Лекция 8. Массивы и указатели. Часть 1.

Ревун Артем Леонидович
ст. преп. кафедры вычислительных систем

Курс «Программирование»,осенний семестр,2024
Сибирский государственный университет телекоммуникаций и информатики (г. Новосибирск)

Содержание раздела

- Массивы. Концепция и ключевое понимание
- Глубокое понимание операции: **&**(унарная), ***(унарная)
- Создание и инициализация массивов
- Указатели и их применение при работе с массивам, а также написание функций, обрабатывающих массивы
- Двумерные массивы и двухмерные указатели … etc.

Программа на языке Си

Области памяти, доступные в языке Си, можно классифицировать следующим образом.

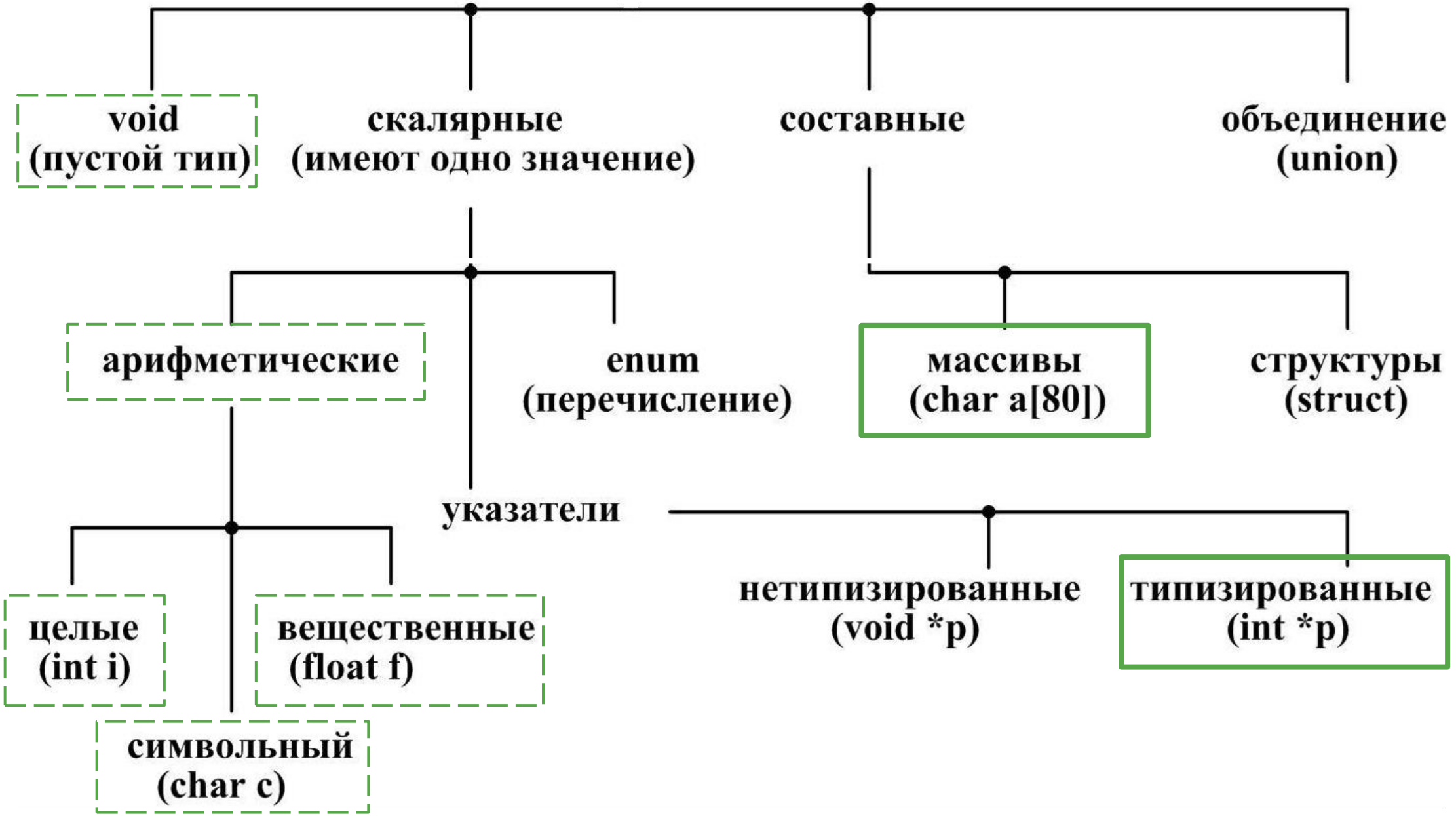
1.Переменные базовых типов данных.

