Министерство образования и науки Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
—
Институт компьютерных наук и технологий
Кафедра «Информационная безопасность компьютерных систем»

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3
«Список учеников»

По дисциплине «Методы программирования»

Выполнили А.Э.Палёный
Студент гр. 13508/13 А.Романов
_____

Проверил
Преподаватель
_____ В.Б.Вагисаров

Санкт-Петербург
2016

Цель

Создать программу, которая будет сохранять, и выводить список учеников.
В программе должна быть возможность сортировки учеников по дате рождения и имени. А также надо выводить учеников из определённой группы.

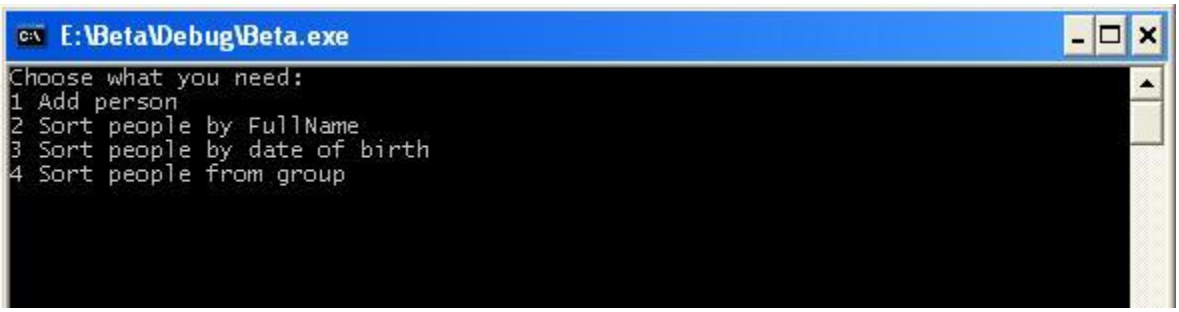
Ход работы

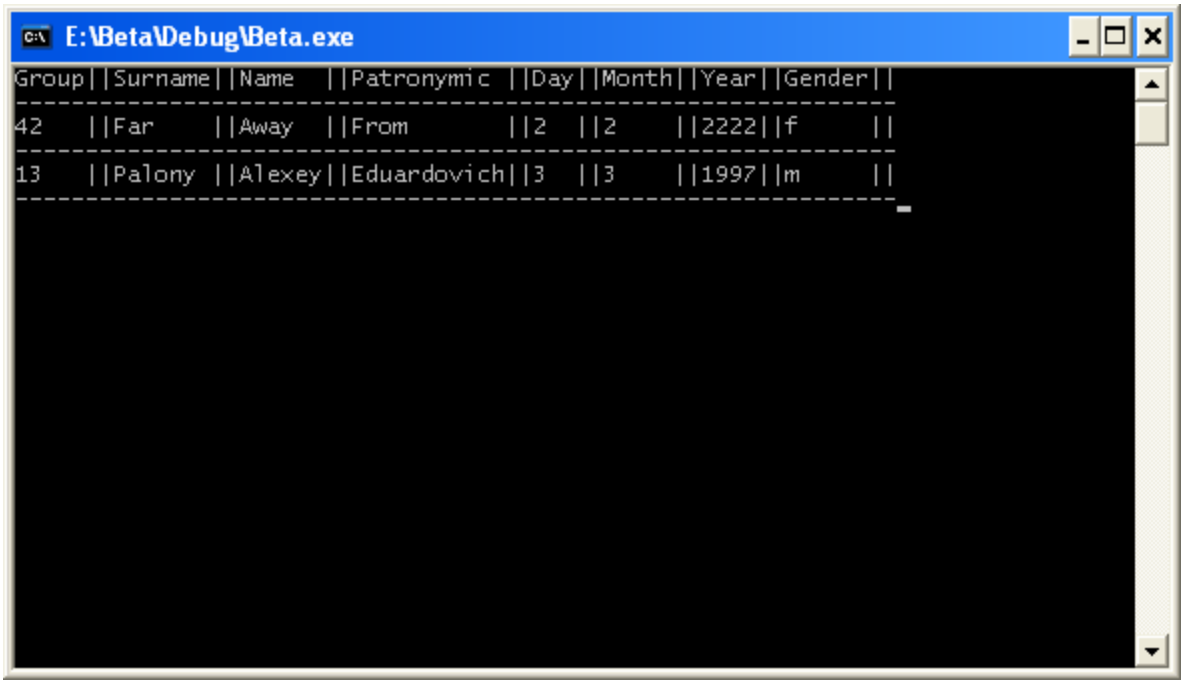
Программа получилась достаточно объёмной, но в тоже время простой.
Итак, алгоритм работы программы:
В начале, программа пытается открыть файл с данными учеников. Далее пользователь вводит информацию об ученике, и программа сразу записывает эту информацию в файл, а также добавляет информацию об ученике в два общих дерева сортировки.
После этого, пользователь имеет возможность отсортировать учеников по:

