**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Вычислительной техники**

отчет

**по лабораторной работе № 6**

**по дисциплине «Программирование»**

Тема: Структуры. Массивы структур

|  |  |  |
| --- | --- | --- |
| Студент гр. 3312 |  | Мохно Даниил. |
| Преподаватель |  | Аббас Саддам |

Санкт-Петербург

2024

**Цель работы.**

Целью работы является изучение структур и массивов структур в языке Си и получение практических навыков в программировании на этом языке.

**Задание (вариант 6)**

Описать структуру с именем NOTE2, содержащую поля: Name – фамилия и инициалы, TELE – номер телефона, DATE – дата рождения (год, месяц, число).

Написать программу, выполняющую:

* ввод с клавиатуры данных в массив BLOCK2, состоящий из 7 элементов типа NOTE2, записи должны быть упорядочены по первым трем цифрам номера телефона;
* вывод на экран информации о человеке, чья фамилия введена с клавиатуры, если такого нет – выдать сообщение.

**Постановка задачи и описание решения**

Для начала объявим структуру. Структура NOTE2 содержит поля Name, Tele и Date. Поле Name, является структурой NAME и содержит поля surname – строка с фамилией, first\_letter\_name – символ первой буквы имени, first\_letter\_middle\_name – символ первой буквы отчества. Поле Tele, является структурой Tele, содержащей поле number – строка с номером телефона. Поле Date является структурой DATE, содержащей целочисленные поля day, month, year. Для удобства, с помощью оператора объявления типа создадим новые типы для всех вышеперечисленных структур. Для удобства, типы будут иметь такие же названия.

В главной функции объявим массив структур BLOCK2 типа NOTE2. Циклом перебираем наш массив, каждый раз передавая указатель на элемент в функцию заполнения полей структуры типа NOTE2 (fill\_note). Затем, заполнив структуру, мы вызываем функцию сортировки массива по 3-м первым цифрам номера телефона (sort\_note). После чего, снова перебирая массив, передаём каждую структуру из массива в функцию для вывода полей структуры (show\_info). Грубо говоря, выводим наш массив структур на экран. А за тем, вызываем функцию поиска нужной структуры в массиве, по строке с фамилией (search).

Так как поля нашей структуры тоже являются структурами, в функции заполнения полей структуры NOTE2 (fill\_note) мы для каждого поля этой структуры вызываем определённую функцию заполнения полей структуры. Для поля Name ф-ю заполнения имени (fill\_name), для поля Tele ф-ю заполнения телефона (fill\_tele), для поля Date ф-ю заполнения даты (fill\_date).

В ф-ии заполнения имени мы для каждого поля структуры Name (fill\_name) запрашиваем ввод у пользователя. Так как Фамилию нам нужно получить полностью мы вызываем функцию получения фамилии, которая получит от пользователя строку и символ переноса строки \n заменит на символ окончания строки \0, и передаём ей поле surname. Так как остальные два поля – инициалы, мы просто запрашиваем от пользователя по символу на поля first\_letter\_name, first\_letter\_middle\_name.

В ф-ии заполнения телефона мы динамически выделяем память в переменную для хранения строки с номером телефона для того, чтобы при следующем вызове ф-ии, строка с номером не сохранялась в памяти, объявляем флаг для обозначения выполнения условия корректности номера телефона равным 0 и запускаем цикл, который будет выполняться пока флаг равен 0. В цикле ставим флаг равным 1 и запрашиваем номер телефона через ф-ю для получения строки, заменяющую \n на \0. Далее проверяем чтобы первый элемент номера телефона был ‘+’ или цифра. Так как в си символы тоже можно интерпретировать как цифры мы просто проверяем чтобы символы были больше или равны 48(‘0’) но меньше или равны 57 (‘9’). Если это условие выполняется мы перебираем строку циклом до символа окончания строки (\0) проверяем чтобы каждый символ был цифрой. Если какой, то символ меньше 48 или больше 57, то ставим флаг равным 0. Если условие по проверке первого символа не выполнилось, мы ставим флаг равен 0. Далее если флаг равен 1, то мы копируем строку, содержащую номер телефона, в поле number структуры TELE, в ином случае, выводим ошибку. Далее освобождаем память в переменной хранения строки с номером телефона.

В функции для проверки даты объявляем флаг для обозначения выполнения условия корректности даты на равным 0 и запускаем цикл, который будет выполняться пока флаг равен 0. Запрашиваем у пользователя день, месяц и год в формате «день.месяц.год» и передаём их и указатель на флаг в функции проверки даты (check\_correct\_date). Если флаг равен 1 присваиваем день месяц и год соответствующим полям структуры типа DATE, в ином случае выводим сообщение об ошибке.

В функции проверки даты мы проверяем чтобы день месяц был больше 0 и меньше или равен 12 и день был больше 0. Далее если месяц равен 2 (февраль), то проверяется, что день находится в диапазоне от 1 до 29 в случае високосного года (делится на 4) или от 1 до 28 в обычном году. Если месяц меньше 8 и не является февралем, то проверяется четность месяца: если месяц четный, то день должен быть в пределах от 1 до 30, в противном случае от 1 до 31. Если месяц больше или равен 8, то снова проверяется четность месяца и соответственно диапазон дней: для четных месяцев от 1 до 31, для нечетных от 1 до 30. Если эти условия выполняются, то мы устанавливаем значение флага равным 1.

В функции сортировки мы используем сортировку пузырьком. Внешний цикл выполняется для каждого элемента массива, за исключением последнего. Внутренний цикл выполняется для сравнения и обмена элементов в пределах неотсортированной части массива. Внутри вложенного цикла используется условие, основанное на сравнении первых трёх цифр номеров телефонов. Сначала происходит сравнение первых цифр номеров телефонов. Если первые цифры различаются, то выполняется обмен местами элементов массива через ф-ю в которую передаём массив и индекс элемента (swap(note, j)), таким образом, структуры сравниваются по первой цифре номера телефона в порядке возрастания. Если первые цифры одинаковы, происходит дополнительная проверка в следующем уровне: сравниваются вторые цифры номеров телефонов. Если вторые цифры различаются, то выполняется обмен местами элементов массива. Если вторые цифры также совпадают, происходит сравнение третьих цифр номеров телефонов. Если третьи цифры различаются, происходит обмен местами элементов массива. Если третьи цифры также совпадают, обмен не выполняется, так как структуры уже равны по номерам телефонов.

Функция обмена элементов местами сохраняет j-тый элемент массива в типа NOTE2 в переменной типа NOTE2, присваивает на место j-го элемента j+1, а на место j+1 элемент из доп. переменной.

