



Факультатив «Углубленное изучение языка C»

Преподаватель:

Балабаев Сергей Андреевич



Система оценивания

- ☐ Баллы в ОРИОКС ставятся за посещаемость ☹️
- ☐ Посещаемость 1 занятия = 1.25 балла
- ☐ Будут 4 БДЗ по 10 баллов
- ☐ В зачет дублируются баллы за семестр



План занятий

- ☐ Занятие – дополнительное
- ☐ Можно (и даже нужно) гуглить
- ☐ Не стесняемся задавать вопросы
- ☐ Если стесняемся, то можно писать в лс
- ☐ Стараемся вдумываться...

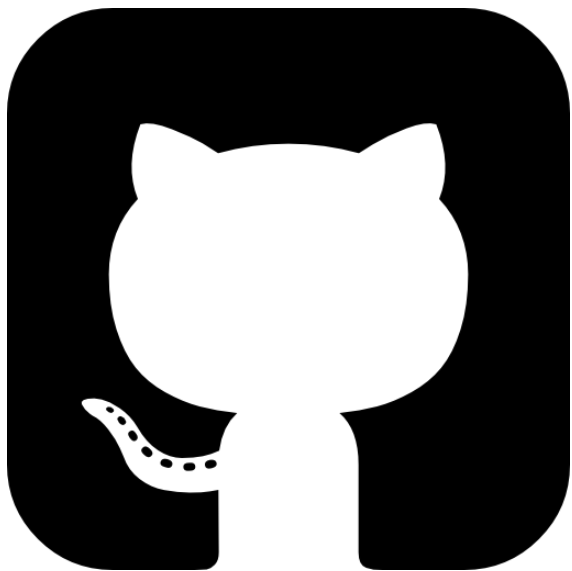


План занятий

| № | Тема | Описание |
|---|-------------------------|--|
| 1 | Введение в курс | Языки программирования. Основы работы с Linux. |
| 2 | Основы языка C | Написание и компиляция простейших программ с использованием gcc. Правила написания кода. |
| 3 | Компиляция | Разбиение программы на отдельные файлы. Make файлы. Компиляция. |
| 4 | Ввод данных. Библиотеки | Работа со входом/выходом. Работа с файлами в языке C. Статические и динамические библиотеки. |
| 5 | Хранение данных. Память | Хранение процесса в памяти компьютера. Виртуальная память, сегментация. Секции программы. |
| 6 | Устройство памяти. | Elf файлы. Указатели и массивы. Типы данных. Gdb и отладка |
| 7 | Аллокация памяти | Аллокация памяти. Битовые операции – сдвиги, логические операции. Битовые поля. Перечисления. Static переменные. Inline функции. |
| 8 | Язык ассемблера | Язык ассемблера. Вызов функции. Безопасные функции. Макросы |



Материалы курса



<https://github.com/SergeyBalabaev/Elective-C-Programming-Language>



Телеграмм канал курса
https://t.me/+ze4N3uj2H_g1Yjgy



vk: sergei_balabaev
tg: @sergeybalabaev



Рекомендованная литература

1. *Ашарина И.В.* Язык программирования C++. Конспект лекций по курсу “Информатика”. - М.: МИЭТ, 2000. - 112 с.: ил
2. *Брайан Керниган, Деннис Ритчи* - Язык программирования Си Москва: Диалектика, 2020.
3. *Прата С.* Язык программирования C. Лекции и упражнения, 6-е изд. : Пер. с англ. — М. :ООО “И.Д. Вильямс”, 2015. — 928 с.

4. *Столяров А.В.* Низкоуровневое программирование. Том 2 Глава 4
5. *Дэвид М. Харрис и Сара Л. Харрис* Цифровая схемотехника и архитектура компьютера. Приложение «С»

6. *Igor Zhirkov* Low-Level Programming: C, Assembly, and Program Execution on Intel 64 Architecture
7. *Richard Reese* Understanding and Using C Pointers-O Reilly Media 2013
8. *Suzanne J. Matthews* Dive into Systems <https://diveintosystems.org/>
9. Видеолекции МФТИ – Тимофей Хирьянов (Youtube)



Вместо введения

Задача

Необходимо перемножить две матрицы размерностью $N \times N$

Решение для $N=2$

$$C_{ij} = \sum_{s=1}^n A_{is} B_{sj} \quad A = \begin{pmatrix} a_1 & b_1 \\ a_2 & b_2 \end{pmatrix} \quad B = \begin{pmatrix} c_1 & d_1 \\ c_2 & d_2 \end{pmatrix}$$

$$\begin{pmatrix} a_1 & b_1 \\ a_2 & b_2 \end{pmatrix} \begin{pmatrix} c_1 & d_1 \\ c_2 & d_2 \end{pmatrix} = \begin{pmatrix} ? & ? \\ ? & ? \end{pmatrix}$$



Вместо введения

Задача

Необходимо перемножить две матрицы размерностью $N \times N$

Решение

$$C_{ij} = A_{i1}B_{1j} + A_{i2}B_{2j}$$
$$C_{11} = A_{11}B_{11} + A_{12}B_{21}$$
$$\begin{pmatrix} a_1 & b_1 \\ a_2 & b_2 \end{pmatrix} \begin{pmatrix} c_1 & d_1 \\ c_2 & d_2 \end{pmatrix} = \begin{pmatrix} a_1c_1 + b_1c_2 & ? \\ ? & ? \end{pmatrix}$$

$$\begin{pmatrix} a_1 & b_1 \\ a_2 & b_2 \end{pmatrix} \begin{pmatrix} c_1 & d_1 \\ c_2 & d_2 \end{pmatrix} = \begin{pmatrix} a_1c_1 + b_1c_2 & a_1d_1 + b_1d_2 \\ a_2c_1 + b_2c_2 & a_2d_1 + b_2d_2 \end{pmatrix}$$



Вместо введения

Задача

Необходимо перемножить две матрицы размерностью $N \times N$

Решение

$$C_{ij} = \sum_{s=1}^n A_{is} B_{sj} \quad \begin{pmatrix} A_{11} & * & A_{1n} \\ * & * & * \\ A_{n1} & * & A_{nn} \end{pmatrix} \begin{pmatrix} B_{11} & * & B_{1n} \\ * & * & * \\ B_{n1} & * & B_{nn} \end{pmatrix} = \begin{pmatrix} ? & * & ? \\ * & * & * \\ ? & * & ? \end{pmatrix}$$

$$C_{ij} = A_{i1} B_{1j} + A_{i2} B_{2j} + \dots + A_{in} B_{nj}$$

$$\begin{pmatrix} A_{11} & * & A_{1n} \\ * & * & * \\ A_{n1} & * & A_{nn} \end{pmatrix} \begin{pmatrix} B_{11} & * & B_{1n} \\ * & * & * \\ B_{n1} & * & B_{nn} \end{pmatrix} = \begin{pmatrix} ? & * & ? \\ * & * & * \\ ? & * & ? \end{pmatrix}$$



Вместо введения

```
for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
        for (int s = 0; s < n; s++)
            C[i][j] = C[i][j] + A[i][s] * B[s][j];
```

