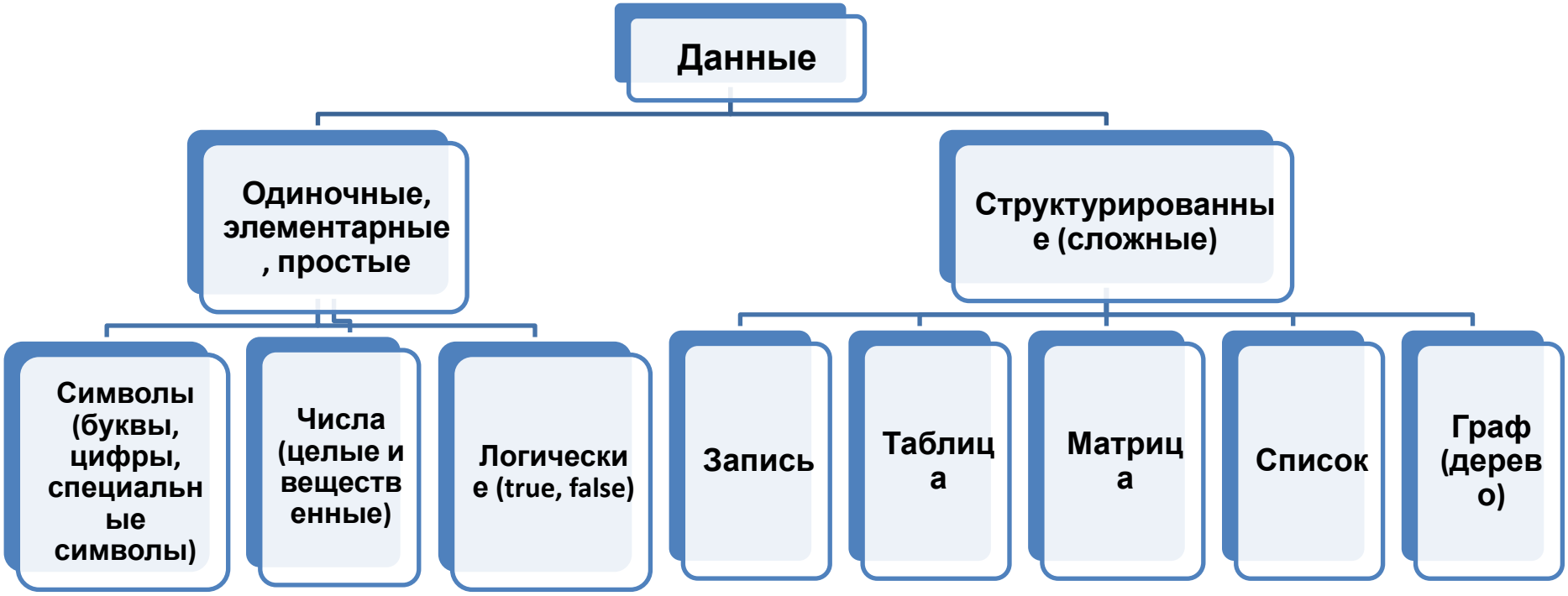
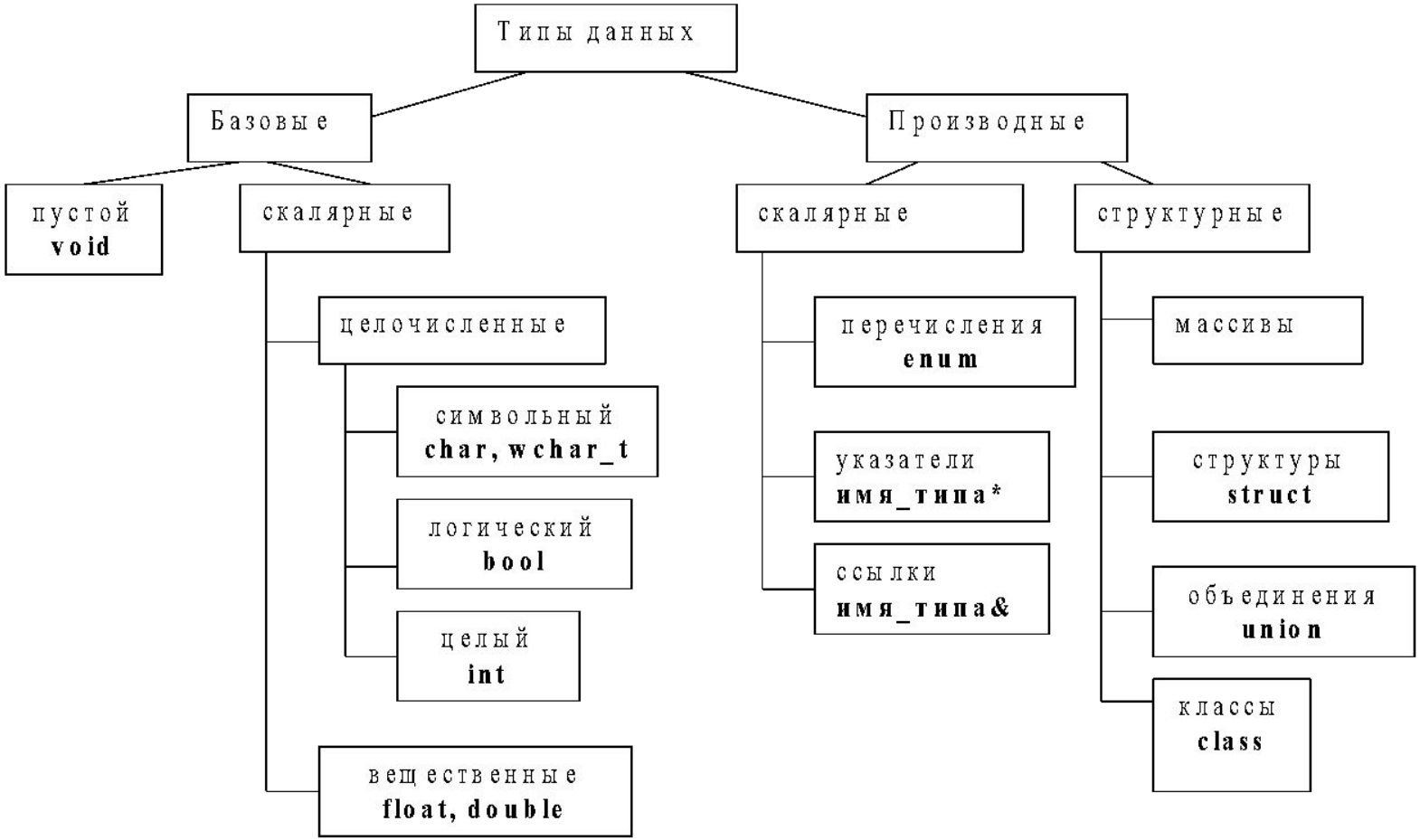
Базовые типы и простейшие структуры данных