2.Массивы базовых типов данных.

3.Переменные и массивы сложных типов данных.

4.Неименованные области (динамическая память).

Типы данных в Си (прогресс)



Концепция массивов

Обработка информации с использованием ЭВМ зачастую направлена на работу с большими объёмами связанных данных.

Именно обработка всего потока данных позволяет прийти к конечному умозаключению и выводу. Чтобы обрабатывать данные хранящиеся в файлах, часто приходится использовать массивы.

Массив – это составной тип данных в основе которого лежит набор однотипных *скалярных* или *составных типов данных*.

Объявление массивов

Для объявления массива указывается его *тип данных, имя массива* и через **оператор квадратные скобки []** *целочисленным значением* указывается, *количество элементов массива*.

```
int numbers[10];
```

Для доступа к конкретному значению массива используются *индексы* элементов массива.

Индексы начинаются с 0 и заканчиваются N-1,где N – количество элементов массива. numbers[0] – первый элемент,numbers[9] – последний элемент.

Имя	-	numbers	-	-	-	-	-	-	-	-	-	-	-	-	-
Адрес	0x04	0x08	0x0C	0x10	0x14	0x18	0x1C	0x20	0x24	0x28	0x2C	0x30	0x34	0x38	0x3C
Значение	2453	37266	25424	37135	20846	67695	14168	33119	0	42	10465	340544	24859	49825	61599
Индекс	-	0	1	2	3	4	5	6	7	8	9	-	-	-	-

Инициализация массивов (1)

Для инициализации массивов, после его указания, через **оператор =** в **операторе {}** *(фигурных скобках)* указываются значения,которые должны быть записаны в память.

```
int numbers[10] = {1,2,3,4,5,6,7,8,9,10};
```

Имя	-	numbers	-	-	-	-	-	-	-	-	-	-	-	-	-
Адрес	0x04	0x08	0x0C	0x10	0x14	0x18	0x1C	0x20	0x24	0x28	0x2C	0x30	0x34	0x38	0x3C
Значение	2453	1	2	3	4	5	6	7	8	9	10	340544	24859	49825	61599
Индекс	-	0	1	2	3	4	5	6	7	8	9	-	-	-	-

Если количество значений инициализирующих массив, **превышает** заявленное *константное значение размера массива*,компилятор даст предупреждение, но не ошибку.

```
int numbers[10] = {1,2,3,4,5,6,7,8,9,10,11,12,13};
```

Инициализация массивов (2)

В стандарте C99 добавлены – *назначенные инициализаторы*, они позволяют устанавливать конкретные значения массивов, все остальные не указанные значения заполняются нулями.

```
int numbers[10] = {[5]= 10,[7]=55};
```

Имя	-	numbers	-	-	-	-	-	-	-	-	-	-	-	-	-
Адрес	0x04	0x08	0x0C	0x10	0x14	0x18	0x1C	0x20	0x24	0x28	0x2C	0x30	0x34	0x38	0x3C
Значение	2453	0	0	0	0	0	10	0	55	0	0	340544	24859	49825	61599
Индекс	-	0	1	2	3	4	5	6	7	8	9	-	-	-	-

Если 10 задать через `const int SIZE = 10;` то использовать назначенные инициализаторы не получится, так как компилятор не может расширить константное значение, а литеральное или препроцессорное может. (см Размеры массивов)

Инициализация массивов (3)