В функции поиска мы выделяем память для строки с именем введённым пользователем. Далее запускаем цикл с пост условием. В нём запрашиваем у пользователя фамилию, через ф-ю получения строки (которая получает строку и заменяет \n на \0). После чего записываем в флаг stop результат ф-ии strcmp, которой параметрами передаём фамилию, введённую пользователем и слово «stop». Если функция вернёт 0, значит строки равны, и пользователь ввёл стоп слово. Если флаг stop не равен 0, то мы присваиваем флагу, указывающему на то, что элемент был найден значение 0, перебираем циклом массив структур NOTE2 и через strcmp сравниваем поле с surname подструктуры name нашей i-ой структуры. Если результат – 0, то добавляем i-ый элемент структуры в k-ый элемент массива найденных элементов и флаг found ставим равным 1. Далее, если элемент найден, то есть флаг равен 1, то мы выводим циклом поля всех найденных элементов, с помощью функции вывода(show\_info), которой передаём структуру типа NAME2. В случае если ни один элемент не найден, мы выводим сообщение. Далее проверяем флаг stop, если он равен 0, то цикл завершается. В конце освобождаем память, выделенную под функцию для хранения фамилии введённой пользователем.

**Описание переменных**

**Функция – int main():**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | i | int | Итератор |
| 2 | BLOCK2 | struct NOTE2 | Массив структур |

**Функция вызова заполнений полей структуры – void fill\_note(NOTE2 \*note):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | i | int | Итератор |
| 2 | \*note | Struct NOTE2 | Указатель на структуру |

**Функция заполнения полей структуры Name – void fill\_name(NAME \*name):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | i | int | Итератор |
| 2 | \*name | struct NAME | Указатель на структуру |

**Функция получения строки void get\_string(char \*str, int mx\_len):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | \*str | char | Указатель на строку, которую надо получить |
| 2 | mx\_len | int | Максимально возможная длинна строки |
| 3 | len | int | Длинна полученной строки |

**Функция заполнения поля структуры TELE – void fill\_tele(TELE \*tele):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | \*tele | struct TELE | Указатель на структуру |
| 2 | \*nubmer | char | Указатель на строку, в которую получаем номер телефона |
| 3 | i | int | Итератор |
| 4 | flag | int | Флаг для обозначения выполнения условия корректности номера телефона |
| 5 | len | int | Длинна полученной строки |

**Функция заполнения полей структуры DATE – void fill\_date(DATE \*date):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | \*date | struct DATE | Указатель на структуру |
| 2 | day | int | День |
| 3 | month | int | Месяц |
| 4 | year | int | Год |
| 5 | flag | int | Флаг для обозначения выполнения условия даты |

**Функция проверки корректности даты – void check\_correct\_date(int day, int month, int year, int \*flag):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 2 | day | int | День |
| 3 | month | int | Месяц |
| 4 | year | int | Год |
| 5 | \*flag | int | Указатель на флаг для обозначения выполнения условия даты |

**Функция сортировки массива структур –void sort\_note(NOTE2 \*note):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | \*note | struct NOTE2 | Указатель на массив структур |
| 2 | i | int | Итератор |
| 3 | j | int | Итератор/индекс элемента массива |

**Функция смены местами двух соседних элементов массива – void swap(NOTE2 \*note, int j):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | \*note | struct NOTE2 | Указатель на структуру |
| 2 | tmp | int | Переменная для временного хранения структуры |
| 3 | j | int | Индекс элемента массива |

**Функция поиска структуры по фамилии – void search(NOTE2 \*arr)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | \*arr | struct NOTE2 | Указатель на массив структур |
| 2 | found | int | Флаг, указывающий на то, что элемент был найден |
| 3 | i | int | Индекс элемента массива |
| 4 | k | int | Индекс элемента массива |
| 5 | stop | int | Флаг, указывающий что было введено слово “stop” |
| 6 | items[] | struct NOTE2 | Массив найденных по фамилии элементов |
| 7 | \*cur\_surname | char | Полученная от пользователя фамилия, по которой ищем структуру |

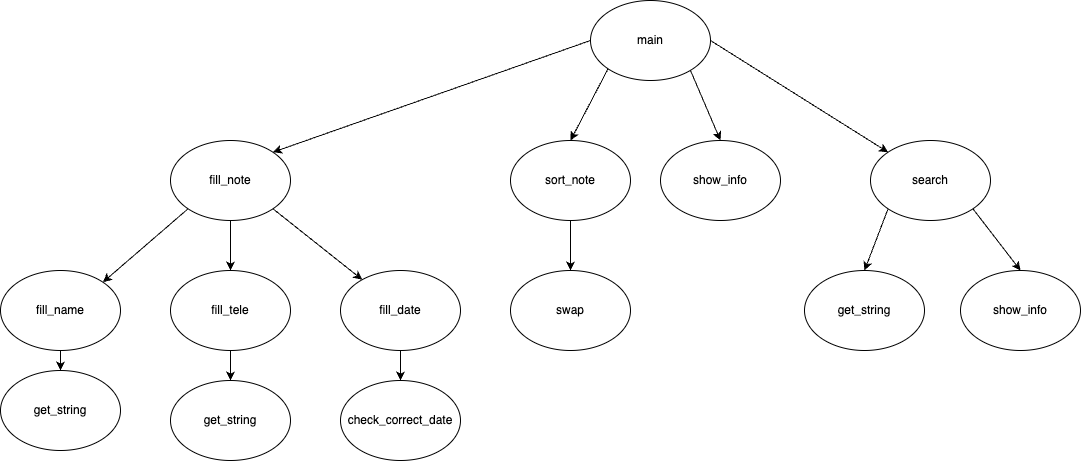
**Функция вывода заголовка таблицы на экран void print\_header()**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | i | int | Итератор |