$i=0, j=0, s=0$ $C[0][0] = C[0][0] + A[0][0] * B[0][0];$

$i=0, j=0, s=1$ $C[0][0] = C[0][0] + A[0][1] * B[1][0];$

$i=0, j=0, s=n-1$ $C[0][0] = C[0][0] + A[0][n-1] * B[n-1][0];$

$$\begin{pmatrix} \boxed{A_{00} * A_{0(n-1)}} \\ * & * \\ A_{(n-1)0} * A_{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} \boxed{B_{00}} * B_{0(n-1)} \\ * & * \\ B_{(n-1)0} * B_{(n-1)(n-1)} \end{pmatrix} = \begin{pmatrix} \boxed{A_{00}B_{00} + \dots + A_{(n-1)0}B_{0(n-1)}} * * \\ * & * \\ ? & ? \end{pmatrix}$$



Вместо введения

```
for (int i = 0; i < n; i++)  
    for (int s = 0; s < n; s++)  
        for (int j = 0; j < n; j++)  
            C[i][j] = C[i][j] + A[i][s] * B[s][j];
```

i=0, s=0, j=0

C[0][0] = C[0][0] + A[0][0] * B[0][0];

i=0, s=0, j=1

C[0][1] = C[0][1] + A[0][0] * B[0][1];

i=0, s=0, j=n-1

C[0][n] = C[0][n-1] + A[0][0] * B[0][n-1];

$$\begin{pmatrix} \boxed{A_{00}} & * & A_{0(n-1)} \\ * & * & * \\ A_{(n-1)0} & * & A_{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} \boxed{B_{00}} & * & \boxed{B_{0(n-1)}} \\ * & * & * \\ B_{(n-1)0} & * & B_{(n-1)(n-1)} \end{pmatrix} = \begin{pmatrix} \boxed{A_{00}B_{00}} & * & \boxed{A_{00}B_{0(n-1)}} \\ * & * & * \\ ? & * & ? \end{pmatrix}$$



Вместо введения

```

for (int i = 0; i < n; i++)
    for (int s = 0; s < n; s++)
        for (int j = 0; j < n; j++)
            C[i][j] = C[i][j] + A[i][s] * B[s][j];

```

$i=0, s=1, j=0$ $C[0][0] = C[0][0] + A[0][1] * B[1][0];$

$i=0, s=1, j=1$ $C[0][1] = C[0][1] + A[0][1] * B[1][1];$

$i=0, s=1, j=n-1$ $C[0][n] = C[0][n-1] + A[0][1] * B[1][n-1];$

$$\begin{pmatrix} A_{00} & * & A_{0(n-1)} \\ * & * & * \\ A_{(n-1)0} & * & A_{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} B_{00} & * & B_{0(n-1)} \\ * & * & * \\ B_{(n-1)0} & * & B_{(n-1)(n-1)} \end{pmatrix} = \begin{pmatrix} A_{00}B_{00} + A_{01}B_{10} & * & A_{00}B_{0(n-1)} + A_{01}B_{1(n-1)} \\ * & * & * \\ ? & * & ? \end{pmatrix}$$



Вместо введения

```
for (int j = 0; j < n; j++)
    for (int s = 0; s < n; s++)
        for (int i = 0; i < n; i++)
            C[i][j] = C[i][j] + A[i][s] * B[s][j];
```

j=0, s=0, i=0

C[0][0] = C[0][0] + A[0][0] * B[0][0];

j=0, s=0, i=1

C[1][0] = C[1][0] + A[1][0] * B[0][0];

j=0, s=0, i=n-1

C[n-1][0] = C[n-1][0] + A[n-1][0] * B[0][0];

$$\begin{pmatrix} A_{00} & * & A_{0(n-1)} \\ * & * & * \\ A_{(n-1)0} & * & A_{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} B_{00} & * & B_{0(n-1)} \\ * & * & * \\ B_{(n-1)0} & * & B_{(n-1)(n-1)} \end{pmatrix} = \begin{pmatrix} A_{00}B_{00} & * & ? \\ * & * & * \\ A_{(n-1)0}B_{00} & * & ? \end{pmatrix}$$



Вместо введения

```
for (int j = 0; j < n; j++)  
    for (int s = 0; s < n; s++)  
        for (int i = 0; i < n; i++)  
            C[i][j] = C[i][j] + A[i][s] * B[s][j];
```

j=0, s=1, i=0 C[0][0] = C[0][0] + A[0][1] * B[1][0];

j=0, s=1, i=1 C[1][0] = C[1][0] + A[1][1] * B[1][0];

j=0, s=1, i=n-1 C[n-1][0] = C[n-1][0] + A[n-1][1] * B[1][0];

$$\begin{pmatrix} A_{00} & * & A_{0(n-1)} \\ * & * & * \\ A_{(n-1)0} & * & A_{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} B_{00} & * & B_{0(n-1)} \\ * & * & * \\ B_{(n-1)0} & * & B_{(n-1)(n-1)} \end{pmatrix} = \begin{pmatrix} A_{00}B_{00} + A_{01}B_{10} & * & ? \\ * & * & * \\ A_{(n-1)0}B_{00} + A_{(n-1)1}B_{10} & * & ? \end{pmatrix}$$



Вместо введения

А есть ли разница?

```

for (int j = 0; j < n; j++)
    for (int s = 0; s < n; s++)
        for (int i = 0; i < n; i++)
            C[i][j] = C[i][j] + A[i][s] * B[s][j];

```

j=0, s=1, i=0 C[0][0] = C[0][0] + A[0][1] * B[1][0];

j=0, s=1, i=1 C[1][0] = C[1][0] + A[1][1] * B[1][0];

j=0, s=1, i=n-1 C[n-1][0] = C[n-1][0] + A[n-1][1] * B[1][0];

$$\begin{pmatrix} A_{00} & * & A_{0(n-1)} \\ * & * & * \\ A_{(n-1)0} & * & A_{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} B_{00} & * & B_{0(n-1)} \\ * & * & * \\ B_{(n-1)0} & * & B_{(n-1)(n-1)} \end{pmatrix} = \begin{pmatrix} A_{00}B_{00} + A_{01}B_{10} & * & ? \\ * & * & * \\ A_{(n-1)0}B_{00} + A_{(n-1)1}B_{10} & * & ? \end{pmatrix}$$



