Структура объединяет разные данные разных типов в произвольном количестве. Поэтому все задачи, которые требуют логического объединения каких-то данных, могут и должны использовать структурный тип. Рассмотрим пример использования структур в некоторой модельной задаче, то есть не имеющей особенного практического смысла, но позволяющей продемонстрировать достоинства структурного типа, его возможности, и механизмы.

Сформулируем задачу. Пусть структура объединяет некую информацию об одном абстрактном объекте (человеке): имя, фамилия, и какие-то данные числового характера, например, возраст, рост, средняя зарплата и т.д., назовем их показателями. Пусть имеются сведения о некотором множестве объектов, то есть о некотором сообществе людей, например, студентов какого-либо факультета, сотрудников некоторой организации, случайной статистической выборки.

Объект называется *среднестатистическим*, если на нем достигается минимум модуля разности среднего арифметического значений его показателей со средним арифметическим всей группы объектов. Аналогично определяется уникальный объект (на нем достигается максимум).

Объект может быть назван *среднестатистическим по k-тому параметру*, (уникальным по k-му параметру), тогда рассматриваются данные о значении k-го показателя этого объекта в сравнении со средним значением k-го показателя по всей совокупности объектов.

Выясним, кто в группе объектов является:

а) среднестатистическим,

б) среднестатистическим по отдельным показателям,

в) просто поставим задачу найти объект в группе по ключевому значению.

Для описания одного объекта используется структура. Попытаемся отразить в примере возможности этого типа данных. В качестве полей структуры используем простые типы данных, массив и строки, причем одна строка будет статической, а вторая динамической. Для хранения имени используем статическую строку, для хранения фамилии динамическую, для хранения показателей используем массив (ограничимся пятью показателями). Информация о среднем арифметическом показателей каждого объекта может быть вычислена при получении объектом данных. Ее можно сохранить, для этого введем в структуру поля "Сумма показателей" и "Среднее арифметическое показателей", это данные целого и вещественного типов.

Поскольку в группе объектов несколько данных, следует объединить их в массив, где один элемент массива описывает один объект.

Выполним функциональную декомпозицию задачи, то есть, определим, какие алгоритмы обработки понадобятся для ее решения.

Сначала подумаем, откуда взять данные. Например, их можно ввести.

Для работы с одним объектом пригодятся функции ввода данных об одном объекте и вывода данных об одном объекте. Имя формального параметра пусть будет Man, это полные данные об одном объекте.

Чтобы иметь возможность визуально сравнить данные, требуется вывод общей информации, наверное, в виде таблицы.

Для поиска среднего арифметического по группе объектов и для поиска среднего по какому-то показателю требуются функции, которые возвращают значение среднего по всей группе объектов и по одному показателю. Эти функции похожи внешне и по смыслу, но первая будет работать со средним данным всего объекта, а вторая, получив в качестве параметра номер показателя, будет работать только с этим элементом массива показателей.

Функция поиска может вернуть номер элемента в массиве структур или указатель на объект. Покажем это на примере функций поиска. Функция поиска среднестатистического объекта вернет его номер в массиве структур. Имея функцию вывода, мы легко выведем его данные на экран с использованием операции разыменования []. Функция поиска среднего по k-тому показателю вернет указатель на структуру. Имея функцию вывода, мы легко выведем его данные на экран с использованием операции разыменования \*.

Все эти функции требуют полные данные обо всей группе объектов. Формальный параметр, используемый в их описании, тоже имеет имя Man, но теперь он имеет совсем другую смысловую нагрузку, обозначая имя всей совокупности данных обо всех объектах, то есть о массиве объектов.

Поиск по ключевому значению, например, по фамилии, не внесет ничего нового, просто добавим его, чтобы лишний раз показать механизмы работы со строками символов.

#include <stdio.h>

#include <conio.h>

#include <io.h>

#include <math.h>

#include <string.h>

// Описание абстрактного типа «Объект» с именем Person использует инструкцию

// typedef, потому что объявлений объектов указанного типа будет несколько.

typedef struct

{

char Name [10]; // Поле «Имя», статический символьный массив.

// Символов в имени не более 10-ти.

char \*Surname; // Поле «Фамилия»,

// динамический символьный массив.

// Для нее должна быть выделена динамическая

// память перед присваиванием значения этой

// переменной.

int Data [5]; // Поле «Показатели», статический массив данных.

int Sum; // Поле «Сумма» накапливает и сохраняет

// сумму баллов.

float Var; // Поле «Среднее» вычисляет и сохраняет

// средний показатель.

} Person; // Имя типа данного "Person"

// Функция ввода данных об одном объекте возвращает объект.

void In (Person & Man) // Одно данное типа Person возвращается по ссылке

{ // через параметр.

puts ("Введите имя: ");

scanf ("%s", &Man.Name);

puts ("Введите фамилию: ");

Man.Surname = new char [10]; // Обязательно выделить память.

// Число символов в фамилии не более 10-ти.

scanf ("%s", Man.Surname); // & при вводе значения указателя не пишем.

puts ("Введите данные: ");

Man.Sum = 0; // Будет накоплена сумма показателей при вводе.

for (int i = 0; i < 5; i ++)

{

scanf ("%4d", &Man.Data [i]);

Man.Sum += Man.Data[i];

}

Man.Var = (float) Man.Sum / 5.; // Вычислен средний показатель.