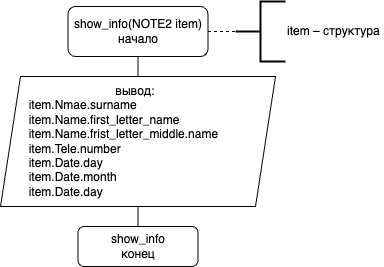
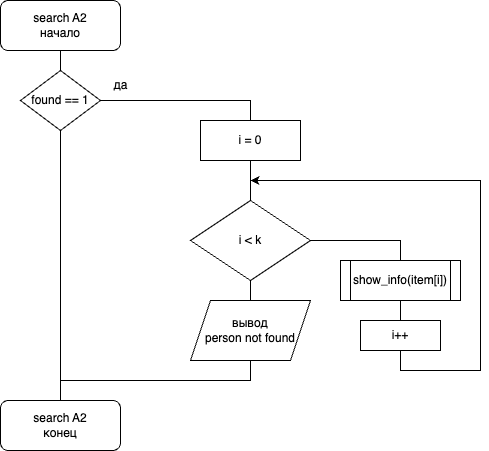
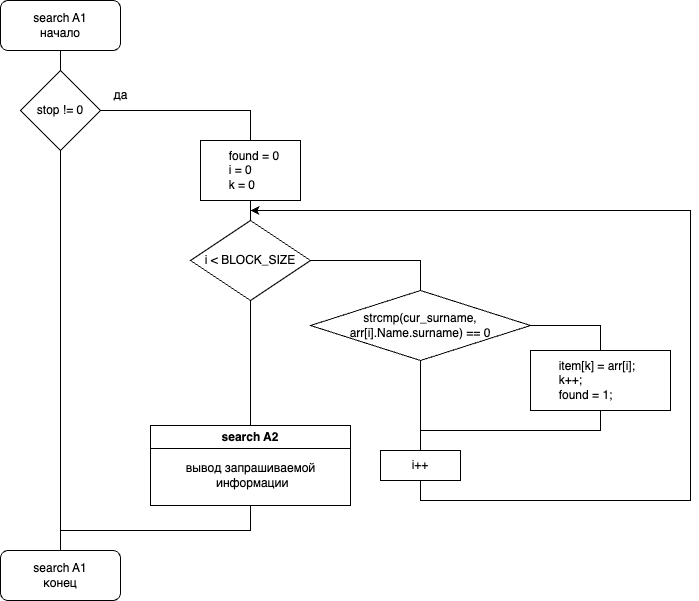
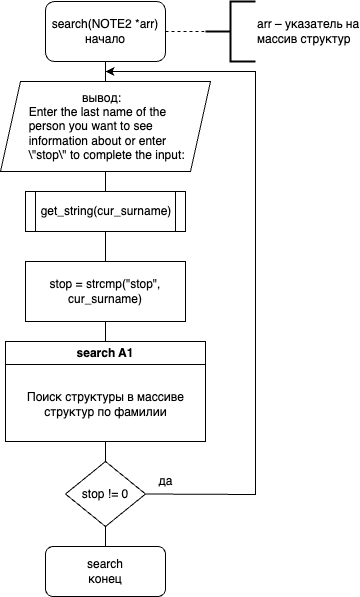
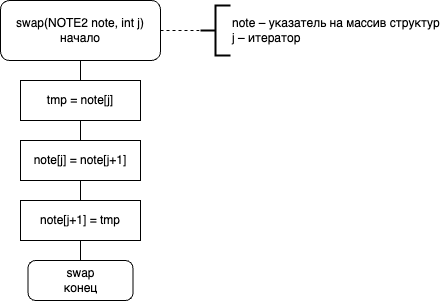
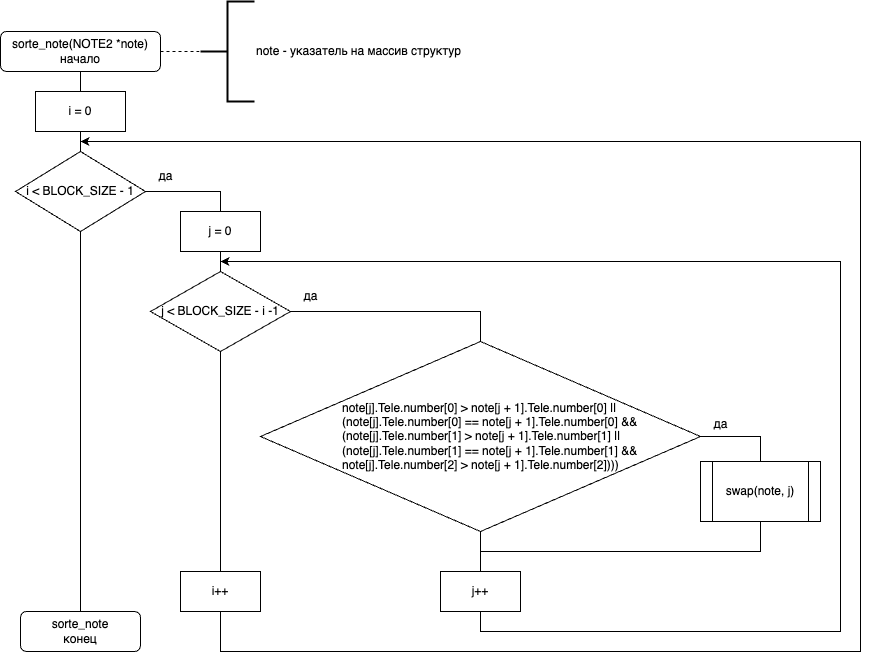
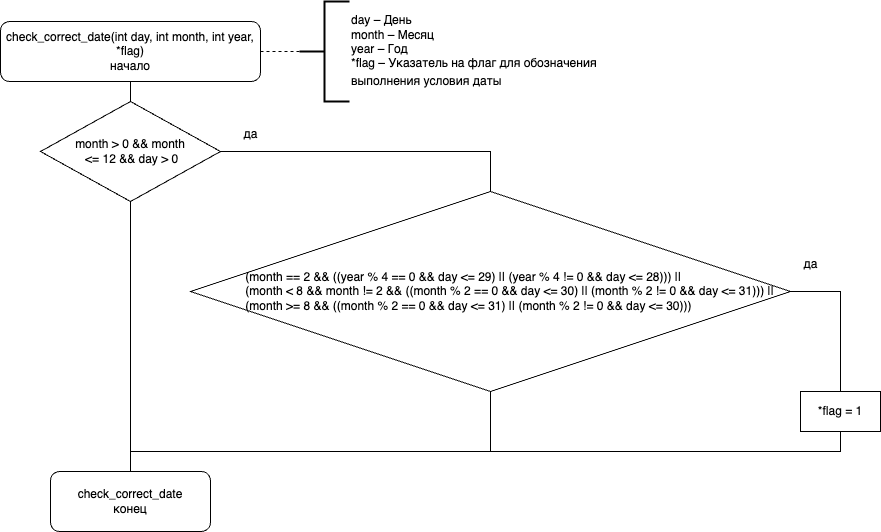
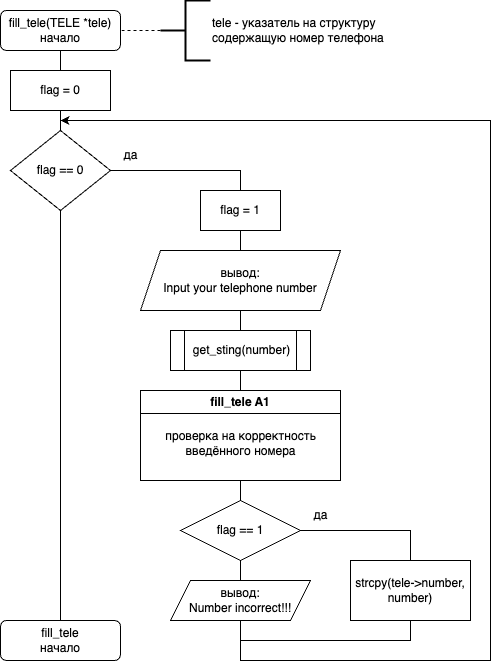
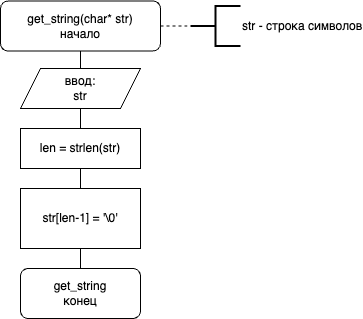
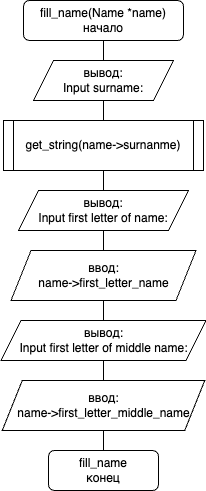