Вместо введения

Хранение массива в памяти

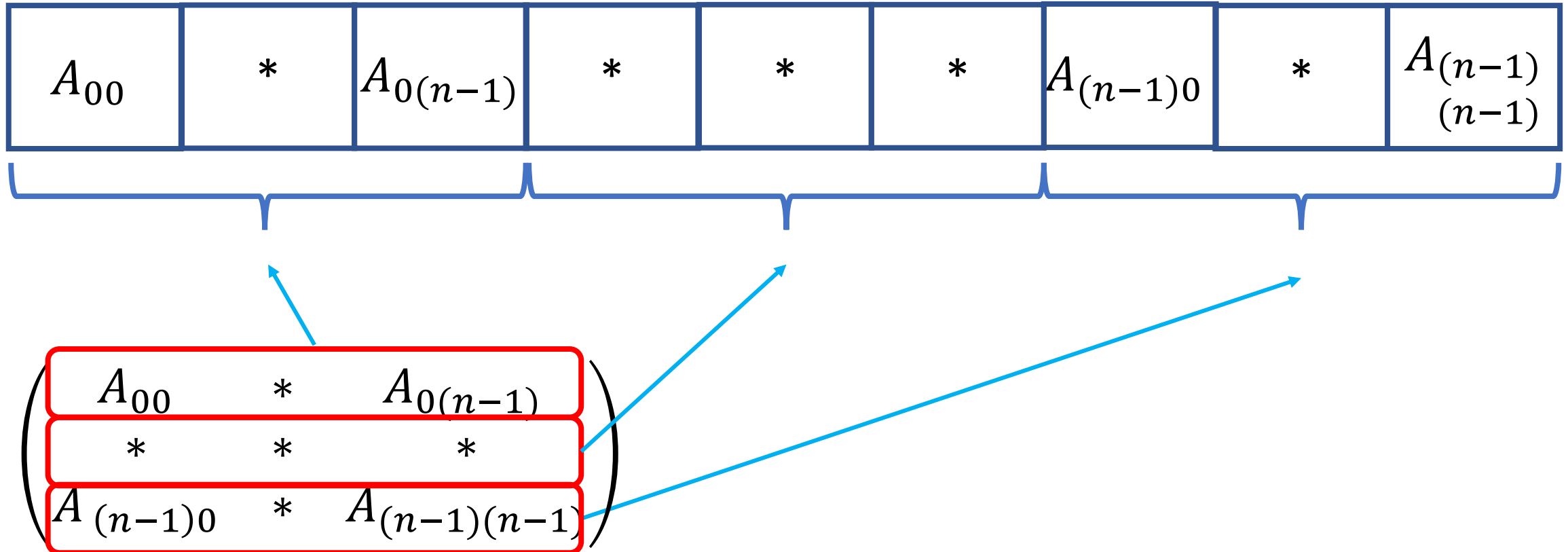
| | | | | | | | | |
|----------|---|--------------|---|---|---|--------------|---|------------------|
| A_{00} | * | $A_{0(n-1)}$ | * | * | * | $A_{(n-1)0}$ | * | $A_{(n-1)(n-1)}$ |
|----------|---|--------------|---|---|---|--------------|---|------------------|

$$\begin{pmatrix} A_{00} & * & A_{0(n-1)} \\ * & * & * \\ A_{(n-1)0} & * & A_{(n-1)(n-1)} \end{pmatrix}$$