1) Дате рождения
2) Имени
3) А также вывести всех учеников из определённой группы.

Первые два пункта реализованы достаточно просто, но в тоже время эффективно. Для сортировки учеников по определённому критерию, уже изначально программа составило два условных дерева и хранит их в памяти. По этому, когда мы выбираем одну из сортировок, программе нужно совершить инфиксный обход дерева и отсортированный список будет уже готов.
Третий пункт тоже очень простенький. Мы сканируем файл с данными учеников, пытаясь, обнаружить людей из определённой группы и выводим результат пользователю.

Результаты работы программы





Исходный код программы

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

struct People
{
	struct People *left;  //less
	struct People *right; //more
	struct People *izquierda;
	struct People *derecho;
	int group;
	char *surname;
	char *name;
	char *patronymic;
	int day;
	int month;
	int year;
	char gender;
};

struct Size
{
	int group;
	int surname;
	int name;
	int patronymic;
	int day;
	int month;
	int year;
};

void Entr(char **str, int *count)
{
	int i;
	int fix=0;
	char ch;
	*count=0;
	*str=(char*)malloc(sizeof(char));
```

```
	for(i=1; ((ch = getchar()) != '\n') || (fix!=1); i++)
	{
		fix=1;
		if(ch!='\n')
		{
			*count+=1;
			*str=(char*)realloc(*str,sizeof(char)*(i+1));
			*(*str+i-1)=ch;
		}else	i--;
	}
	*(*str+i-1)=0;
}

void destr(struct People *Head)
{
	if(Head==NULL)	return;

	destr(Head->left);
	destr(Head->right);

	if(Head->surname!=NULL)	free(Head->surname);
	Head->surname=NULL;
	if(Head->name!=NULL)	free(Head->name);
	Head->name=NULL;
	if(Head->patronymic!=NULL)		free(Head->patronymic);
	Head->patronymic=NULL;
	free(Head);
	return;
}

short IsValueInt(char *str, int *result)
{
	int i,val;
	char cmp[10]={'2','1','4','7','4','8','3','6','4','7'};
	char Mcmp[11]={'-','2','1','4','7','4','8','3','6','4','8'};
	short flag=1;

	val=((*str=='-') ? 11:10);
	if(val==10)		*result=((*str>*cmp) ? -1:(*str==*cmp) ? 0:-2);
	else			*result=((*str>*Mcmp) ? -1:(*str==*Mcmp) ? 0:-2);
	flag+=*result;
	for(i=0; i<val; i++)
	{
		if(str[i]==0)		break;
		if(str[i]<47 && str[i]!='-' || str[i]>58)	return 0;
		if(flag>0 && i!=0){
			if(val==10){
				if(str[i]<=cmp[i]){
					if(str[i]<cmp[i])	flag=0;
				}else	*result=-1;
			}else{
				if(str[i]<=Mcmp[i]){
					if(str[i]<Mcmp[i])	flag=0;
				}else	*result=-1;
			}
		}
	}
	if(*result==-1 && i==val)	return 0;
	*result = atoi(str);
	return 1;
}

int Matching(struct People *Basical, struct People *From, short Operation)//Basical is Older
{
	int i;
	if(Operation == 1)
```