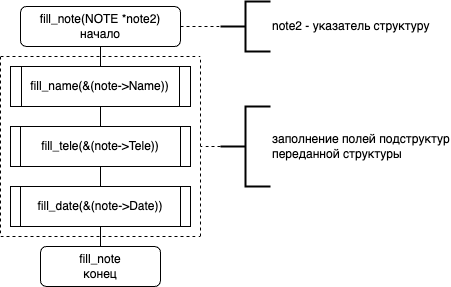
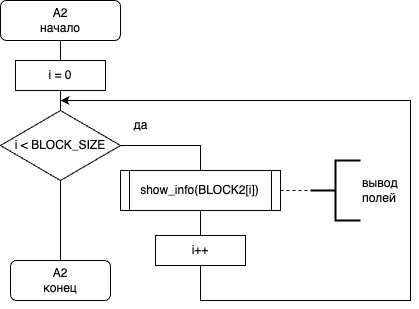
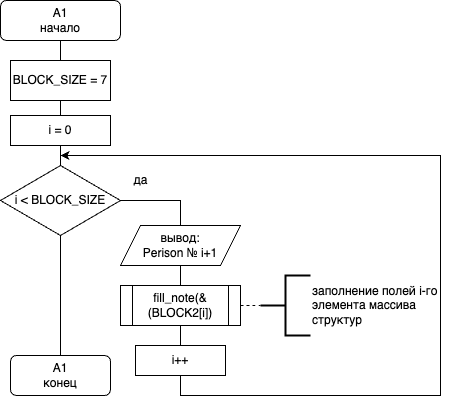
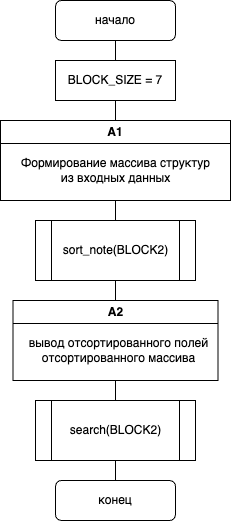
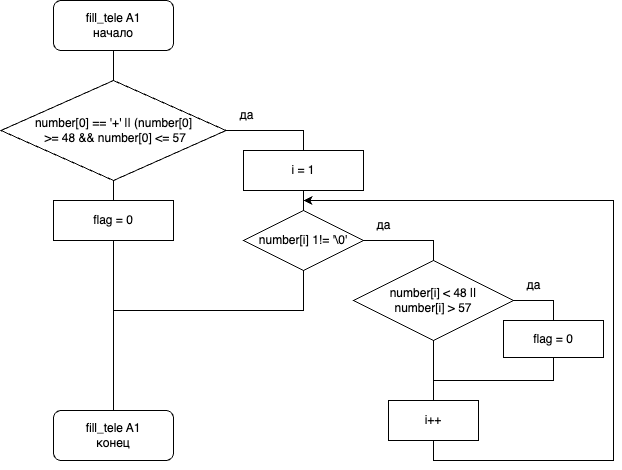
**Функция вывода полей структуры на экран – void show\_info(NOTE2 item)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | item | struct NOTE2 | Структура |
| 2 | i | int | Итератор |