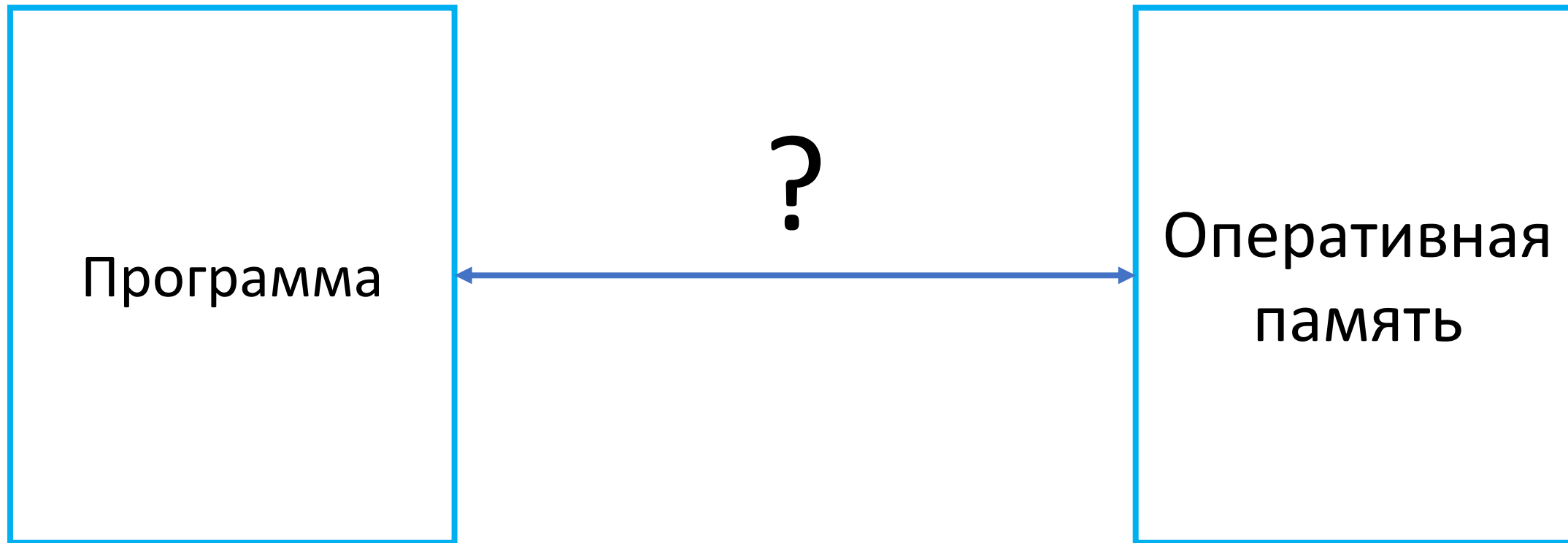
Вместо введения

Хранение массива в памяти



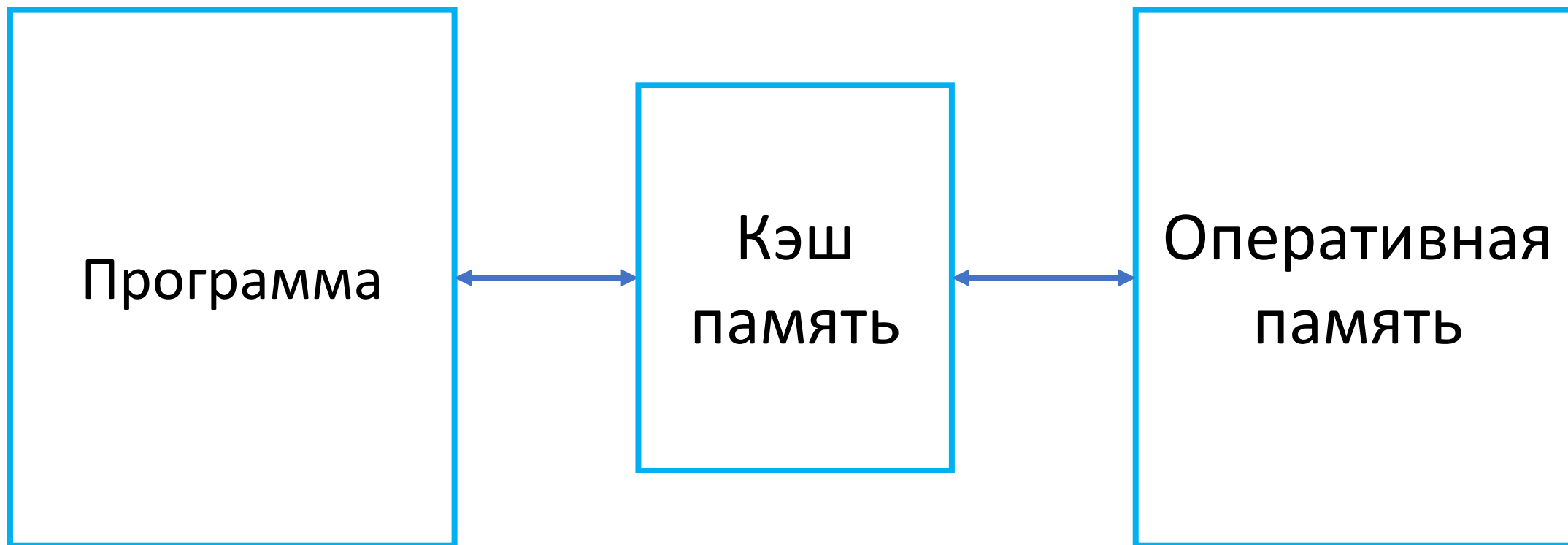


Вместо введения



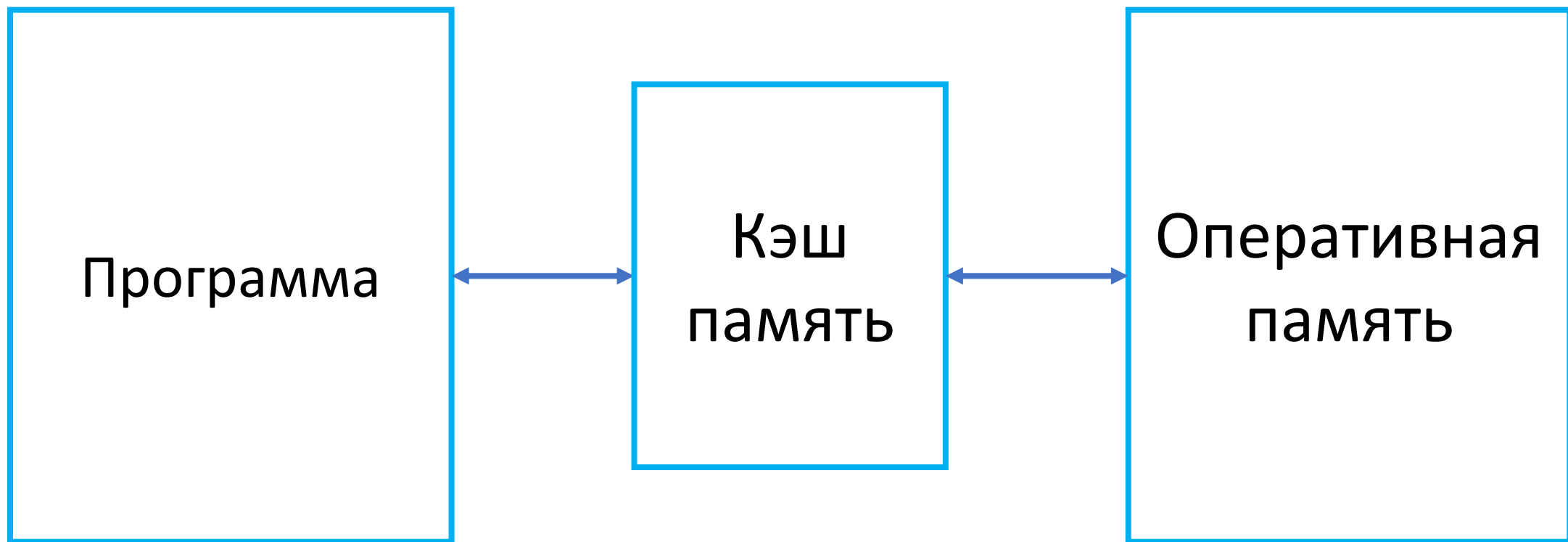


Вместо введения





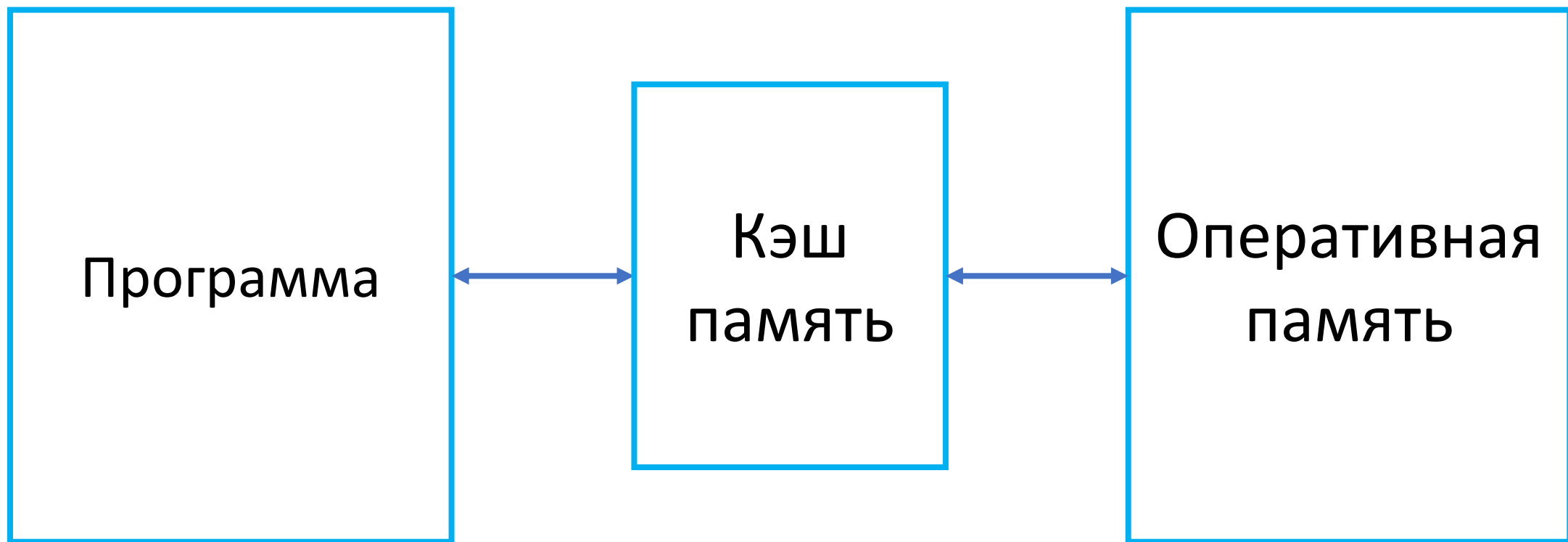
Вместо введения



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|



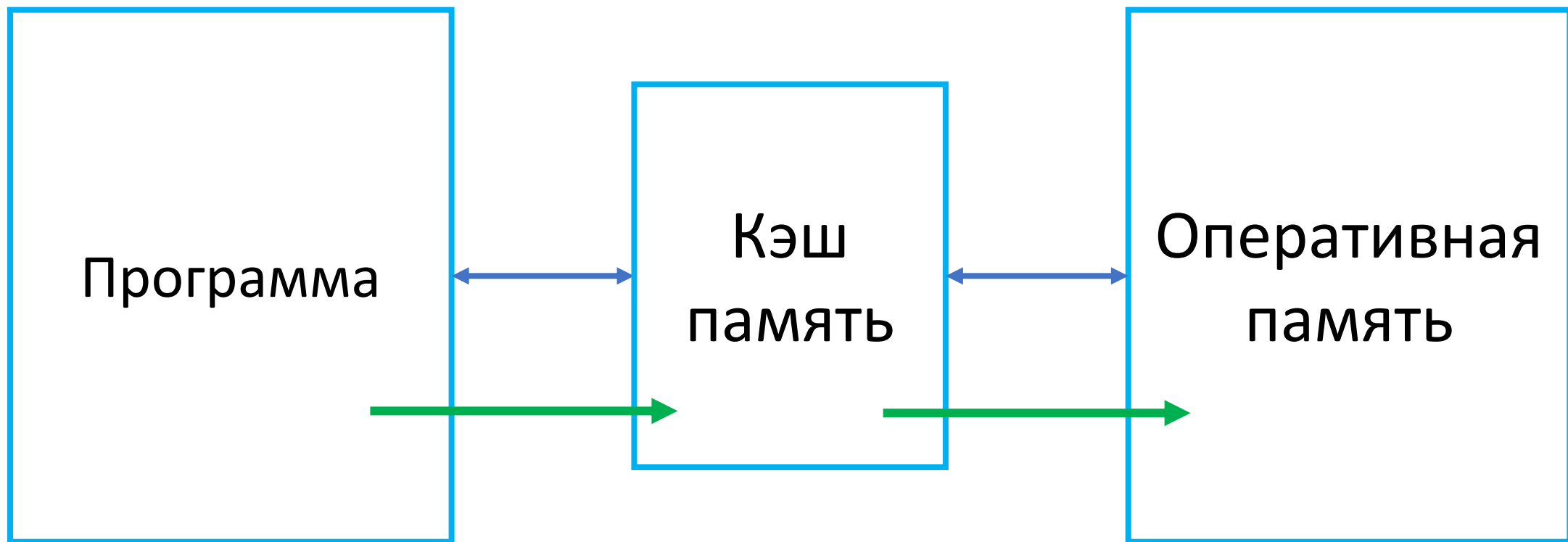
Вместо введения



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|



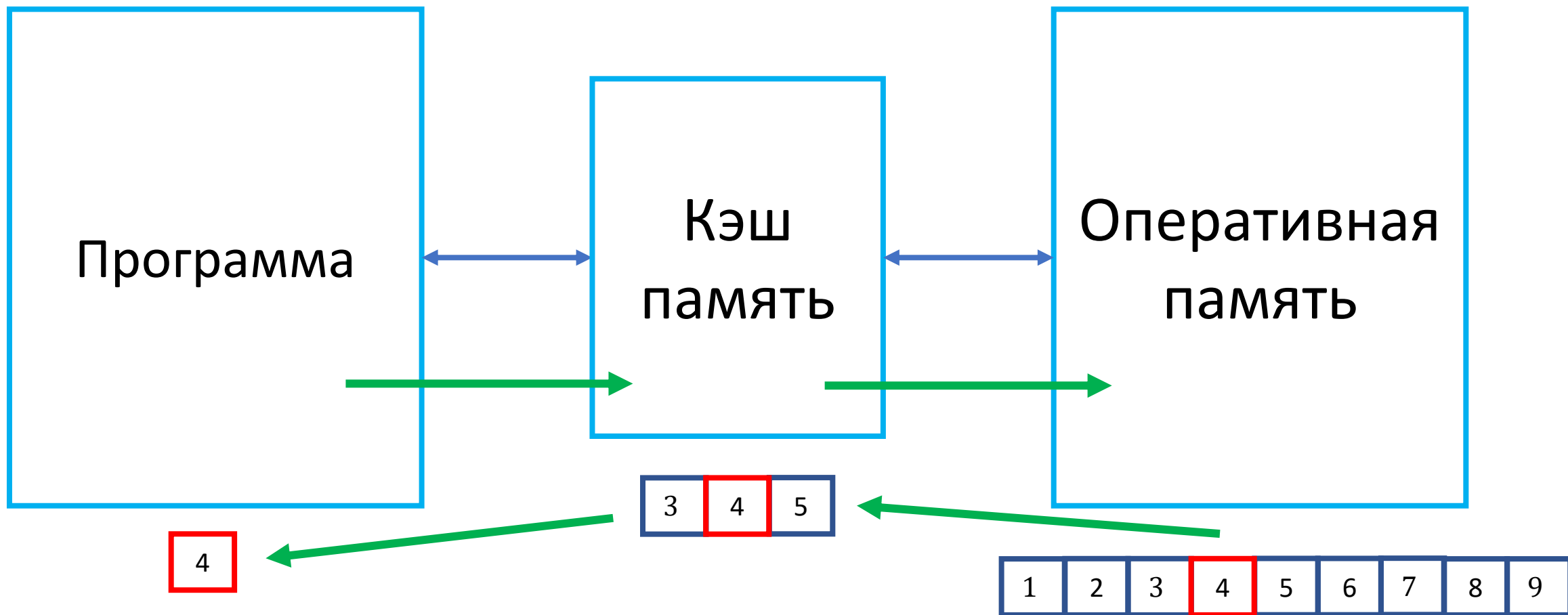
Вместо введения



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|

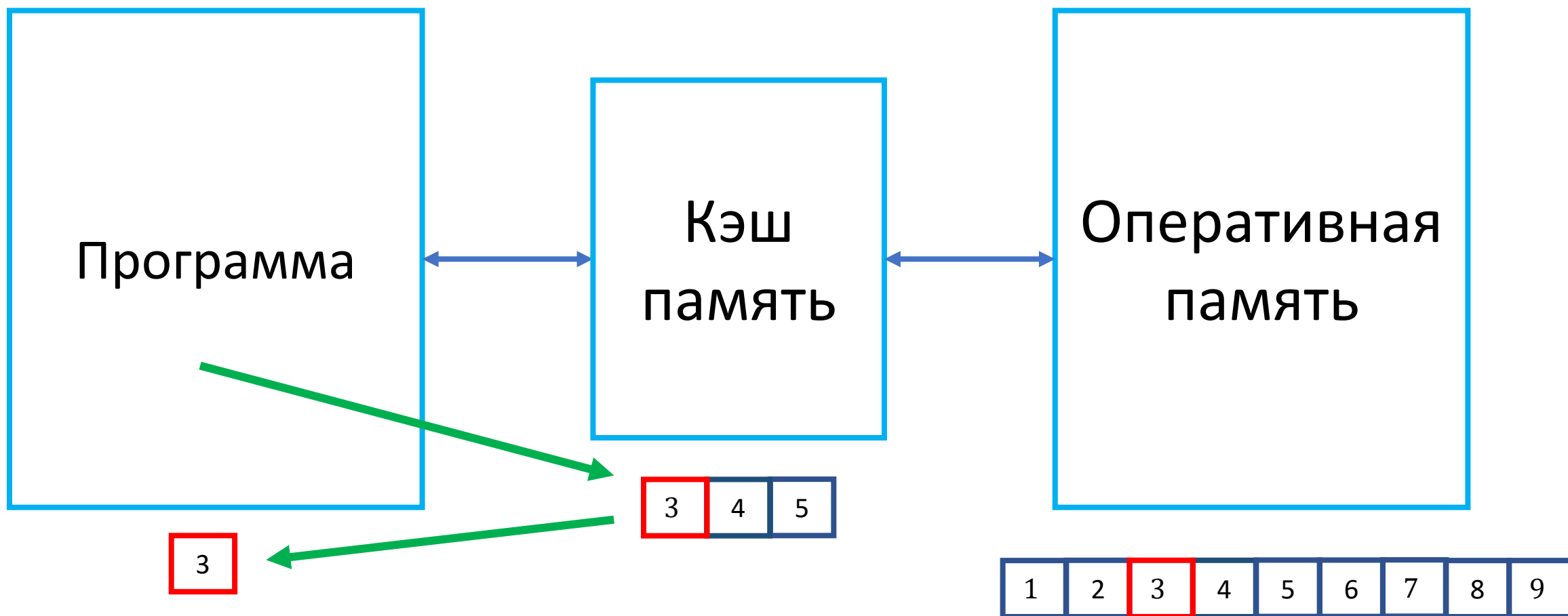


Вместо введения