```
	{
		if(Basical->year == From->year)
		{
			if(Basical->month == From->month)
					return (Basical->day < From->day) ? 1:0;
			else	return (Basical->month < From->month) ? 1:0;
		}else		return (Basical->year < From->year) ? 1:0;
	}else{
		i=strcmp(Basical->surname, From->surname);
		if(i==0)
		{
			i=strcmp(Basical->name, From->name);
			if(i==0)
					return (strcmp(Basical->patronymic, From->patronymic)==0) ? 1:0;
			else	return (i==1) ? 1:0;
		}else		return (i==1) ? 1:0;
	}
}

void Bypass(struct People *Head, int flag, int key, int Operation, int group, int lname, int name, int pname,
	int day, int month, int year)
{
	int i,j;
	if(Head == NULL) return;
	j=group+lname+name+pname+day+month+year+22;

	if(Operation)
		Bypass(Head->left, flag, key, Operation, group, lname, name, pname, day, month, year);
	else
		Bypass(Head->izquierda, flag, key, Operation, group, lname, name, pname, day, month, year);

	if(flag)
	{
		if(key==Head->group)
		{
			printf("\n%-*d||%-*s||",group,Head->group,lname,Head->surname);
			printf("%-*s||%-*s||",name,Head->name,pname,Head->patronymic);
			printf("%-*d||%-*d||",day,Head->day,month,Head->month);
			printf("%-*d||%-*c||\n",year,Head->year,6,Head->gender);
			for(i=0; i<j; i++)	printf("-");
		}
	}else{
		printf("\n%-*d||%-*s||",group,Head->group,lname,Head->surname);
		printf("%-*s||%-*s||",name,Head->name,pname,Head->patronymic);
		printf("%-*d||%-*d||",day,Head->day,month,Head->month);
		printf("%-*d||%-*c||\n",year,Head->year,6,Head->gender);
		for(i=0; i<j; i++)	printf("-");
	}

	if(Operation)
		Bypass(Head->right, flag, key, Operation, group, lname, name, pname, day, month, year);
	else
		Bypass(Head->derecho, flag, key, Operation, group, lname, name, pname, day, month, year);
	return;
}

void Add(struct People *Head,struct People *Current, short Operation)
{
	int i;
	struct People *prev;

	if(Head==NULL){ Head=Current;	return;}
	if(Operation)
	{
		if(Head == Current || Head == NULL)  return;
		while(Head!=NULL)
		{
```

```
                        prev=Head;
                        i=Matching(Head,Current,Operation);
                        if(i == 1) {        Head=Head->left;}
                        else               {        Head=Head->right;}
                }
                if(i == 1) prev->left=Current;
                else               prev->right=Current;
        }else{
                if(Head == Current || Head == NULL)  return;
                while(Head!=NULL)
                {
                        prev=Head;
                        i=Matching(Head,Current,Operation);
                        if(i == 1) {        Head=Head->izquierda;}
                        else               {        Head=Head->derecho;}
                }
                if(i == 1) prev->izquierda=Current;
                else               prev->derecho=Current;
        }
        return;
}

void AddPerson(struct People *Head, int *group, int *lname, int *name, int *pname,
        int *day, int *month, int *year)
{
        FILE *Output;
        char *str;
        int DSize;
        int CountPeople=0;
        int IsNew=0;
        char c;

        Output = fopen("data.bin", "rb+");
        if(Output == 0)
        {
                Output=NULL;
                Output = fopen("data.bin", "wb+");
                IsNew++;
        }

        if(!IsNew)
        {
                fread(&CountPeople,sizeof(int),1,Output);
                fseek( Output , -1*sizeof(int) , SEEK_CUR);
        }
        CountPeople++;
        fwrite(&CountPeople,sizeof(int),1,Output);
        fseek(Output , 0 , SEEK_END);
        //Íîìåð ãðóïïû
        for(;;){
                printf("Enter the number of group\n");
                Entr(&str, &DSize);
                if(*group < DSize) *group=DSize;
                if(IsValueInt(str, &DSize))    break;
                printf("Sorry try again\n");
        }
        Head->group=DSize;
        fwrite(&DSize, sizeof(int), 1, Output);
        //Ôàìèëèÿ
        printf("Enter the surname\n");
        Entr(&str, &DSize);
        if(*lname < DSize) *lname=DSize;
        fwrite(&DSize, sizeof(int), 1, Output);
        Head->surname=str;
        fwrite(str, sizeof(char), DSize, Output);
        //Èìÿ
        printf("Enter the name\n");
```