Одиночное данное имеет собственное имя (идентификатор) и **одно** значение. Значение – содержимое ячеек памяти, отведенных под хранение. Каждое данное имеет определенный тип, допустимый в конкретном языке программирования

Путем объединения простых данных можно получить структурированные данные. Различные варианты объединения приводят к возникновению множества структур данных. **Структура данных** – совокупность данных и связей (отношений) между ними. При описании структуры данных определяется набор возможных операций и порядок доступа к данным

Классификация типов данных (C/C++)



Тип данных определяет:

1. Внутреннее представление данных в памяти компьютера.
2. Множество значений, которые могут принимать величины этого типа.
3. Операции, в которых могут участвовать данные этого типа.

Размер и диапазон базовых типов данных

Имя типа	Название	Размер (в байтах)	Диапазон значений
bool	логический	1	true и false
char (signed char)	символьный (знаковый)	1	-128 ... 127
unsigned char	беззнаковый символьный	1	0 ... 255
int (signed int)	целый (знаковый)	4	-2^{31} ... $2^{31}-1$
unsigned int	беззнаковый целый	4	0 ... $2^{32}-1$
short int (signed short int)	короткий целый (знаковый)	2	-32768 ... 32767
unsigned short int	беззнаковый короткий целый	2	0 ... 65535
long int (signed long int)	длинный целый (знаковый)	4	-2^{31} ... $2^{31}-1$
unsigned long int	беззнаковый длинный целый	4	0 ... $2^{32}-1$
float	с плавающей точкой одинарной точности	4	±1.17549435E-38 ±3.40282347E+38
double	с плавающей точкой двойной точности	8	±2.22507385E-308 ±1.79769313E+308
long double	с плавающей точкой повышенной (расширенной) точности	10	±3.36210314E-4932 ±1.18973149E+4932

Структурный тип `struct`

Для адекватного представления объектов реального мира используют структурный тип (структура). Структура состоит из фиксированного числа элементов **разного** типа. Элементы структуры называются полями.

Синтаксис определения структуры:

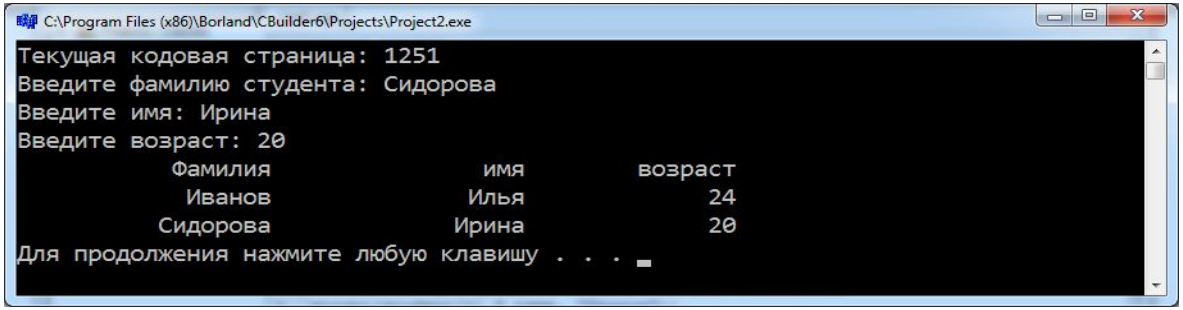
```
struct  [имя_структуры]

{список_полей} [список_переменных_структурного_типа];
```

Само по себе определение структурного типа не влечет за собой выделение памяти, а только задает внутреннюю организацию структурных переменных, которые будут созданы позже на основе этого типа. Только после объявления переменной структурного типа выделяется память, достаточная для хранения всех ее полей.

```
#include <iostream.h> //для ввода с клавиатуры и вывода на экран
#include <windows.h> //для system()
#include <iomanip.h> //для setw()
const int lenght = 20; //максимальное количество элементов в массиве
struct person
{
char f_name[32];
char name[32];
short age;
} Titov;


void main()
{
system("chcp 1251");
person student[lenght];
//инициализация нулевого элемента массива константами
strcpy(student[0].f_name, "Иванов");
strcpy(student[0].name, "Илья");
student[0].age=24;
//инициализация первого элемента путем ввода данных с клавиатуры
cout << "Введите фамилию студента: ";
cin >> student[1].f_name;
cout << "Введите имя: ";
cin >> student[1].name;
cout << "Введите возраст: ";
cin >> student[1].age;
//вывод на экран
cout <<setw(18)<<"Фамилия"<<setw(18)<<"имя" <<setw(15)<<"возраст"<<endl ;
for(int i=0; i<2; i++)
cout<< setw(18)<<student[i].f_name <<setw(18)<<student[i].name <<setw(15)<<
   student[i].age<<endl;
system("pause");
}
```



Простейшая структура данных – массив

Массив – упорядоченная линейная совокупность однородных данных с последовательным хранением элементов.