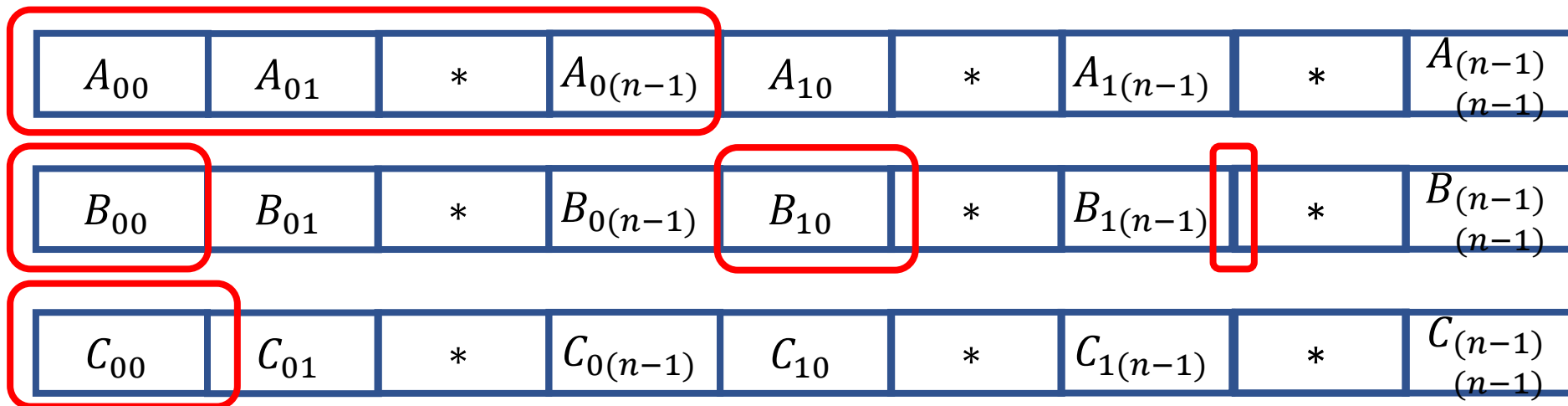
Вместо введения





Вместо введения

Оперативная память

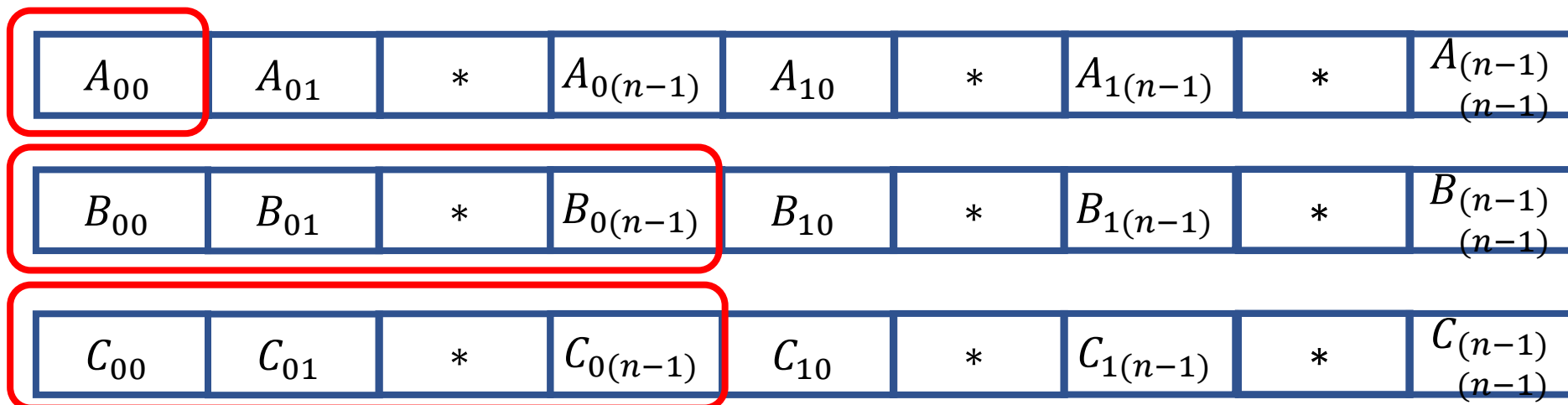


$$\begin{pmatrix} A_{00} & * & A_{0(n-1)} \\ * & * & * \\ A_{(n-1)0} & * & A_{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} B_{00} & * & B_{0(n-1)} \\ * & * & * \\ B_{(n-1)0} & * & B_{(n-1)(n-1)} \end{pmatrix} = \begin{pmatrix} A_{00}B_{00} + \dots + A_{(n-1)0}B_{0(n-1)} & * & * \\ * & * & * \\ ? & * & ? \end{pmatrix}$$



Вместо введения

Оперативная память

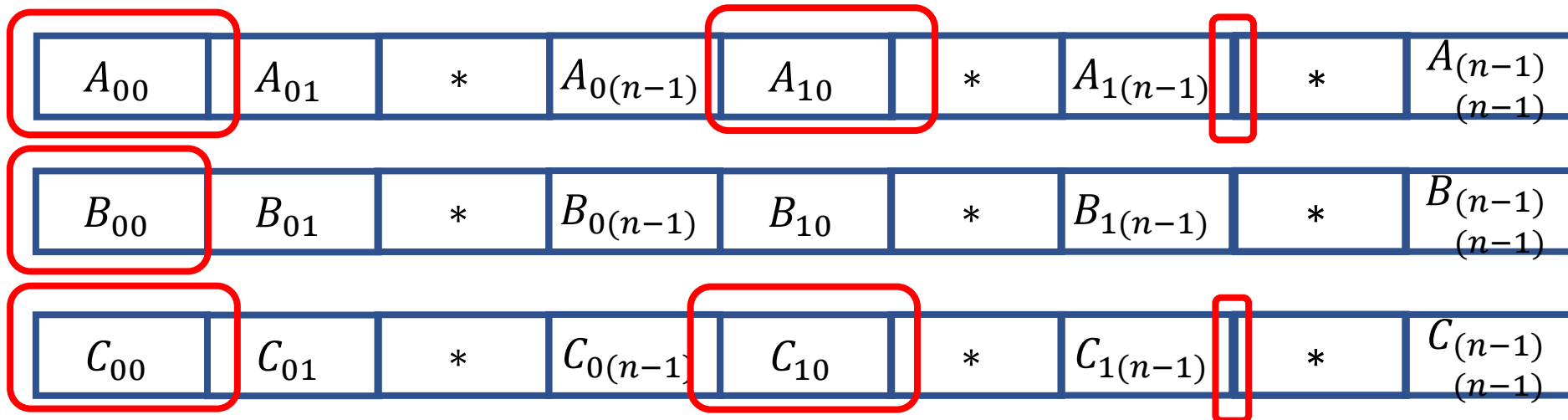


$$\begin{pmatrix} A_{00} & * & A_{0(n-1)} \\ * & * & * \\ A_{(n-1)0} & * & A_{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} B_{00} & * & B_{0(n-1)} \\ * & * & * \\ B_{(n-1)0} & * & B_{(n-1)(n-1)} \end{pmatrix} = \begin{pmatrix} A_{00}B_{00} & * & A_{00}B_{0(n-1)} \\ * & * & * \\ ? & * & ? \end{pmatrix}$$



Вместо введения

Оперативная память



$$\begin{pmatrix} A_{00} & * & A_{0(n-1)} \\ * & * & * \\ A_{(n-1)0} & * & A_{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} B_{00} & * & B_{0(n-1)} \\ * & * & * \\ B_{(n-1)0} & * & B_{(n-1)(n-1)} \end{pmatrix} = \begin{pmatrix} A_{00}B_{00} & * & ? \\ * & * & * \\ A_{(n-1)0}B_{00} & * & ? \end{pmatrix}$$