```
	Entr(&str, &DSize);
	if(*name < DSize)  *name=DSize;
	fwrite(&DSize, sizeof(int), 1, Output);
	Head->name=str;
	fwrite(str, sizeof(char), DSize, Output);
	//îò÷åñòâî
	printf("Enter the patronymic\n");
	Entr(&str, &DSize);
	if(*pname < DSize)		*pname=DSize;
	fwrite(&DSize, sizeof(int), 1, Output);
	Head->patronymic=str;
	fwrite(str, sizeof(char), DSize, Output);

	for(;;){
		printf("Enter number of day from born\n");
		Entr(&str, &DSize);
		if(*day < DSize)    *day=DSize;
		if(IsValueInt(str, &DSize) && DSize>0 && DSize<32)    break;
		printf("Sorry try again\n");
	}
	Head->day=DSize;
	fwrite(&DSize, sizeof(int), 1, Output);
	for(;;){
		printf("Enter number of month from born\n");
		if(*month < DSize)*month=DSize;
		Entr(&str, &DSize);
		if(IsValueInt(str, &DSize) && DSize>0 && DSize<13)    break;
		printf("Sorry try again\n");
	}
	Head->month=DSize;
	fwrite(&DSize, sizeof(int), 1, Output);
	for(;;){
		printf("Enter number of year from born\n");
		Entr(&str, &DSize);
		if(*year < DSize)    *year=DSize;
		if(IsValueInt(str, &DSize) && DSize>1900 && DSize<2017)		break;
		printf("Sorry try again\n");
	}
	Head->year=DSize;
	fwrite(&DSize, sizeof(int), 1, Output);
	printf("Enter you gender\n");
	scanf("%c",&c);
	while(c!='m' && c!='f')
	{
		if(c!=10)
			printf("Incorrect symbol. Please try again:\n");
		scanf("%c",&c);
	}
	Head->gender=c;
	fwrite(&c, sizeof(char), 1, Output);
	fclose(Output);
	DSize=0;
}

void Download(struct People **Head, int *group, int *lname, int *name, int *pname,
	int *day, int *month, int *year)
{
	FILE *Output;
	struct People *Current;
	int DSize=0;
	int CountPeople, temporary;
	int i;
	struct Size *Count;

	*group=*lname=*name=*pname=0;
	*day=*month=*year=0;
```

```
	Output = fopen("data.bin", "rb");
	if(Output == 0)
	{
		printf("Error opening file");
    getchar();
		return;
	}
	Count=(struct Size*)calloc(1,sizeof(struct Size));
	fread(&CountPeople, sizeof(int),1,Output);
	Current=(struct People*)malloc(sizeof(struct People));
	Current->name=NULL;
	Current->surname=NULL;
	Current->patronymic=NULL;
	Current->left=NULL;
	Current->right=NULL;
	Current->izquierda=NULL;
	Current->derecho=NULL;

	*Head=Current;
	for(i=0; i<CountPeople; i++)
	{
		DSize=fread(&(Current->group),sizeof(int),1,Output);
		if(DSize > *group) *group=DSize;

		fread(&(DSize),sizeof(int),1,Output);
		Current->surname=new char[DSize+1];
		fread(Current->surname,sizeof(char),DSize,Output);
		*(Current->surname+DSize)=0;
		if(DSize > *lname) *lname=DSize;

		fread(&(DSize),sizeof(int),1,Output);
		Current->name=new char[DSize+1];
		fread(Current->name,sizeof(char),DSize,Output);
		*(Current->name+DSize)=0;
		if(DSize > *name)  *name=DSize;

		fread(&(DSize),sizeof(int),1,Output);
		Current->patronymic=new char[DSize+1];
		fread(Current->patronymic,sizeof(char),DSize,Output);
		*(Current->patronymic+DSize)=0;
		if(DSize > *pname)		*pname=DSize;