Если при описании *назначенного инициализатора*, продолжить указывать значения, то они будут записаны в массив если количество вводимых элементов способно в нём поместится.

```
int numbers[10] = {[5]= 10,15,20,25,[0]=100};
```

Имя	-	numbers	-	-	-	-	-	-	-	-	-	-	-	-	-
Адрес	0x04	0x08	0x0C	0x10	0x14	0x18	0x1C	0x20	0x24	0x28	0x2C	0x30	0x34	0x38	0x3C
Значение	2453	100	0	0	0	0	10	15	20	25	0	340544	24859	49825	61599
Индекс	-	0	1	2	3	4	5	6	7	8	9	-	-	-	-

Если количество значений инициализирующих массив, **превышает** заявленное *константное значение размера массива*, компилятор даст предупреждение, но не ошибку.

```
int numbers[10] = {[5]= 10,15,20,25,30,35,40};
```

Инициализация массивов (4)

Во избежание ошибок с размером инициализации массива, лучше придерживаться следующего подхода:

```
int numbers[] = {[5]= 10,15,20,25,30,35,40};
```

Имя	-	numbers	-	-	-	-	-	-	-	-	-	-	-	-	-
Адрес	0x04	0x08	0x0C	0x10	0x14	0x18	0x1C	0x20	0x24	0x28	0x2C	0x30	0x34	0x38	0x3C
Значение	2453	0	0	0	0	0	10	15	20	25	30	35	40	49825	61599
Индекс	-	0	1	2	3	4	5	6	7	8	9	10	11	-	-

Ограничения массивов

```
#define SIZE 10
int numbers[SIZE] = {[5]= 10,[7]=55};
int copynum[SIZE];
copynum = numbers;
copynum = {1,2,3,4,5};
```

Не допустимо: присваивать значения одного массива другому, а также применять форму задания значений списком, кроме как при инициализации массивов.

Границы массива (1)

```
#define SIZE 5
int numbers[SIZE] = {1,2,4,8,16};
```

Исходя из философии языка Си, ответственность за использование массива возлагается на программиста. Создав массив из 5 значений вы должны **гарантировать** применение к нему только *индексов* от 0…4 и компилятор не обязан вас проверять. ~~А на самом деле,в большинстве случаев, он и не может вас проверить.~~

При не корректном применение *индексов* при работе с массивом, поведение программы становится не предсказуемой (выглядит работающей, завершается аварийно, в определенных случаях работает странно).

Границы массива (2)

```
#define SIZE 5
int valuel = 44; int arr[SIZE];
int value2 = 88; int i;
printf ("valuel = %d,value2 = %d\n",valuel,value2);
for (i = -1; i <= SIZE; i++)
    arr[i] = 2 * i + 1;
for (i = -1; i < 7; i++)
    printf("%2d %d\n",i,arr[i]);
    printf("valuel = %d,value2 = %d\n",valuel,value2);
    printf("адрес arr[-1]: %p\n",&arr[-1]);
    printf("адрес arr[4]: %p\n",&arr[4]);
    printf("адрес valuel: %p\n",&valuel);
    printf("адрес value2: %p\n",&value2);
```

Границы массива (3)

```
value1 = 44,value2 = 88
-1 -1
 0 1
 1 3
 2 5
 3 7
 4 9
 5 532
 6 44
value1 = 44,value2 = -1
адрес arr[-1]: 0000004bd03ff7bc
адрес arr[4]: 0000004bd03ff7d0
адрес value1: 0000004bd03ff7d8
адрес value2: 0000004bd03ff7bc
```

Размеры массива (1)

До этого момента, при объявлении массивов применялись целочисленные **константы**:

```
#define SIZE = 5
int arr[SIZE]; //константа препроцессора
double lots[144]; //целочисленный литерал
```

Выдержка из стандарта: При объявлении массива в квадратных скобках должно быть помещено **константное целочисленное выражение**, выражение сформированное из целочисленных констант. В этом смысле оператор **sizeof** является целочисленной константой, но значение **const** – нет. Значение такого выражения должно было быть больше 0.

