Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)
Кафедра экономической математики, информатики и статистики (ЭМИС)

СТРУКТУРЫ И УКАЗАТЕЛИ

Отчет по лабораторной работе по дисциплине «Объектно-ориентированное программирование»

Студент группы 549
 Баулин С.К.
«__» _____ 2020 г.
Кандидат физико-математических наук, доцент кафедры ЭМИС
_____ _____ Шельмина Е. А.
оценка «__» _____ 2020 г.

Томск 2020

Лабораторная работа №2

Структуры и указатели

Цель работы

Повторить работу со структурами и указателями в C++.

Краткий теоретический материал

Языки программирования C/C++ поддерживает определяемые пользователем структуры - структурированный тип данных. Он является собранием одного или более объектов (переменных, массивов, указателей, других структур и т.д.), которые для удобства работы с ними сгруппированы под одним именем.

Структуры:

- облегчают написание и понимание программ.
- помогают сгруппировать данные, объединяемые каким-либо общим понятием.
- позволяют группу связанных между собой переменных использовать как множество отдельных элементов, а также как единое целое.

Как и массив, структура представляет собой совокупность данных. но отличается от него тем, что к ее элементам (компонентам) необходимо обращаться по имени и ее элементы могуч быть различного типа. Структуры целесообразно использовать там. где необходимо объединить данные, относящиеся к одному объекту.

Определение структуры состоит из двух шагов:

- объявление шаблона структуры (задание нового типа данных, определенного пользователем);
- определение переменных типа объявленного шаблона.

Объявление шаблонов структур. Общий синтаксис объявления шаблона структуры: struct имя_шаблона

{

тип1 имя_переменной1;

тип1 имя_переменной1;

//другие члены данных;

};

Имена шаблонов должны быть уникальными в пределах их области определения для того, чтобы компилятор мог различать различные типы шаблонов. Задание шаблона осуществляется с помощью ключевого слова struct, за которым следует имя шаблона структуры и список элементов, заключенных в фигурные скобки.

Имена элементов в одном шаблоне также должны быть уникальными. Однако в разных шаблонах можно использовать одинаковые имена элементов.

Задание только шаблона не влечет резервирования памяти компилятором. Шаблон представляет компилятору необходимую информацию об элементах структурной переменной для резервирования места в оперативной памяти и организации доступа к ней при определении структурной переменой и использовании отдельных элементов структурной переменной.

Определение структуры-переменной ничем не отличается от объявления обычной переменной с предопределенным типом. Общий синтаксис:

struct имя_шаблона имя_переменной;

Доступ к компонентам структуры. Доступ к полям осуществляется с помощью оператора «.» при непосредственной работе со структурой или «->» - при использовании указателей на структуру Эти операторы называются селекторами членов класса. Общий синтаксис для доступа к компонентам структуры следующий:

имя_переменной_структуры.член_данных;

имя_указателя->имя_поля;

(*имя_указателя).имя_поля;

При определении структурных переменных можно инициализировать пх поля. Эта возможность подобна инициализации массива и следует тем же правилами:

имя_шаблона имя_переменной_структуры = {значение1, значение2,...};

Компилятор присваивает значение1 первой переменной в структуре, значение2 - второй переменной структуры п т.д., и тут необходимо следовать некоторым правилам:

- присваиваемые значения должны совпадать по типу с соответствующими полями структуры:
- можно объявлять меньшее количество присваиваемых значений, чем количество полей. Компилятор присвоит нули остальными полями структуры:
- список инициализации последовательно присваивает значения полям структуры, вложенных структур и массивов.

Копирование структур-переменных. Язык С(С++) позволяет оператору присваивания копировать

значения одной структуры-переменной в другую переменную. при условии, что обе структуры-переменные относятся к одному7 и тому же типу. Таким образом, единственный оператор может скопировать несколько членов данных, которые включают массивы и вложенные структуры.

Однако следует учитывать, что оператор присваивания выполняет то, что называется поверхностной копией в применении к структурам-переменных. Поверхностная копия представляет собой копирование бит за битом значений полей переменной-источника в соответствующие поля переменной-цели. При этом может возникнуть проблема с такими членами данных, как указатели, поэтому использовать поверхностное копирование структур надо осторожно.