		DSize=fread(&(Current->day),sizeof(int),1,Output);
		if(DSize > *day)    *day=DSize;
		DSize=fread(&(Current->month),sizeof(int),1,Output);
		if(DSize > *month)*month=DSize;
		DSize=fread(&(Current->year),sizeof(int),1,Output);
		if(DSize > *year)   *year=DSize;
		fread(&(Current->gender),sizeof(char),1,Output);

		Add(*Head,Current,0);
		Add(*Head,Current,1);
		Current=(struct People*)malloc(sizeof(struct People));
		Current->name=NULL;
		Current->surname=NULL;
		Current->patronymic=NULL;
		Current->left=NULL;
		Current->right=NULL;

		Current->izquierda=NULL;
		Current->derecho=NULL;

	}
	destr(Current);
	fclose(Output);
}
```

```
void Sorting(struct People **Head, int Operation, int flag, int key, int group, int lname, int name, int pname,
	int day, int month, int year)
{
	int i,j;
	char x1[6]="Group";
	char x2[8]="Surname";
	char x3[5]="Name";
	char x4[11]="Patronymic";
	char x5[4]="Day";
	char x6[6]="Month";
	char x7[5]="Year";
	char x8[7]="Gender";

	if(group < 5)		group=5;
	if(lname < 7)		lname=7;
	if(name < 4)		name=4;
	if(pname < 10)		pname=10;
	if(day < 3)		day=3;
	if(month < 5)		month=5;
	if(year < 4)		year=4;

	j=group+lname+name+pname+day+month+year+22;

	printf("%-*s||%-*s||",group,x1,lname,x2);
	printf("%-*s||%-*s||",name,x3,pname,x4);
	printf("%-*s||%-*s||",day,x5,month,x6);
	printf("%-*s||%-*s||\n",year,x7,5,x8);
	for(i=0; i<j; i++)	printf("-");

	Bypass(*Head, flag, key, Operation-1, group, lname, name, pname, day, month, year);
	//Head=NULL;
}

int main()
{
	char c;
	short ext=1;
	int key=0;
	struct People *Head=NULL;
	struct People *Dump=NULL;
	int group=0, lname=0, name=0, pname=0, day=0, month=0, year=0;
	FILE *Output;	//For filter

	Download(&Head, &group, &lname, &name, &pname, &day, &month, &year);

	while(ext)
	{
		printf("Choose what you need:\n");
		printf("1 Add person\n");
		printf("2 Sort people by FullName\n");
		printf("3 Sort people by date of birth\n");
		printf("4 Sort people from group\n");

		scanf("%c",&c);

		system("cls");

		switch (c)
		{
		case '1':
			{
				Dump=(struct People*)malloc(sizeof(struct People));
				Dump->left=NULL;
				Dump->right=NULL;
				Dump->izquierda=NULL;
				Dump->derecho=NULL;
```

```
                                AddPerson(Dump, &group, &lname, &name, &pname, &day, &month, &year);
                                if(Head==NULL)
                                        Head=Dump;
                                else{
                                        Add(Head,Dump,0);
                                        Add(Head,Dump,1);
                                }
                                break;
                        }
                case '2':
                        {
                                Sorting(&Head, 1,0,0, group, lname, name, pname, day, month, year);
                                break;
                        }
                case '3':
                        {
                                Sorting(&Head, 2,0,0, group, lname, name, pname, day, month, year);
                                break;
                        }
                case '4':
                        {
                                printf("Enter the number of group\n");
                                scanf("%d",&key);
                                printf("\n");

                                Output = fopen("data.bin", "rb");
                                if(Output == 0)
                                {
                                        printf("Error opening file");
                                        getchar();
                                        return 0;
                                }
                                fclose(Output);
                                system("cls");
                                Sorting(&Head, 1,1,key, group, lname, name, pname, day, month, year);
                                break;
                        }
                        default:
                        {
                                printf("\nIncorrect symbol");
                                ext=0;
                        }
                }
                getchar();
                getchar();
                system("cls");
        }
        destr(Head);
}
```