Размеры массива (2)

```
int n = 5;
int m = 8;
float a1[5];
float a2[5*2 + 1];
float a3[sizeof(int) + 1];
float a4[-4];
float a5[0];
float a6[2.5];
float a7[(int)2.5];
float a8[n];
float a9[m];
```

Какие выражения не верны?

Размеры массива (3)

```
int n = 5;
int m = 8;
float a1[5];
float a2[5*2 + 1];
float a3[sizeof(int) + 1];
float a4[-4];          //отрицательное значение
float a5[0];           //нулевое значение
float a6[2.5];         //не целое число
float a7[(int)2.5];
float a8[n];           //Компиляторы C90 не разрешают
float a9[m];           //два последних объявления.
```

Размеры массива (4)

Однако в стандарте C99 вводится понятие *массивы переменной длинны,*но в стандарте C11,от этой смелой инициативы отошли,и сделали это лишь дополнительной возможностью языка,но не обязательной.

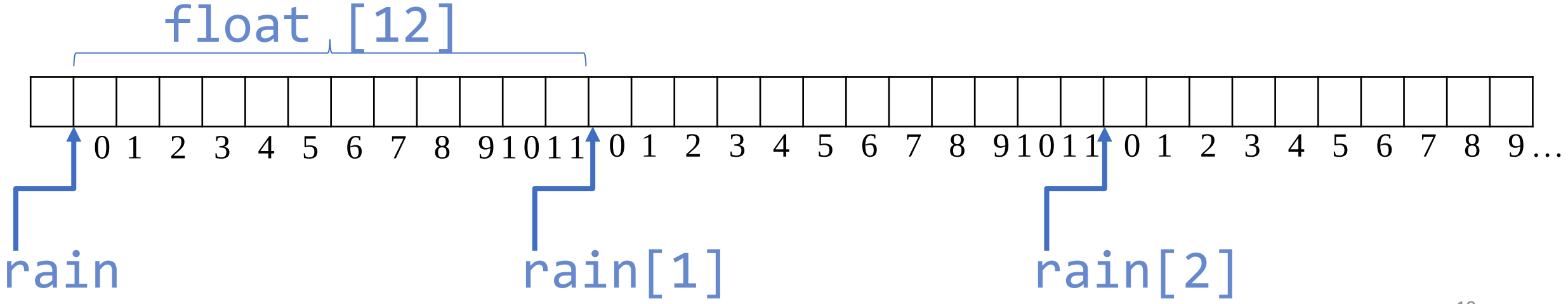
Таким образом планировали повысить совместимость языков Си с языком FORTRUN.

Не смотря на это,такие массивы ограничены тем,что их нельзя инициализировать при его объявлении.

Многомерные массивы (1)

Стандарт Си предполагает создание многомерных массивов.