Формулировка заданий с полученными результатами.

Задание 1. Реализовать решение задачи двумя способами: с помощью массива структур и с помощью указателя на структуру.

Ввести массив структур в соответствии с вариантом. Рассортировать массив в алфавитном порядке по первому полю, входящему в структуру. В программе реализовать меню:

1) Ввод массива структур;
2) Сортировка массива структур;
3) Поиск в массиве структур по заданном}' параметру;
4) Изменение заданной структуры;
5) Удаление структуры из массива:
6) Вывод на экран массива структур;
7) Выход.

Структура «Сотрудник»: фамилия, имя, отчество; должность; год рождения; заработная плата.

Код и результат выполненного задания представлены на рисунках 1-16.

```
struct Staffs
{
    char surname[20], name[20], patrname[20];
    char pos[20];
    int year;
    float zarpl;
} sf;
struct Staffs person[100];
struct Staffs temp;
int counter = 0; //глобальная переменная - счетчик
```

Рисунок 1 – Скриншот структуры «Сотрудник»

Вывод

Повторена работа со структурами и указателями

```
int menu()
{
    system("cls");
    int enter;
    cout << "\t\tМеню" << endl;
    cout << "--------------------------------" << endl;
    cout << "1. Добавление сотрудника" << endl;
    cout << "2. Сортировка списка сотрудников по фамилии" << endl;
    cout << "3. Поиск сотрудников по фамилии" << endl;
    cout << "4. Изменение данных о сотруднике" << endl;
    cout << "5. Удаление данных о сотруднике" << endl;
    cout << "6. Вывод на экран список сотрудников" << endl;
    cout << "7. Выход" << endl;
    cin >> enter;
    return enter;
}
```

Рисунок 2 – Скриншот функции меню

```
void input()
{
    system("cls");
    if (counter < 100)
    {
        cout << "Сотрудник №" << counter + 1 << endl;
        cout << "Введите фамилию сотрудника: ";
        cin >> person[counter].surname;
        cout << "Введите имя сотрудника: ";
        cin >> person[counter].name;
        cout << "Введите отчество сотрудника: ";
        cin >> person[counter].patrname;
        cout << "Введите должность сотрудника: ";
        cin >> person[counter].pos;
        cout << "Введите год рождения сотрудника: ";
        cin >> person[counter].year;
        cout << "Введите зарплату сотрудника: ";
        cin >> person[counter].zarpl;
        counter++;
    }
    else
        cout << "Добавлено максимальное кол-во сотрудников!" << endl;
    system("pause");
}
```

Рисунок 3 – Скриншот функции ввода

```
void sort()
{
    system("cls");
    for (int i = 0; i < counter - 1; i++)
    {
        for (int j = i + 1; j < counter; j++)
        {
            if (strcmp(person[i].surname, person[j].surname) > 0)
            {
                temp = person[i];
                person[i] = person[j];
                person[j] = temp;
            }
        }
    }
    cout << "Сортировка прошла успешно!\n";
    system("pause");
}
```

Рисунок 4 – Скриншот функции сортировки

```
void find()
{
    system("cls");
    char fs[20];
    bool check = true;
    if (!counter)
    {
        cout << "Список пуст!" << endl;
        system("pause");
        return;
    }
    cout << "Введите фамилию сотрудника: ";
    cin >> fs;

    for (int i = 0; i < counter; i++)
    {
        if (strcmp((*(person + i)).surname, fs) == 0)
        {
            cout << i + 1 << ". "
                 << (*(person + i)).surname << " "
                 << (*(person + i)).name << " "
                 << (*(person + i)).patrname << " "
                 << "Должность: " << (*(person + i)).pos << " "
                 << (*(person + i)).year << " г. "
                 << "З/п: " << (*(person + i)).zarpl << endl;
            check = false;
        }
    }
    if (check)
    {
        cout << "Ничего не найдено!" << endl;
    }
    system("pause");
}
```

