

НГТУ
НЭТИ

Факультет прикладной
математики
и информатики

Ю. В. ТРАКИМУС
В. П. ХИЦЕНКО

ПРАКТИКУМ ПО ОСНОВАМ ПРОГРАММИРОВАНИЯ

МАССИВЫ
СТРОКИ
ПОДПРОГРАММЫ

НОВОСИБИРСК
2021

УДК 004.42(075.8)
Т652

Рецензенты:
д-р техн. наук, профессор кафедры ПМт *М. Э. Рояк*
канд. техн. наук, доцент кафедры ТПИ *Е. А. Хайленко*

Работа подготовлена на кафедре прикладной математики НГТУ

Тракимус Ю. В.

Т652 Практикум по основам программирования. Массивы. Строки. Подпрограммы: учебное пособие / Ю. В. Тракимус, В. П. Хиценко. – Новосибирск: Изд-во НГТУ, 2021. – 96 с.

ISBN 978-5-7782-4596-9

Настоящее учебное пособие посвящено особенностям использования сложных регулярных типов данных (массивов и символьных строк) и применению подпрограмм в языке программирования высокого уровня С. Пособие содержит примеры решения задач, практические задания и может быть рекомендовано для подготовки к занятиям по основам программирования и при самостоятельном изучении курса. Предназначено для студентов I курса факультета прикладной математики и информатики.

Тракимус Юрий Викторович, Хиценко Валентина Павловна

ПРАКТИКУМ ПО ОСНОВАМ ПРОГРАММИРОВАНИЯ
МАССИВЫ. СТРОКИ. ПОДПРОГРАММЫ

Учебное пособие

В авторской редакции

Выпускающий редактор *И. П. Брованова*
Дизайн обложки *А. В. Ладыжская*
Компьютерная верстка *Л. В. Веселовская*

Налоговая льгота – Общероссийский классификатор продукции
Издание соответствует коду 95 3000 ОК 005-93 (ОКП)

Подписано в печать 27.12.2021. Формат 60 × 84 1/16. Бумага офсетная.
Тираж 50 экз. Уч.-изд. л. 5,58. Печ. л. 6,0. Изд. № 277. Заказ № 62. Цена договорная

Отпечатано в типографии
Новосибирского государственного технического университета
630073, г. Новосибирск, пр. К. Маркса, 20

УДК 004.42(075.8)

ISBN 978-5-7782-4596-9

© Тракимус Ю. В., Хиценко В. П., 2021
© Новосибирский государственный
технический университет, 2021

ОГЛАВЛЕНИЕ

Глава 1 Сложные типы. Массивы 5
Понятие сложного типа 5
Тип массив 6
Указатели и массивы 9
Применение массивов при решении задач 10
Примеры 13
Практическое задание «Массивы» 18
Задачи для самостоятельной работы 22
Контрольные вопросы 29

Глава 2 Обработка текстовой информации 30
Понятие обработки текста 30
Строковый тип 30
Представление текста в оперативной памяти 32
Примеры 34
Практическое задание «Строки» 42
Задачи для самостоятельной работы 44
Контрольные вопросы 49

Глава 3 Подпрограммы 50
Декомпозиция задачи 50
Понятие подпрограммы 50
Функции языка С 52
Функции без параметров 53
Функции с параметрами 54
Функции с параметрами типа массив 58
Прототип функции 62
Рекурсивные подпрограммы 63

Объекты программы и их взаимодействие 66
Примеры 68
Практическое задание «Подпрограммы» 78
Задачи для самостоятельной работы 83
Контрольные вопросы.....94

Библиографический список 96

Глава 1
Сложные типы. Массивы

Понятие сложного типа

Сложные типы характеризуются тем, что любое значение такого типа состоит из множества компонентов (элементов) каким-то образом связанных между собой в единое целое, т. е. имеет сложную структуру. Значение сложного типа строится по следующим правилам:

а) элемент значения сложного типа может иметь как простую, так и сложную структуру, на самом нижнем уровне которой элементы только простого типа (при этом уровень вложенности может ограничиваться или нет). Таким образом, значения сложных типов в общем случае имеют иерархическую структуру;

б) внутри сложной структуры тип всех элементов может быть:

– одинаков в однородной структуре,

– различен в неоднородной структуре;

в) количество элементов в структуре может быть:

– фиксировано в течение времени существования структуры (структуры фиксированного размера или статические),

– переменным, т. е. динамически меняться во время выполнения программы путем включения или исключения элементов в процессе работы со структурой (структуры переменного размера или динамические);

г) обращение (доступ) к элементам значения сложного типа может быть:

– непосредственное (прямое) - вычисляемое (по индексу или месту в структуре) или не вычисляемое (по имени элемента);

– последовательное - характерное для структур переменного размера.

Вид обращения определяется способом объединения компонент в единую структуру;

д) значение структуры может храниться либо в оперативной (внутренняя структура), либо во внешней памяти (файл).