**Структура вызова функций**

****

**Схема алгоритма**

****

**Контрольные примеры**

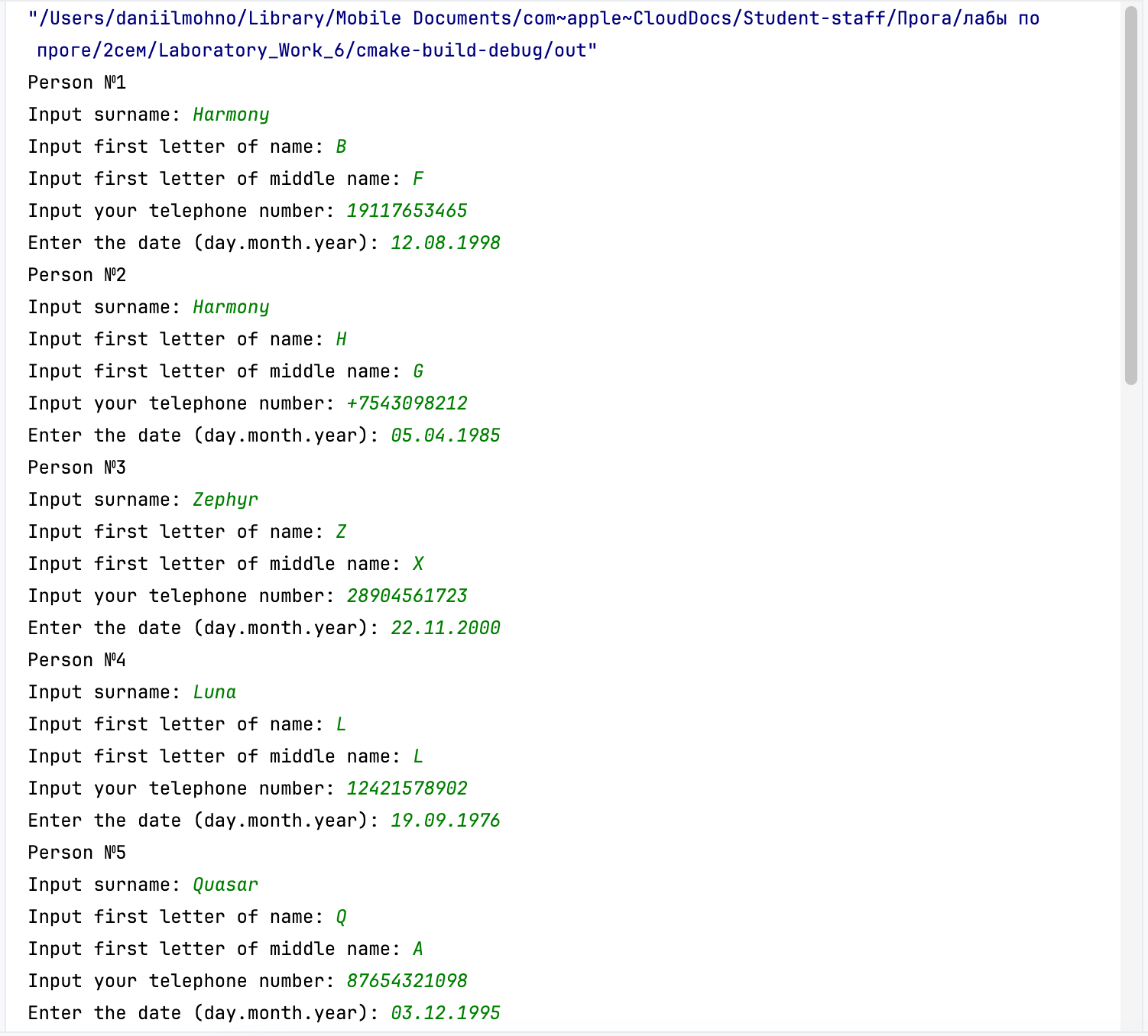
|  |  |  |
| --- | --- | --- |
| № | Входные данные | Выходные данные |
| 1 | Harmony  B  F  19117653465  12.08.1998  Harmony  H  G  +7543098212  05.04.1985  Zephyr  Z  X  28904561723  22.11.2000  Luna  L  L  12421578902  19.09.1976  Quasar  Q  A  87654321098  03.12.1995  Nimbus  N  R  12345678901  07.06.1989  Celestia  C  K  56789012345  30.01.2005  Harmony  Petrov  stop | -------------------------------------------------------  | Surname N. M. | Phone Number | Day.Month.Year |  -------------------------------------------------------  | Harmony H. G. | +7543098212 | 05.04.1985 |  -------------------------------------------------------  | Nimbus N. R. | 12345678901 | 07.06.1989 |  -------------------------------------------------------  | Luna L. L. | 12421578902 | 19.09.1976 |  -------------------------------------------------------  | Harmony B. F. | 19117653465 | 12.08.1998 |  -------------------------------------------------------  | Zephyr Z. X. | 28904561723 | 22.11.2000 |  -------------------------------------------------------  | Celestia C. K. | 56789012345 | 30.01.2005 |  -------------------------------------------------------  | Quasar Q. A. | 87654321098 | 03.12.1995 |  -------------------------------------------------------  -------------------------------------------------------  | Surname N. M. | Phone Number | Day.Month.Year |  -------------------------------------------------------  | Harmony H. G. | +7543098212 | 05.04.1985 |  -------------------------------------------------------  | Harmony B. F. | 19117653465 | 12.08.1998 |  -------------------------------------------------------  Person not found |
| 2 | Aurora  A  M  98765432100  15.03.1992  Solaris  S  P  +12345678987  04.07.1980  Orion  O  F  34567890123  29.02.2003  29.02.2004  Nova  N  E  +876543A1032  +87654321012  31.07.1975  Galaxy  G  H  65432109876  31.08.1996  Stellar  S  S  +23456789012  31.11.1987  02.11.1987  Comet  C  B  78901234567  14.09.2001  Berlin  Orion  stop | Incorrect date!!!  Number incorrect!!!  Incorrect date!!!  -------------------------------------------------------  | Surname N. M. | Phone Number | Day.Month.Year |  -------------------------------------------------------  | Solaris S. P. | +1234567898 | 04.07.1980 |  -------------------------------------------------------  | Stellar S. S. | +2345678901 | 02.11.1987 |  -------------------------------------------------------  | Nova N. E. | +8765432101 | 31.07.1975 |  -------------------------------------------------------  | Orion O. F. | 34567890123 | 29.02.2004 |  -------------------------------------------------------  | Galaxy G. H. | 65432109876 | 31.08.1996 |  -------------------------------------------------------  | Comet C. B. | 78901234567 | 14.09.2001 |  -------------------------------------------------------  | Aurora A. M. | 98765432100 | 15.03.1992 |  -------------------------------------------------------  Person not found  -------------------------------------------------------  | Surname N. M. | Phone Number | Day.Month.Year |  -------------------------------------------------------  | Orion O. F. | 34567890123 | 29.02.2004 |  ------------------------------------------------------- |