Положение элемента (порядок) определяется индексом. Количество индексов называется **мерностью** массива:

- одномерный массив, вектор, строка;
- двумерный массив, матрица;
- многомерный массив.

Произведение максимальных значений индексов определяет **размер** массива (количество элементов).

```
int A[2][3]; //размер массива 6
```

Допустимый набор операций над массивом определяется типом данных его элементов (элементарных или структурированных).

Статический массив	Динамический массив
размещается в статической памяти (программный стек), ее выделение происходит на этапе компиляции программы	размещается в динамической памяти (куча), ее выделение происходит на этапе выполнения программы
в процессе исполнения программы его размер не может изменяется	по ходу работы программы можно менять размер массива

`person student[lenght];`	`cout << "Введите количество студентов";`
	`int k;`
	`cin >> k;`
	`person *p = new person[k];`

Строка – символьный массив

Строка в C – это массив символов (char), оканчивающийся нулевым символом ‘\0’ (конец строки). Обращение ко всей строке осуществляется по имени массива. Доступ к отдельному символу строки – по индексу.

```
char text[] = ”This is a string”; // 17 байт
cout<<text;  //вывод строки на экран
cin>>text;  //ввод с клавиатуры до пробела или ENTER, т.е. 1 слово
cin.getline(text, 16);  /*считывает не более 16 символов с клавиатуры в массив до
                                нажатия ENTER*/
cin.getline(text, 16, ’.’);  /*считывает не более 16 символов до точки*/
text[10]=’S’;
```

Библиотека функций по обработке строк доступна при подключении заголовочного файла

```
#include <string.h>
```

Функция	Описание	Примеры
strlen(str)	Возвращает длину (количество символов) строки str до нулевого символа в виде целого беззнакового числа	`int l=strlen(text);//l=16`
strcpy(s1, s2)	Копирует строку s2 в s1 (с '\0'). Массив s1 не должен быть меньше s2	`strcpy(buff, "This is a test");` `cout<<buff; //This is a test`
strcat(s1, s2)	Добавляет s2 к s1. Результат сцепления помещается в s1 и добавляется нулевой символ. s1 должна быть достаточного размера	`char s1[20]="Happy ", s2[]="New Year";` `strcat(s1,s2); //в s1 Happy New Year`

strcmp(s1, s2)	Сравнивает строки s2 и s1:
	- возвращает отрицательное значение, если s1 < s2,
	- нулевое, если s1 == s2,
	- положительное, если s1 > s2. Учитывает регистр букв
		`char s1[]="Happy New Year";`
		`char s2[]="Happy New Year";`
		`char s3[]="Happy Holidays";`
		`int n = strcmp(s1,s2); // n = 0`
		`n = strcmp(s1,s3); // n = 1`
		`n = strcmp(s3,s1); // n =-1`

strcmpi(s1, s2)	Аналогично, но не учитывает регистр букв	
strlwr(s) **strupr(s)**	Преобразует буквенные символы в нижний регистр (строчные буквы) или верхний (пописные)	`char s1[]="Happy New Year";` `cout<<strlwr( s1 );//happy new year`
atoi(s)	Преобразует строку s в целое число. Кроме цифр строка может содержать пробелы вначале и знаки '+' или '-'. Преобразование идет до первого недопустимого символа. Если в строке нет символов, пригодных к преобразованию, то возвращает 0	`char s1[] = "123.e-2";` `cout << atoi(s1);//123`
atof(s)	Преобразует строку s в вещественное число. Кроме цифр строка может содержать пробелы вначале, знаки '+' или '-', десятичную точку и символы e или E. Преобразование идет до первого недопустимого символа. Если в строке нет символов, пригодных к преобразованию, то возвращает 0	`char s1[] = "123.e-2";` `cout << atof(s1);//1.23`

```
char strings[][9] = {"string 1","string 2","string 3"};//Массив из 3
строк
  for (int i=0; i<3 ;i++)
  {
     cout<< strings[i]<<endl ;
  }
```