Сложный тип доопределяется пользователем (пользователь определяет структуру значения), поэтому у него нет имени как у простого базового типа.

Тип массив

Массив – это пример сложного (структурированного) типа. Значение сложного типа массив (тип массив или регулярный тип) состоит из однородных элементов, количество которых (размер значения типа массив) фиксируется при определении типа, доступ к элементам – прямой вычисляемый.

Элементы массива занимают непрерывную область оперативной памяти, т. е. последовательно без промежутков располагаются друг за другом. Элементы в значении массива нумеруются, начиная с нуля.

Определение типа массив обычно осуществляется одновременно с определением переменной. Определение переменной регулярного типа (массива) имеет вид:

<спецификация типа> <идентификатор> [<константное выражение>];

Здесь квадратные скобки являются терминальными символами. Константное выражение определяет число компонентов (элементов) в массиве, поэтому его значение целого типа. Идентификатор – это имя переменной типа массив. Тип элементов массива задается спецификацией типа. Тип элементов может быть любой (кроме файлового). Если тип компонентов:

– простой, то определяемая структура будет одномерной (линейной);

– сложный, то определяемая структура будет многомерной (нелинейной).

Для обращения к элементам массива необходимо указать имя массива и место (индекс) элемента в структуре:

имя массива [<индекс>]

Индекс задается выражением, значение которого должно быть целого типа и определяет номер компонента. Значение индекса принадлежит диапазону от нуля до размера массива, уменьшенного на единицу.

Примеры задания переменных типа массив:

const int Nmax = 20;
float x[*Nmax* + 1];
int vector [10];
char line [80];

Переменная *x* – это линейный (одномерный) массив с 21 элементом: *x*[*0*], *x*[*1*], …, *x*[*Nmax*]. Тип каждого элемента (компонента) вещественный.

Переменная *vector* – это линейный (одномерный) массив с 10 элементами: *vector* [*0*], *vector* [*1*], ... , *vector* [*9*]. Тип каждого компонента целый.

Переменная line одномерный массив с 80 компонентами: *line* [*0*], *line* [*1*], … , *line* [*79*], элементы типа *char*.

Многомерный массив – это массив, элементы которого также типа массив.

Определение переменной типа многомерный массив:

<спецификация типа> <идентификатор> [<K_1>][K_2]…[*Kn*];

Здесь K_1, K_2, ... , *Kn* – константные выражения. Причем K_1 задает размер массива по первому измерению, K_2 размер по второму измерению, а *Kn* по *n*-му измерению.

Например, описание *x*[*k1*][*k2*] задает массив *x*, состоящий из *k*1 элементов. Каждый элемент *x* имеет тип массив, состоящий из *k*2 элементов. Таким образом, двумерный массив рассматривается как одномерный массив, каждый элемент которого также одномерный массив. Иначе можно сказать, что *x* – это двумерный массив (матрица), где *k*1 – размер по первому измерению, т. е. количество строк в двумерном массиве - матрице, *k*2 – размер по второму измерению, т. е. количество столбцов в матрице. Элементы матрицы хранятся в памяти ЭВМ по строкам.

Для обращения к элементам *n*-мерного массива необходимо указать имя массива и место (индексы) элемента в структуре:

имя массива [<индекс1>][<индекс2>]…[<индекс *n*>]

В многомерном массиве можно ссылаться на *p*-мерный подмассив *n*-мерного массива ($p \leqslant n$).

Над элементами массива можно выполнять все операции, допустимые для типа элемента.

Примеры задания переменных типа многомерный массив:

int mas [5] [25];
float x[10][10], *y*[*n*1][*n*2][*n*3];

Переменная *mas* это двумерный массив из 5 строк и 25 столбцов, состоящий из 125 компонент:
mas[0][0], *mas*[0][1], *mas*[0][2], ... , *mas*[0][24],
mas[1][0], *mas*[1][1], ... , *mas*[1][24],
mas[2][0], ... , *mas*[2][24], ... , *mas*[4][0], ... , *mas*[4][24], тип компонент *int*.

Если указать только один индекс, например *mas*[*i*] – это означает одномерный подмассив типа *float* [25], т. е. строку матрицы с номером *i*.

Переменная *x* – двумерный массив с 10 строками и 10 столбцами – квадратная матрица размером 10 ×10, а *y* – трехмерный массив размером *n*1×*n*2×*n*3. Тип компонентов переменных *x* и *y* – *float*. Здесь *n*1, *n*2, *n*3 – именованные константы, значения которых могут быть определены директивой *define* или с помощью модификатора *const*.

