
    if (n<=2) {
        for (j=0; j<ncol; ++j) {
            putchar('+');
            for (k=0; k<col_len[j]+2; ++k)
                putchar('-');
```

```
        }
        printf("+\n");
    }
}

long *
get_col_len(char ***rows, int nrow, int ncol) {
    int i, j;
    long *col_len;
    col_len = (long*)malloc(ncol*sizeof(long));
    for (i=0; i<nrow; ++i)
        for (j=0; j<ncol; ++j) {
            if (i==0)
                col_len[j] = 0;
            if (strlen(rows[i][j]) > col_len[j])
                col_len[j] = strlen(rows[i][j]);
        }
    return col_len;
}

char ***
search_in_rows(char ***rows, const char *smsg, int *snrow, int (*sfunc)(char**, const char*)) {
    char ***srows;
    int i, count;
    for (i=1, count=0, srows=NULL; i<*snrow; ++i) {
        if (sfunc(rows[i], smsg)) {
            srows = (char***)realloc(srows, (count+1)*sizeof(char**));
            srows[count] = rows[i];
            count++;
        }
    }
    *snrow = count;
    return srows;
}

int
match_by_num(char **row, const char *smsg) {
    return atoi(row[3])>=atoi(smsg);
}

long
rtrimlen(const char *s) {
    long len;
    for (len=(long)strlen(s)-1; len>=0 && isspace(s[len]); --len)
        ;
    return len+1;
}
```

3 Результат работы программы:

```
[john@arch data1]$ cat table2.csv
Last Name;Age;Gender;Blood Pressure
Smith;35;Male;120
Johnson;28;Female;110
Williams;42;Male;130
Brown;19;Female;100
Davis;50;Male;140
[john@arch data1]$ ./a.out
+-----------+-----+--------+----------------+
| Last Name | Age | Gender | Blood Pressure |
+-----------+-----+--------+----------------+
| Smith     | 35  | Male   | 120            |
| Williams  | 42  | Male   | 130            |
| Davis     | 50  | Male   | 140            |
+-----------+-----+--------+----------------+
```