Вывод на экране:

```
string 1
string 2
string 3
```

Классификация структур данных

1. Отношение порядка:

- упорядоченные

В упорядоченных структурах элементы размещаются по порядку в соответствии со значением некоторого признака. Наиболее простым признаком является порядковый номер элемента. Установление порядка в соответствии с номером называется **нумерацией**.

Установление порядка в соответствии со значением элемента называется **ранжированием**.

Примеры: массивы, линейные списки

- неупорядоченные

  Характерно отсутствие упорядоченности по какому-либо признаку.

  Примеры: множество (не важен порядок элементов, а только их принадлежность/не принадлежность множеству), граф.

2. **Однородность** данных:

- однородные

  Структуры, содержащие данные только одного типа или одной структуры.

  Примеры: массив, множество, список.

- неоднородные

  Структуры, объединяющие данные разных типов.

  Пример: структурный тип (запись), мультисписок.

3. Подчиненность:

- линейные

  В линейных структурах все элементы равнозначны.
  Примеры: массив, множество, список.

- нелинейные

  В нелинейных структурах между элементами существует неравноправие, например, отношение подчиненности (иерархия).
  Пример: дерево, мультисписок.

4. Взаимное физическое расположение элементов:

- последовательные

  Элементы структуры данных в памяти располагаются последовательно друг за другом, занимая соседние ячейки памяти. Адрес любого элемента(кроме первого) можно вычислить, зная адрес предыдущего элемента и его размер.
  Пример: массив.

- связные

  Элементы могут занимать не соседние ячейки памяти. Связь между ними осуществляется посредством указателей.
  Примеры: связный список, дерево.

5. Вид памяти, используемой для хранения данных:

- оперативные структуры

  Это структуры данных, размещенные в статической и динамической памяти компьютера.

  Примеры: массив, список, дерево.

- файловые структуры

  Файловыми структурами или файлами называют структуры данных для внешней памяти.

  Примеры: последовательные файлы, B-деревья.

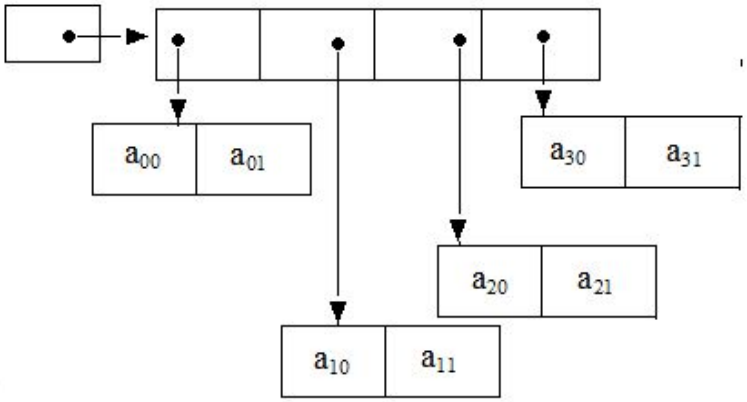
Каждая структура данных характеризуется её **логическим** и **физическим** представлениями. Чаще всего, говоря о той или иной структуре данных, имеют в виду её логическое представление. Физическое представление обычно не соответствует логическому, и кроме того, может быть реализовано разными способами.

Например, двумерный массив A размером 4x2 имеет единственное логическое представление (1) в виде таблицы, состоящей из 4 строк и 2 столбцов, и как минимум два физических(2), (3):

(2)

a_{11}	a_{12}
a_{21}	a_{22}
a_{31}	a_{32}
a_{41}	a_{42}

(1)



(3)