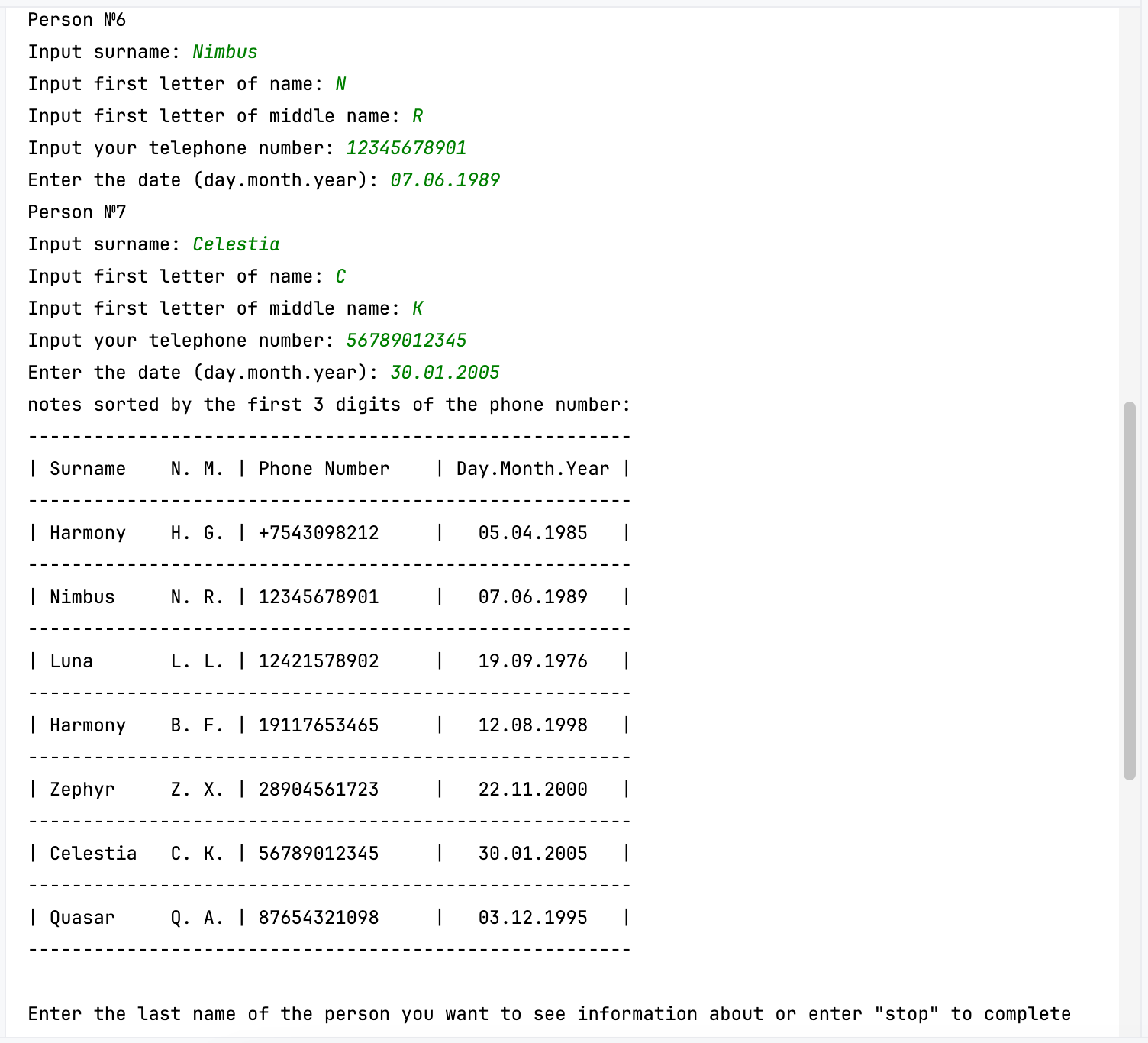
**Текст программы**

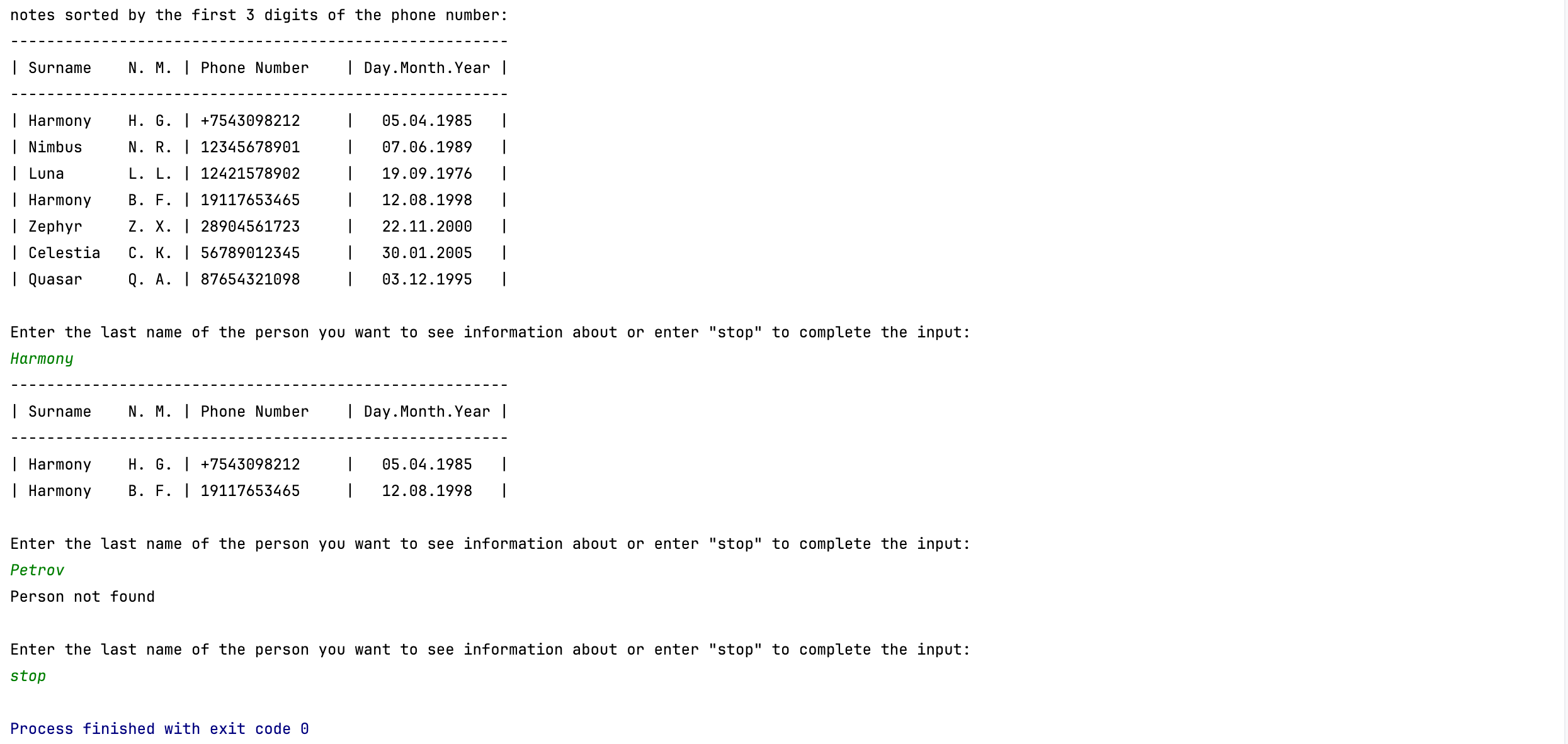
|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #define BLOCK\_SIZE 7  #define MAX\_LEN\_NAME 100  #define TELEPHONE\_LENGTH 13  typedef struct NAME  {  char surname[MAX\_LEN\_NAME];  char first\_letter\_name;  char first\_letter\_middle\_name;  } NAME;  typedef struct TELE  {  char number[TELEPHONE\_LENGTH];  } TELE;  typedef struct DATE  {  int day;  int month;  int year;  } DATE;  typedef struct NOTE2  {  NAME Name;  TELE Tele;  DATE Date;  } NOTE2;  void fill\_note(NOTE2 \*note);  void fill\_name(NAME \*name);  void get\_string(char \*str, int mx\_len);  void fill\_tele(TELE \*tele);  void fill\_date(DATE \*date);  void check\_correct\_date(int day, int month, int year, int \*flag);  void sort\_note(NOTE2 \*note);  void swap(NOTE2 \*note, int j);  void print\_header();  void show\_info(NOTE2 item);  void search(NOTE2 \*arr);  int main()  {  NOTE2 BLOCK2[BLOCK\_SIZE];  int i;  for (i = 0; i < BLOCK\_SIZE; i++)  {  printf("Person №%i\n", i + 1);  fill\_note(&(BLOCK2[i]));  }  sort\_note(BLOCK2);  printf("notes sorted by the first 3 digits of the phone number:\n");  print\_header();  for (i = 0; i < BLOCK\_SIZE; i++)  {  show\_info(BLOCK2[i]);  }  search(BLOCK2);  return 0;  }  void fill\_note(NOTE2 \*note)  {  fill\_name(&(note->Name));  fill\_tele(&(note->Tele));  fill\_date(&(note->Date));  }  void fill\_name(NAME \*name)  {  printf("Input surname: ");  get\_string(name->surname, MAX\_LEN\_NAME);  printf("Input first letter of name: ");  scanf("%c", &(name->first\_letter\_name));  getchar();  printf("Input first letter of middle name: ");  scanf("%c", &(name->first\_letter\_middle\_name));  getchar();  }  void get\_string(char \*str, int mx\_len)  {  int len;  fgets(str, mx\_len, stdin);  len = strlen(str);  str[len - 1] = '\0';  }  void fill\_tele(TELE \*tele)  {  char \*number;  int i;  int flag;  number = malloc(TELEPHONE\_LENGTH \* sizeof(char));  flag = 0;  while (!flag)  {  flag = 1;  printf("Input your telephone number: ");  get\_string(number, TELEPHONE\_LENGTH);  if (number[0] == '+' || (number[0] >= 48 && number[0] <= 57))  {  for (i = 1; number[i] != '\0'; i++)  {  if (number[i] < 48 || number[i] > 57)  flag = 0;  }  }  else  flag = 0;  if (flag)  strcpy(tele->number, number);  else  printf("Number incorrect!!!\n");  }  free(number);  }  void fill\_date(DATE \*date)  {  int day, month, year;  int flag;  flag = 0;  while (!flag)  {  printf("Enter the date (day.month.year): ");  scanf("%d.