}

// Функция вывода полных данных об одном объекте.

void Out (Person Man) // Передается данное типа Person

{

printf ("%s ", Man.Name);

printf ("%s ", Man.Surname);

for (int i = 0; i < 5; i ++)

printf ("%4d",Man.Data[i]); // Показатели выводятся одной строкой.

printf ("\nСумма баллов: %d ", Man.Sum);

printf ("Среднее: %6.2f\n", Man.Var);

}

// Функция вывода данных обо всех объектах в форме таблицы.

void Out\_All (Person Man [], int n) // Передается n данных типа Person

{

printf ("====================================================\n");

printf ("ИМЯ \t ФАМИЛИЯ \t ДАННЫЕ \t CУММА СРЕДНЕЕ\n");

printf ("====================================================\n");

for (int k = 0;k < n; k++)

{

printf ("% –10s ", Man[k].Name);

printf ("% –10s ", Man[k].Surname);

for (int i = 0; i < 5; i++)

printf ("%4d", Man[k].Data[i]);

printf("%4d ",Man[k].Sum);

printf("%6.2f \n",Man[k].Var);

}

printf ("====================================================\n");

}

// Функция вычисляет средний показатель по всем объектам, суммируя данные

// полей Var по всему массиву Man.

float Sred (Person Man [], int n) // Передается n данных типа Person

{

float Var = 0; // Переменная Var не имеет отношения к полю

// структуры с именем Var.

for (int i = 0; i < n; i++)

Var += Man[i].Var;

return Var / (float) n; // Среднее по всем средним.

}

// Функция вычисляет средний показатель по k-тому параметру.

// Номер показателя (от 0 до 5), это один из параметров функции.

float Sred\_k (Person Man [], int n, int k) // Передается n данных типа Person

{ // и номер показателя k.

float Var\_k = 0; // Var\_k – средний по одному показателю.

for (int i = 0; i < n; i++)

Var\_k += (float) Man[i].Data[k]; // Складывает один k-тый показатель по

// всем объектам.

return Var\_k / (float) n; // Среднее по k-тому показателю.

}

// Функция определяет номер среднестатистического объекта в массиве.

int Sred\_Stat (Person Man [], int n) // Передается n данных типа Person

{

float Var\_All = Sred (Man,n); // Средний показатель по всем объектам.

float Var\_min = fabs(Var\_All-Man[0].Var);

int Nom = 0; // Наименьший имеет номер 0.

for (int i = 0; i < n; i++)

if ( fabs (Var\_All – Man[i].Var) < Var\_min)

Nom = i; // Запоминаем номер самого среднего.

return Nom;

}

// Функция определяет объект, среднестатистический по k-тому показателю,

// и возвращает указатель на найденный объект.

Person \* Sred\_Stat\_k (Person Man [], int n, int k) // Передается n данных типа Person

{ // и номер показателя.

float Var\_k = Sred\_k (Man, n, k); // Находим среднее значение k-того

// показателя по всем объектам.

float Var\_min = fabs (Var\_k – Man[0].Data[k]);

Person \* Nom = Man; // Указатель Nom запоминает адрес

// объекта.

for (int i = 0; i < n; i++)

if ( fabs (Var\_k – Man[i].Data[k]) < Var\_min)

Nom = Man + i; // Запоминаем адрес самого среднего.

return Nom;

}

// Функция поиска объекта по фамилии. Передается n данных типа Person, и

// Who – строка, содержащая фамилию искомого объекта. Выполняется прямой

// поиск по всей группе объектов путем последовательного сравнения искомой

// строки с полем Surname каждого элемента массива Man.

// Функция возвращает указатель на найденный объект или NULL.

Person \* Found\_Fam (char \*Who, Person Man [], int n)

{

for (int i = 0; i < n; i++)

if ( strcmp (Who ,Man[i].Surname) == 0) // ==0, значит, найдено.

return Man+i; // Возвращает адрес.

return NULL; // Возвращает NULL, если поиск неудачен.

}

// Продемонстрируем текст программы, которая объявляет данные и

// осуществляет управление ими путем вызова функций обработки данных.

void main (void)

{

int n;

// Объявлен массив структур, то есть данных типа Person.

Person All\_Person [20];

printf ("Введите количество\n"); // Реальное количество данных n.

scanf ("%d", &n);

// Ввод данных выполняется в цикле вызова функции In ввода одного объекта.

for (int i = 0; i < n; i++)

{

In (All\_Person[i]);

// Out (All\_Person[i]);

}

// Вывод на экран полной таблицы.

Out\_All(All\_Person,n);

// Для поиска номера среднестатистического объекта вызывается функция Sred\_Stat.

// Ей передается весь массив структур.

int Found = Sred\_Stat (All\_Person, n);

printf ("Самый средний имеет номер: %d\n", Found);

printf ("Его данные: \n");

Out (All\_Person [Found]);

// Снова выведем таблицу на экран.

Out\_All(All\_Person,n);

// Для поиска значения объекта, среднестатистического по k-му показателю

// вызывается функция Sred\_Stat\_k. Ей передается весь массив структур.

// Она возвращает указатель на объект.

Person \*Found\_k;

for (int k = 0;k < 5;k ++)

{

printf ("Средний по %d -му показателю\n", k);

Found\_k = Sred\_Stat\_k (All\_Person, n, k);

Out (\*Found\_k);

}

// Поиск объекта по фамилии.

char Surname [10]; // Тоже фамилия, но не та, что в Person.

puts ("Введите строку для поиска\n");

scanf ("%s", Surname); //

Person \* Who; // Функция возвращает указатель.

Who = Found\_Fam (Surname, All\_Person, n);

if (Who != NULL) // Проверяем, найден объект или нет.

Out (\*Who);

else

puts("Такого нет.\n");

}