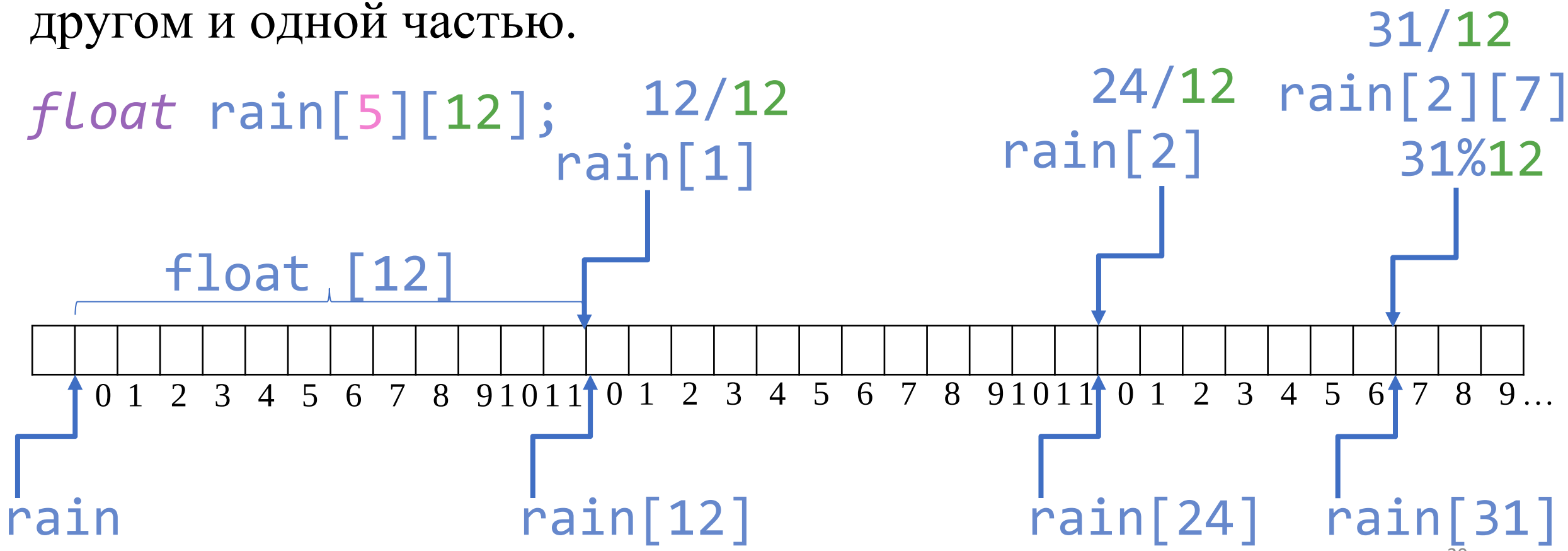
```
float rain[5][12]; //создаем массив rain с 5 не значениями неизвестного типа.
float rain[5][12]; //массив из 12 значений типа float
float rain[5][12]; //массив из 5 массивов по 12 значений float
```



Многомерные массивы (2)

Многомерные массивы это лишь абстракция,которая нужна для более удобного визуального восприятия.

На самом деле данные в оперативной памяти хранятся друг за другом и одной частью.



Многомерные массивы (3)