%d.%d", &day, &month, &year);  check\_correct\_date(day, month, year, &flag);  if (flag)  {  date->day = day;  date->month = month;  date->year = year;  }  else  {  printf("Incorrect date!!!\n");  }  getchar();  }  }  void check\_correct\_date(int day, int month, int year, int \*flag)  {  if (month > 0 && month <= 12 && day > 0)  {  if ((month == 2 && ((year % 4 == 0 && day <= 29) || (year % 4 != 0 && day <= 28))) ||  (month < 8 && month != 2 && ((month % 2 == 0 && day <= 30) || (month % 2 != 0 && day <= 31))) ||  (month >= 8 && ((month % 2 == 0 && day <= 31) || (month % 2 != 0 && day <= 30))))  {  \*flag = 1;  }  }  }  void sort\_note(NOTE2 \*note)  {  int i, j;  for (i = 0; i < BLOCK\_SIZE - 1; i++)  {  for (j = 0; j < BLOCK\_SIZE - i - 1; j++)  {  if (note[j].Tele.number[0] > note[j + 1].Tele.number[0] ||  (note[j].Tele.number[0] == note[j + 1].Tele.number[0] &&  (note[j].Tele.number[1] > note[j + 1].Tele.number[1] ||  (note[j].Tele.number[1] == note[j + 1].Tele.number[1] &&  note[j].Tele.number[2] > note[j + 1].Tele.number[2]))))  {  swap(note, j);  }  }  }  }  void swap(NOTE2 \*note, int j)  {  NOTE2 tmp;  tmp = note[j];  note[j] = note[j + 1];  note[j + 1] = tmp;  }  void print\_header(){  int i;  for (i = 0; i < 55; i++) putchar('-');  putchar('\n');  printf("| %-10s %s. %s. | %-15s | %s.%s.%s |\n", "Surname", "N", "M", "Phone Number", "Day", "Month", "Year");  for (i = 0; i < 55; i++) putchar('-');  putchar('\n');  }  void show\_info(NOTE2 item)  {  int i;  // выводим информацию  printf("| %-10s %c. %c. | %-15s | %02d.%02d.%04d |\n",  item.Name.surname, item.Name.first\_letter\_name,  item.Name.first\_letter\_middle\_name, item.Tele.number,  item.Date.day, item.Date.month, item.Date.year);  }  void search(NOTE2 \*arr)  {  int i, k;  int found;  int stop;  char \*cur\_surname;  NOTE2 items[BLOCK\_SIZE];  cur\_surname = malloc(MAX\_LEN\_NAME \* sizeof(char));  do  {  printf("\nEnter the last name of the person you want to see information about or enter \"stop\" to complete the input:\n");  get\_string(cur\_surname, MAX\_LEN\_NAME);  stop = strcmp("stop", cur\_surname);  if (stop != 0)  {  found = 0;  for (i = 0, k = 0; i < BLOCK\_SIZE; i++)  {  if (strcmp(cur\_surname, arr[i].Name.surname) == 0)  {  items[k] = arr[i];  k++;  found = 1;  }  }  if (found)  {  print\_header();  for (i = 0; i < k; i++) {  show\_info(items[i]);  }  }  else  {  printf("Person not found\n");  }  }  } while (stop != 0);  free(cur\_surname);  } |