Рисунок 5 – Скриншот функции поиска

```
void change()
{
    system("cls");
    int num;
    int check;
    cout << "Введите номер сотрудника: ";
    cin >> num;
    do
    {
        system("cls");
        cout << person[num - 1].surname << " " << person[num - 1].name << " " << person[num - 1].patrname << " "
             << "Должность: " << person[num - 1].pos << " " << person[num - 1].year << "г. "
             << "З/п: " << person[num - 1].zarpl << " руб."
             << endl;
        cout << "-----------------------------------------" << endl;
        cout << "\t\tИзменить" << endl;
        cout << "-----------------------------------------" << endl;
        cout << "1. Фамилию сотрудника" << endl;
        cout << "2. Имя сотрудника" << endl;
        cout << "3. Отчество сотрудника" << endl;
        cout << "4. Должность сотрудника" << endl;
        cout << "5. Год рождения сотрудника" << endl;
        cout << "6. Зарплату сотрудника" << endl;
        cout << "7. Выход" << endl;
        cin >> check;
        switch (check)
        {
        case 1:
            cout << "Введите фамилию сотрудника: "; cin >> person[num - 1].surname;
            break;
        case 2:
            cout << "Введите имя сотрудника: "; cin >> person[num - 1].name;
            break;
        case 3:
            cout << "Введите отчество сотрудника: "; cin >> person[num - 1].patrname;
            break;
        case 4:
            cout << "Введите должность сотрудника: "; cin >> person[num - 1].pos;
            break;
        case 5:
            cout << "Введите год рождения сотрудника: "; cin >> person[num - 1].year;
            break;
        case 6:
            cout << "Введите зарплату сотрудника: "; cin >> person[num - 1].zarpl;
            break;
        case 7:
            system("pause");
            return;
        default:
            cout << "Не верный запрос.\n";
        }
    } while (true);
    system("pause");
}
```

Рисунок 6 – Скриншот функции изменения данных

```
void del()
{
    system("cls");
    int num;
    cout << "Введите номер сотрудника, данные о котором нужно удалить: ";
    cin >> num;
    for (int i = (num - 1); i < counter; i++)
    {
        person[i] = person[i + 1];
    }
    counter--;
    cout << "Данные успешно удалены!\nСотрудников: " << counter;
}
```

Рисунок 7 – Скриншот функции удаления

```
void output()
{
    system("cls");
    if (counter == 0)
    {
        cout << "Список сотрудников пуст!" << endl;
        system("pause");
        return;
    }
    cout << "Список сотрудников: " << endl;
    for (int i = 0; i < counter; i++)
    {
        cout << i + 1 << ". "
             << person[i].surname << " "
             << person[i].name << " "
             << person[i].patrname << " "
             << "Должность: " << person[i].pos << " "
             << person[i].year << "г. "
             << "З/п: " << person[i].zarpl << " руб." << endl;
    }
    system("pause");
}
```

Рисунок 8 – Скриншот функции вывода

```cpp
int main()
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    setlocale(LC_ALL, "Russian");
    while (true)
    {
        switch (menu())
        {
        case 1:
            input();
            break;
        case 2:
            sort();
            break;
        case 3:
            find();
            break;
        case 4:
            change();
            break;
        case 5:
            del();
            break;
        case 6:
            output();
            break;
        case 7:
            return 0;
        default:
            cout << "Не верный запрос.\n";
        }
    }

    return 0;
}
```

Рисунок 9 – Скриншот функции main

```
                Меню
------------------------------
1. Добавление сотрудника
2. Сортировка списка сотрудников по фамилии
3. Поиск сотрудников по фамилии
4. Изменение данных о сотруднике
5. Удаление данных о сотруднике
6. Вывод на экран список сотрудников
7. Выход
```