Вместо введения

$$1. \begin{pmatrix} \boxed{A_{00}} & * & \boxed{A_{0(n-1)}} \\ * & * & * \\ A_{(n-1)0} & * & A_{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} \boxed{B_{00}} & * & B_{0(n-1)} \\ * & * & * \\ \boxed{B_{(n-1)0}} & * & B_{(n-1)(n-1)} \end{pmatrix} = \begin{pmatrix} \boxed{A_{00}B_{00} + \dots + A_{(n-1)0}B_{0(n-1)}} & * & * \\ * & * & * \\ ? & * & ? \end{pmatrix}$$

$$2. \begin{pmatrix} \boxed{A_{00}} & * & A_{0(n-1)} \\ * & * & * \\ A_{(n-1)0} & * & A_{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} \boxed{B_{00}} & * & \boxed{B_{0(n-1)}} \\ * & * & * \\ B_{(n-1)0} & * & B_{(n-1)(n-1)} \end{pmatrix} = \begin{pmatrix} \boxed{A_{00}B_{00}} & * & \boxed{A_{00}B_{0(n-1)}} \\ * & * & * \\ ? & * & ? \end{pmatrix}$$

$$3. \begin{pmatrix} \boxed{A_{00}} & * & A_{0(n-1)} \\ * & * & * \\ \boxed{A_{(n-1)0}} & * & A_{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} \boxed{B_{00}} & * & B_{0(n-1)} \\ * & * & * \\ B_{(n-1)0} & * & B_{(n-1)(n-1)} \end{pmatrix} = \begin{pmatrix} \boxed{A_{00}B_{00}} & * & ? \\ * & * & * \\ \boxed{A_{(n-1)0}B_{00}} & * & ? \end{pmatrix}$$



Вместо введения

Вывод!

Кроме понимания алгоритмов нужно знать то «железо», под которое пишешь программу













Почему мы учим именно язык «С»

| Jan 2021 | Jan 2020 | Change | Programming Language | Ratings | Change |
|----------|----------|--------|----------------------|---------|--------|
| 1 | 2 | ⬆ | C | 17.38% | +1.61% |
| 2 | 1 | ⬇ | Java | 11.96% | -4.93% |
| 3 | 3 | | Python | 11.72% | +2.01% |
| 4 | 4 | | C++ | 7.56% | +1.99% |
| 5 | 5 | | C# | 3.95% | -1.40% |
| 6 | 6 | | Visual Basic | 3.84% | -1.44% |
| 7 | 7 | | JavaScript | 2.20% | -0.25% |
| 8 | 8 | | PHP | 1.99% | -0.41% |
| 9 | 18 | ⬆ | R | 1.90% | +1.10% |
| 10 | 23 | ⬆ | Groovy | 1.84% | +1.23% |












Почему мы учим именно язык «С»

| Jan 2022 | Jan 2021 | Change | Programming Language | | Ratings | Change |
|----------|----------|--------|---|-------------------|---------|--------|
| 1 | 3 | ▲ |  | Python | 13.58% | +1.86% |
| 2 | 1 | ▼ |  | C | 12.44% | -4.94% |
| 3 | 2 | ▼ |  | Java | 10.66% | -1.30% |
| 4 | 4 | |  | C++ | 8.29% | +0.73% |
| 5 | 5 | |  | C# | 5.68% | +1.73% |
| 6 | 6 | |  | Visual Basic | 4.74% | +0.90% |
| 7 | 7 | |  | JavaScript | 2.09% | -0.11% |
| 8 | 11 | ▲ |  | Assembly language | 1.85% | +0.21% |
| 9 | 12 | ▲ |  | SQL | 1.80% | +0.19% |
| 10 | 13 | ▲ |  | Swift | 1.41% | -0.02% |












Почему мы учим именно язык «С»

| Feb 2023 | Feb 2022 | Change | Programming Language | | Ratings | Change |
|----------|----------|--------|---|-------------------|---------|--------|
| 1 | 1 | |  | Python | 15.49% | +0.16% |
| 2 | 2 | |  | C | 15.39% | +1.31% |
| 3 | 4 | ▲ |  | C++ | 13.94% | +5.93% |
| 4 | 3 | ▼ |  | Java | 13.21% | +1.07% |
| 5 | 5 | |  | C# | 6.38% | +1.01% |
| 6 | 6 | |  | Visual Basic | 4.14% | -1.09% |
| 7 | 7 | |  | JavaScript | 2.52% | +0.70% |
| 8 | 10 | ▲ |  | SQL | 2.12% | +0.58% |
| 9 | 9 | |  | Assembly language | 1.38% | -0.21% |



Почему мы учим именно язык «С»

| Feb 2024 | Feb 2023 | Change | Programming Language | | Ratings | Change |
|----------|----------|--------|---|--------------|---------|--------|
| 1 | 1 | |  | Python | 15.16% | -0.32% |
| 2 | 2 | |  | C | 10.97% | -4.41% |
| 3 | 3 | |  | C++ | 10.53% | -3.40% |
| 4 | 4 | |  | Java | 8.88% | -4.33% |
| 5 | 5 | |  | C# | 7.53% | +1.15% |
| 6 | 7 | ^ |  | JavaScript | 3.17% | +0.64% |
| 7 | 8 | ^ |  | SQL | 1.82% | -0.30% |
| 8 | 11 | ^ |  | Go | 1.73% | +0.61% |
| 9 | 6 | v |  | Visual Basic | 1.52% | -2.62% |