На элементы массива *y* можно ссылаться: *y*[0], *y*[1], *y*[2], …, *y*[*n*1 - 1], где *y*[*i*] означает двумерный подмассив с номером *i* размером *n*2×*n*3;

элементы более нижнего уровня обозначаются *y*[0][0], *y*[0][1], ... , *y*[0][*n*2-1], *y*[1][0], ... , *y*[*n*1-1][*n*2-1], где *y*[*i*][*j*] означает одномерный подмассив массива *y* размером *n*3;

элементы самого нижнего уровня обозначаются *y*[0][0][0], *y*[0][0][1], ... , *y*[0][0][*n*3-1], *y*[0][1][0], ... , *y*[0][1][*n*3 - 1], ... , *y*[*n*1 - 1][0][0], ... , *y*[*n*1 - 1][*n*2 - 1][*n*3 - 1].

Указатели и массивы

Обратиться к элементу массива можно еще одним способом – используя для этого указатель. Указатель – это переменная, значением которой является адрес другой переменной (т. е. адрес единицы памяти, которая выделена для переменной). Указатель может ссылаться только на объекты заданного типа.

Имя массива является константой-указателем на первый элемент массива. Для вышеприведенных примеров, имени массива *vecto*r соответствует адрес первого элемента этого массива (т. е. элемента *vector*[0]), имя *line* – константа-указатель на элемент *line*[0]. Увеличивая на единицу значение указателя, получаем адрес следующего элемента (числовое значение адреса при этом увеличивается на размер памяти элемента массива). Имя *vector* указывает на первый элемент массива, а выражение *vector*+*i*, на *i*-й элемент после первого (т. е. *vector*+*i* – это адрес *i*–того элемента массива *vector*). Выражение *(*vector* + *i*) – это значение *i*–того элемента массива *vector*. Таким образом, записи *vector*[*i*] и *(*vector*+*i*) эквивалентны. Символ ′*′ это унарная операция косвенной адресации (разыменование). Ее результатом является значение, на которое указывает операнд - указатель.

Имя массива *mas* – указатель, поставленный в соответствие первому из пяти элементов типа *float* [25], выражение *(*mas* + 0) – указатель на элемент *mas*[0]; *(*mas* + *i*) – указатель на элемент *mas*[*i*]. Если выражение *(*mas* + *i*) это указатель на элемент *mas*[*i*] типа *float* [25], то выражение *(*mas* + i) + *j* это указатель на элемент *mas*[*i*][*j*]. Выражение *(*(*mas* + *i*) + *j*) это значение элемента с адресом *(*mas* + *i*) + *j*, т. е. элемента *mas*[*i*][*j*].

Для трехмерного массива *y* выражение *(*y* + *i*) – указатель на элемент *y*[*i*], выражение *(*y*+*i*)+*j* – указатель на элемент *y*[*i*][*j*], а выражение *(*(*y* + *i*) + *j*) + *k* – указатель на элемент *y*[*i*][*j*][*k*], выражение *(*(*(*y* + *i*) + *j*) + *k*) – это значение элемента *y*[*i*][*j*][*k*].

Если определить переменную указатель, то доступ к элементам массива можно осуществлять через эту переменную.

Определение переменной типа указатель на значение специфицированного типа:

<спецификация типа> * <идентификатор>;

Здесь символ ′*′ является признаком типа указатель.

Пример определения переменных типа указатель:

*int *pv*;
*float *py*;

Переменная *pv* – это указатель на значение типа *int* (т. е. значение *pv* это адрес области памяти, в которой может храниться число типа *int*), а *py* – указатель на значение типа *float*. Для вышеописанных массивов в результате присваивания *pv = vector*, *pv* будет указывать на первый элемент массива, иначе говоря, значение *pv* – это адрес элемента *vector*[0];

**pv* – это выражение с унарной операцией косвенной адресации (разыменование), ее результатом является значение первого элемента массива, что эквивалентно *vector*[0];

*(*pv* + *i*) – это выражение с унарной операцией косвенной адресации, результатом которой будет значение *i*–того элемента массива, что эквивалентно *vector*[*i*], если *pv* указывает на первый элемент массива (если нет, то на *i*-ый после *pv*, а *pv* - *i* на *i*-ый перед *pv*).

Переменную *py* можно использовать для ссылки на любой из массивов с элементами типа *float*. Если присвоить *py* =&*mas*[0][0], то *py* равно указателю на *mas*[0][0], выражение *(*py* + *i*) + *j* это указатель на элемент *mas*[*i*][*j*]. Если присвоить *py* = &*x*[*i*][*j*], то *py* указывает на *x*[*i*][*j*] элемент массива *x*, выражение *py* + 1 – это указатель на элемент, следующий после элемента *x*[*i*][*j*]. Изменяя значение указателя (например, *py*++) можно переходить от одного элемента массива к другому.

Применение массивов при решении задач

Необходимость в использовании массивов при решении задач возникает всякий раз, когда приходится иметь дело с большим, но конечным числом однотипных данных. В таком случае все данные можно объединить в каком-то порядке следования в единую структуру, которой дается одно имя. Это позволяет в компактном и единообразном