**Примеры выполнения программы**

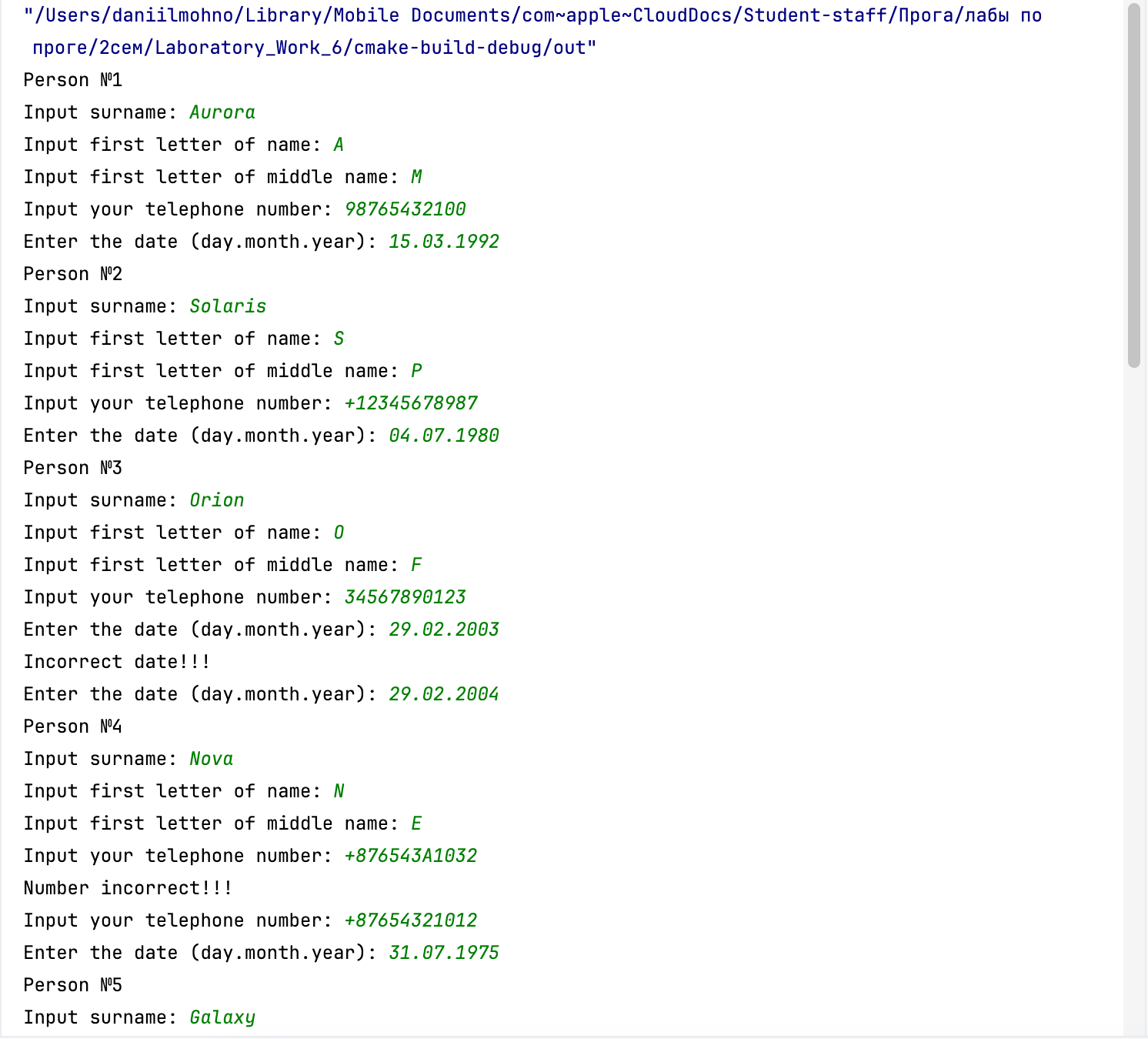
**Пример 1:**

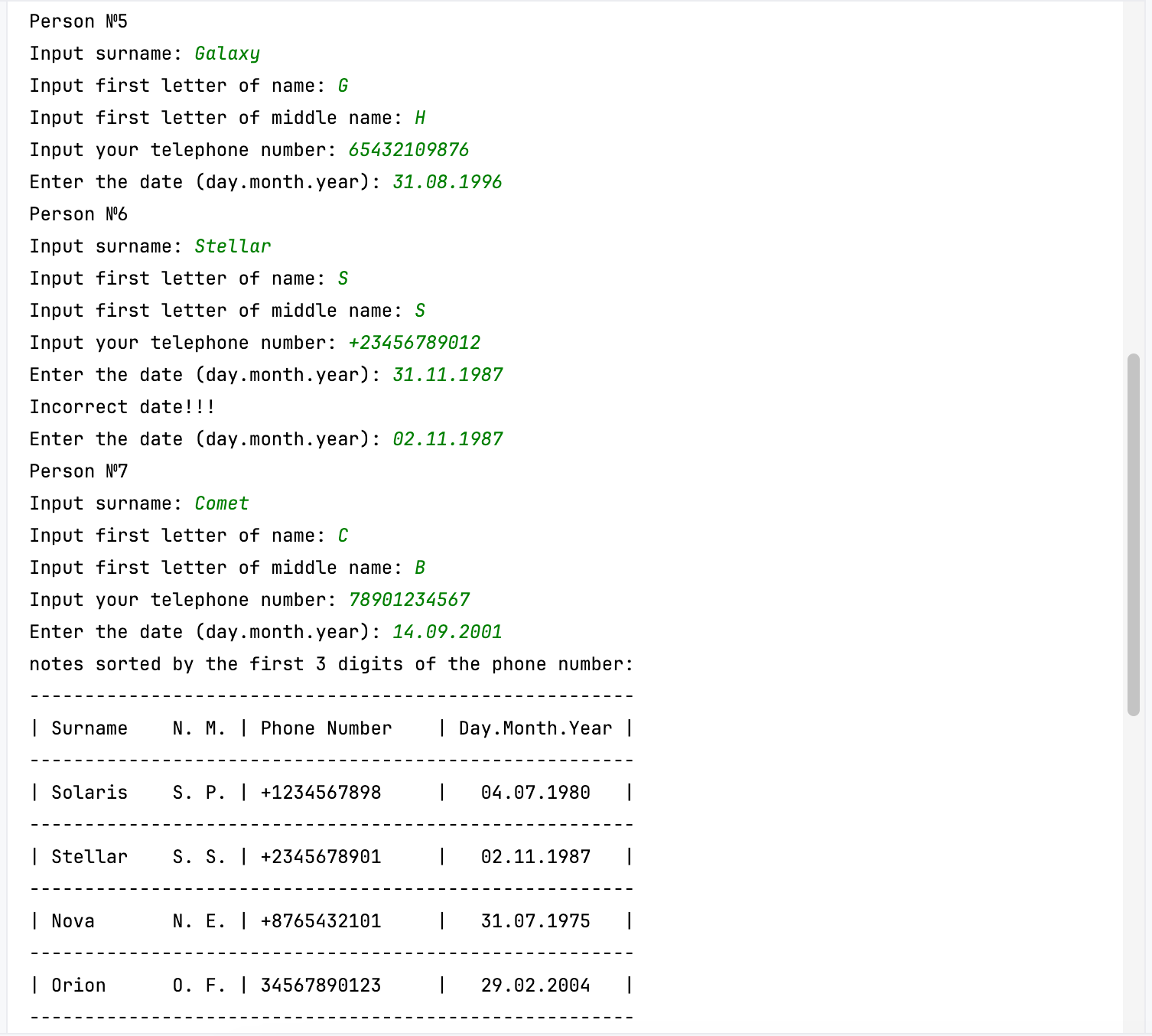
****

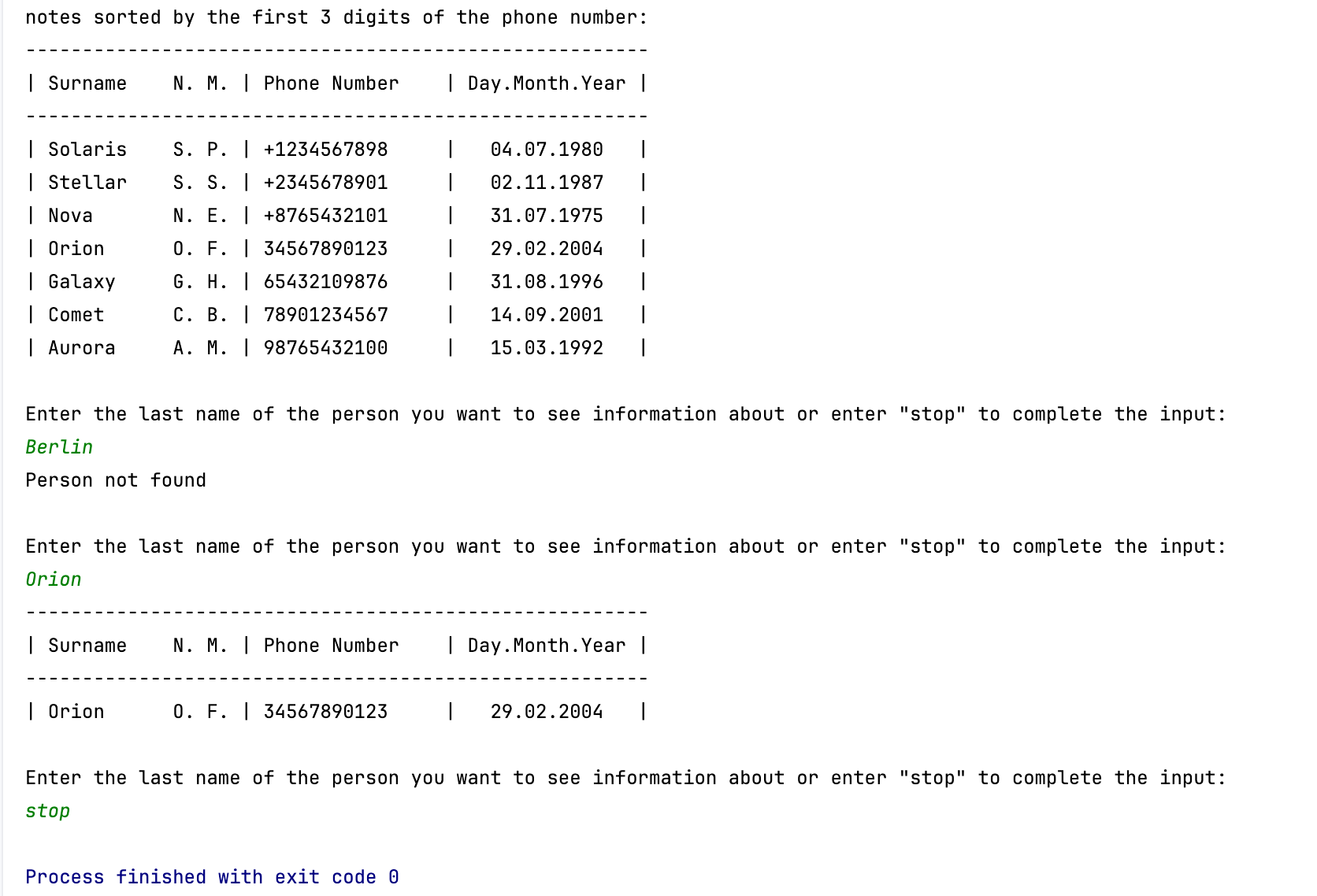
****

****

**Пример 2:**

****

****

****

**Выводы.**

В результате выполнения работы были изучены структуры и массивы структур в языке Си и получены практические навыки в программировании на этом языке.