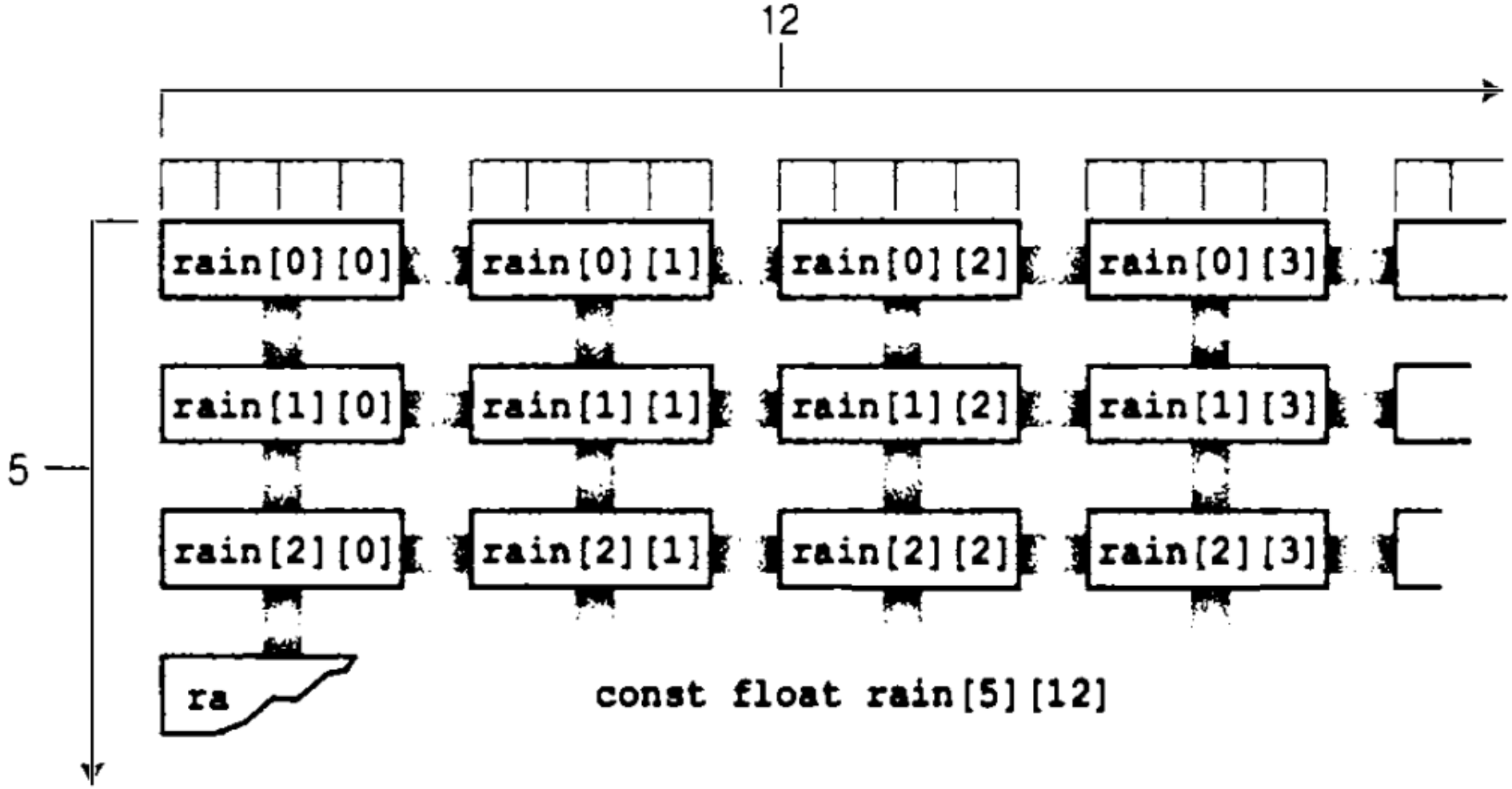


Рис. 10.1. Двумерный массив

Пример применения многомерного массива (1)

```
#define MONTHS 12
#define YEARS 5
const float rain[YEARS][MONTHS] = {
{4.3,4.3,4.3,3.0,2.0,1.2,0.2,0.2,0.4,2.4,3.5,6.6},
{8.5,8.2,1.2,1.6,2.4,0.0,5.2,0.9,0.3,0.9,1.4,7.3},
{9.1,8.5,6.7,4.3,2.1,0.8,0.2,0.2,1.1,2.3,6.1,8.4},
{7.2,9.9,8.4,3.3,1.2,0.8,0.4,0.0,0.6,1.7,4.3,6.2},
{7.6,5.6,3.8,2.8,3.8,0.2,0.0,0.0,0.0,1.3,2.6,5.2}
};
```

Пример применения многомерного массива (2)

```
int year, month;
float subtot, total;
printf("ГОД КОЛИЧЕСТВО ОСАДКОВ (в дюймах)\n");
for (year = 0, total = 0; year < YEARS; year++)
{
    for (month = 0, subtot = 0; month < MONTHS;
month++)
    subtot += rain[year][month];
    printf("%5d %15.1f\n", 2010 + year, subtot);
    total += subtot;
}
```

Пример применения многомерного массива (3)

```
printf("Среднегодовое количество осадков
составляет %.1f дюймов. \n",total/YEARS);
printf("СРЕДНЕМЕСЯЧНОЕ КОЛИЧЕСТВО ОСАДКОВ:\n") ;
printf ("Янв Фев Мар Апр Май Июн Июл Авг Сен Окт
Ноя Дек\n");
for (month = 0; month < MONTHS; month++) {
for (year = 0, subtot = 0; year < YEARS; year++)
subtot += rain[year][month];
printf("%4.1f ", subtot/YEARS); }
printf("\n");
return 0;
```