Рисунок 10 – Скриншот результата работы функции меню

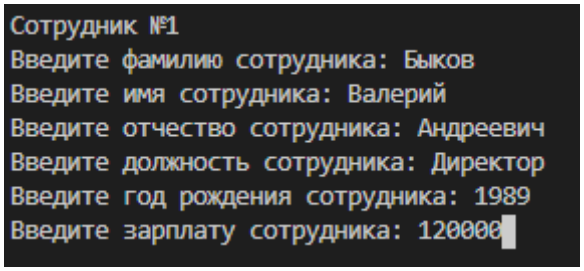

Рисунок 11 – Скриншот результата работы функции ввода

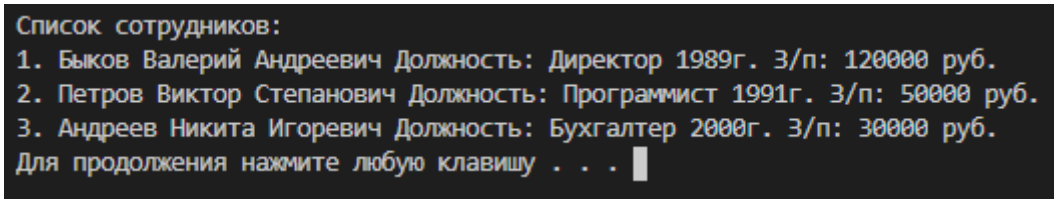

Рисунок 12 – Скриншот результата работы функции вывода

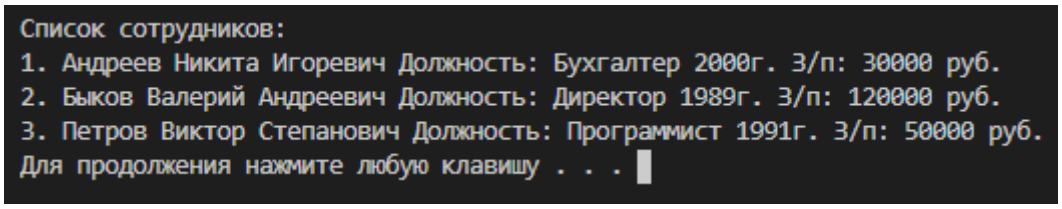

Рисунок 13 – Скриншот результата работы функции сортировки

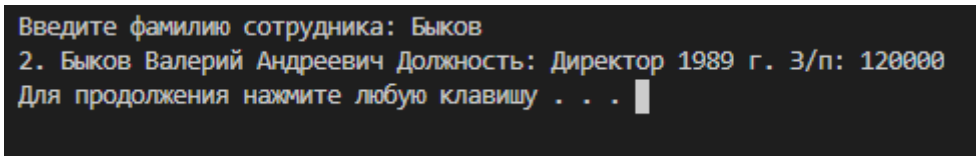

Рисунок 14 – Скриншот результата работы функции поиска

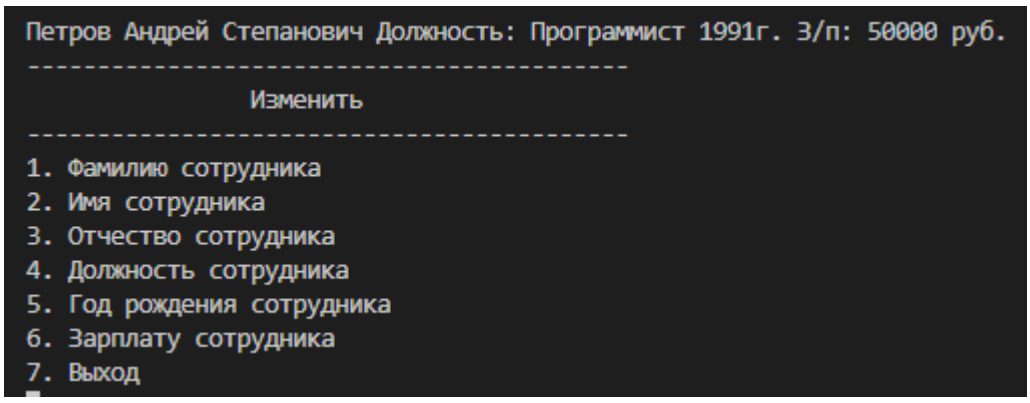

Рисунок 15 – Скриншот результата работы функции изменения

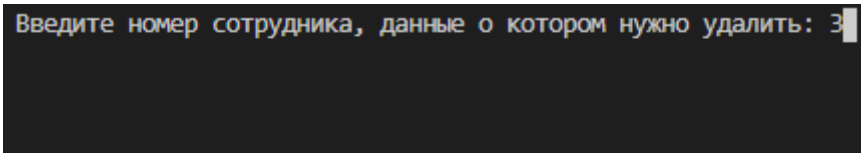

Рисунок 16 – Скриншот результата работы функции удаления