Пример применения многомерного массива (4)

```
ГОД КОЛИЧЕСТВО ОСАДКОВ (в дюймах)
 2010              32.4
 2011              37.9
 2012              49.8
 2013              44.0
 2014              32.9
Среднегодовое количество осадков составляет 39 дюймов.
СРЕДНЕМЕСЯЧНОЕ КОЛИЧЕСТВО ОСАДКОВ:
 Янв  Фев  Мар  Апр  Май  Июн  Июл  Авг  Сен  Окт  Ноя  Дек
 7.3  7.3  4.9  3.0  2.3  0.6  1.2  0.3  0.5  1.7  3.6  6.7
```

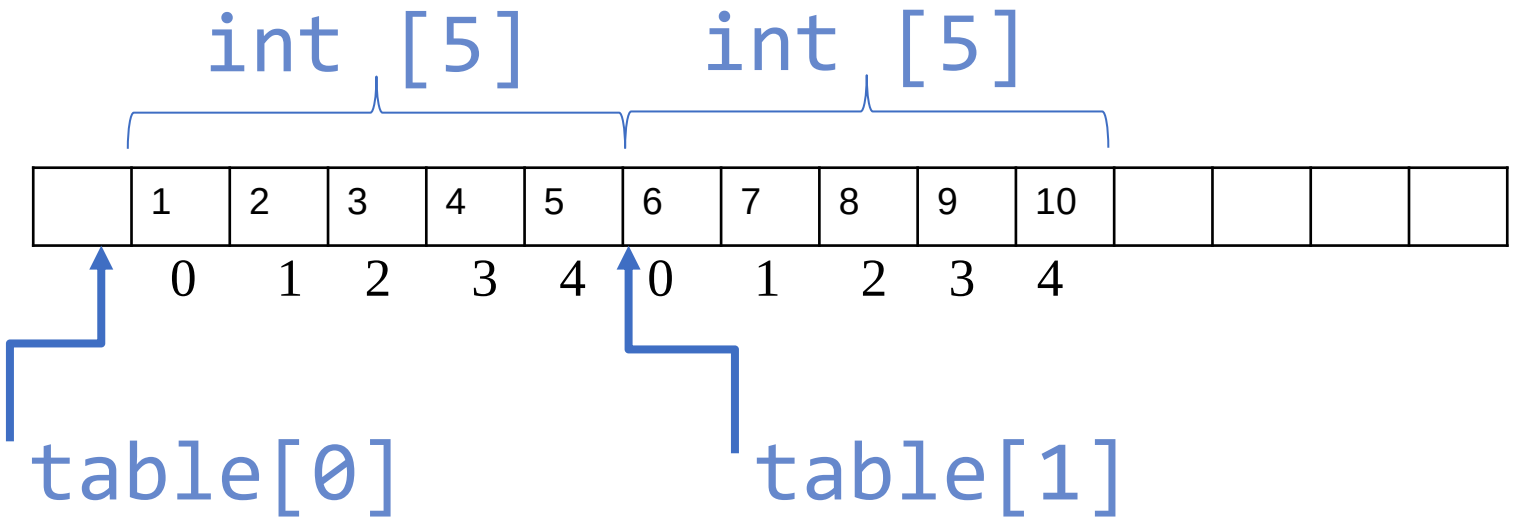
Инициализация многомерных массивов

По концепции схожи с инициализацией обычного массива

```
#define ROWS 2
#define COLUMS 5
int table[ROWS][COLUMS]={{1,2,3,4,5},{6,7,8,9,10}};
```

или

```
int table[ROWS][COLUMS]={1,2,3,4,5,6,7,8,9,10};
```



Задачи на практику

Читать:

Главу 10. Массивы и указатели. (стр 367-383)

Практическая работа:

ПР 8. (Зачёт) Массивы и указатели. Часть 1.Файл

Спасибо за внимание

Ревун Артем Леонидович
ст. преп. Кафедры вычислительных систем

Курс «Программирование»,осенний семестр,2024
Сибирский государственный университет телекоммуникаций и информатики (г. Новосибирск)