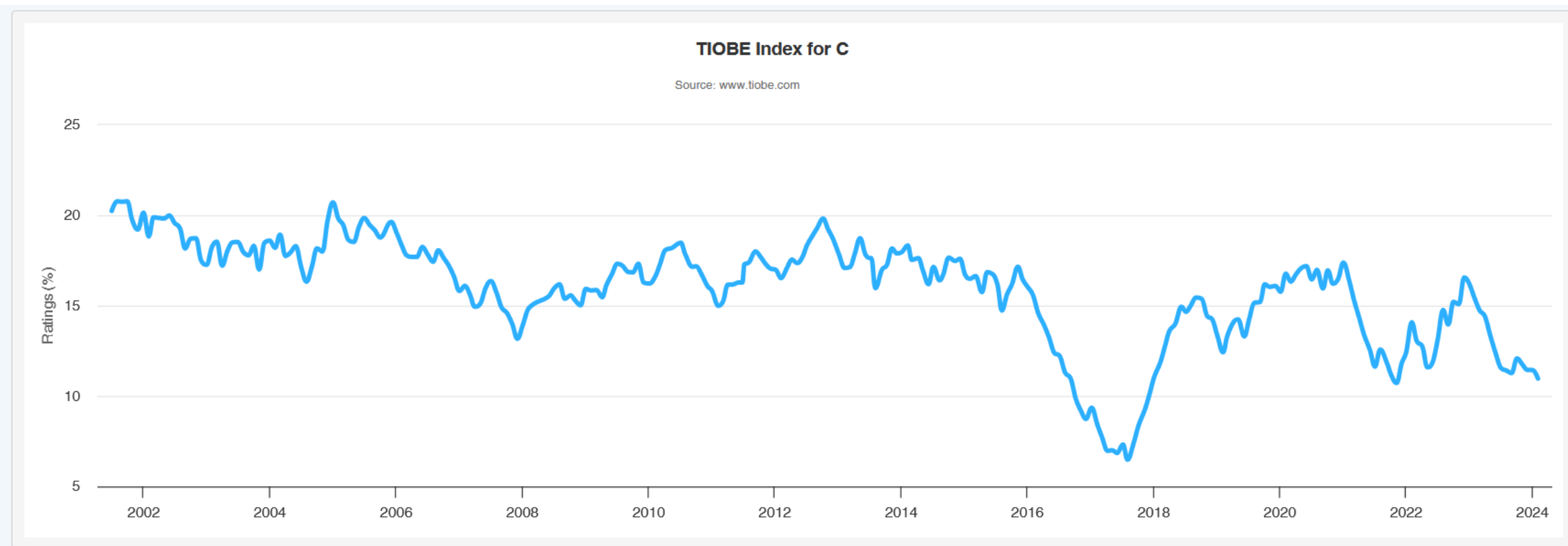


Статистика по github

| Language | Primary Language for % of Repositories | | Occur in % of Repositories | | Total Number of Bytes | |
|------------------|--|--------|----------------------------|--------|-----------------------|----------|
| Python | | 23.21% | | 31.94% | | 1.10 GB |
| TypeScript | | 13.19% | | 18.83% | | 0.19 GB |
| JavaScript | | 7.50% | | 29.60% | | 0.35 GB |
| Rust | | 4.19% | | 5.52% | | 63.21 MB |
| Go | | 4.02% | | 5.17% | | 0.18 GB |
| C# | | 4.02% | | 5.05% | | 0.31 GB |
| C++ | | 3.84% | | 8.61% | | 1.14 GB |
| Jupyter Notebook | | 3.24% | | 5.14% | | 1.07 GB |
| Java | | 2.88% | | 4.30% | | 0.48 GB |
| Kotlin | | 2.71% | | 4.38% | | 60.23 MB |
| C | | 2.63% | | 8.37% | | 3.80 GB |

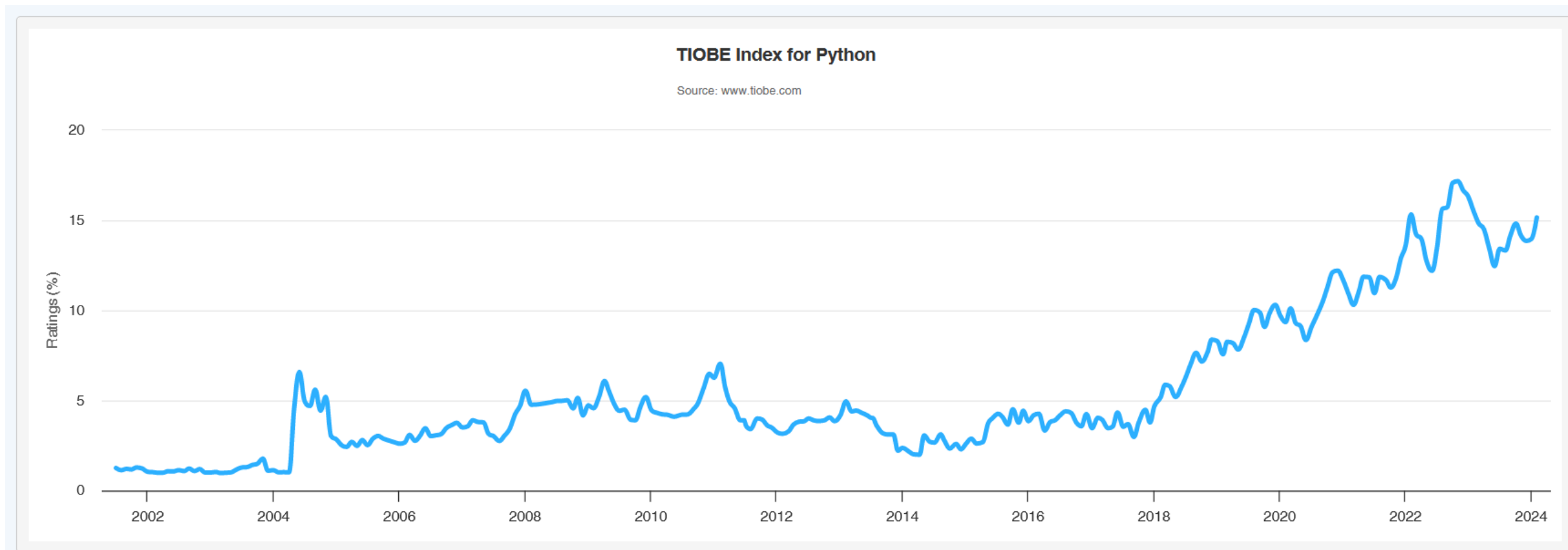


Почему мы учим именно язык «С»



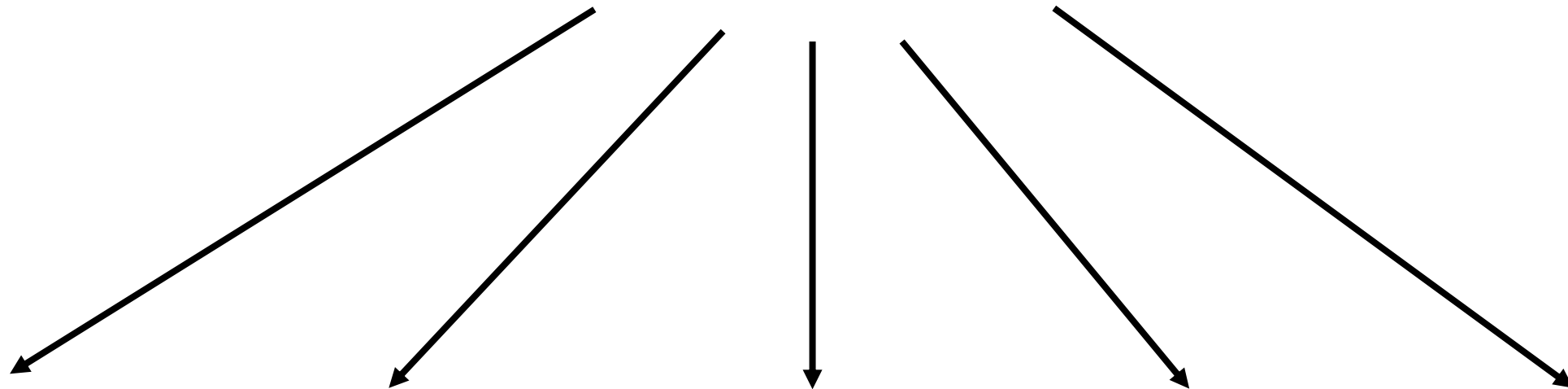


Почему не Python?





Области применения языков программирования



Программирование
микроконтроллеров



- C
- Assembler

Веб разработка



- PHP
- C#
- JavaScript

Приложения для
смартфонов



- Java
- Kotlin

Научные
исследования



- Python
- Matlab
- C++

Разработка
приложений



- C++
- Java
- Python

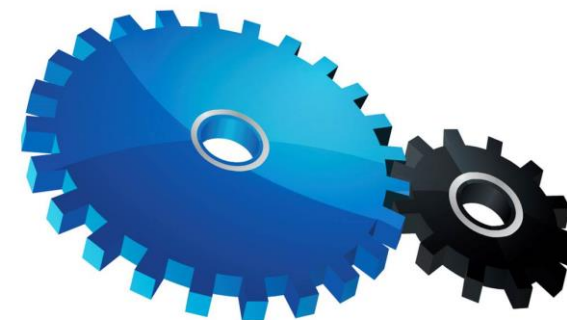
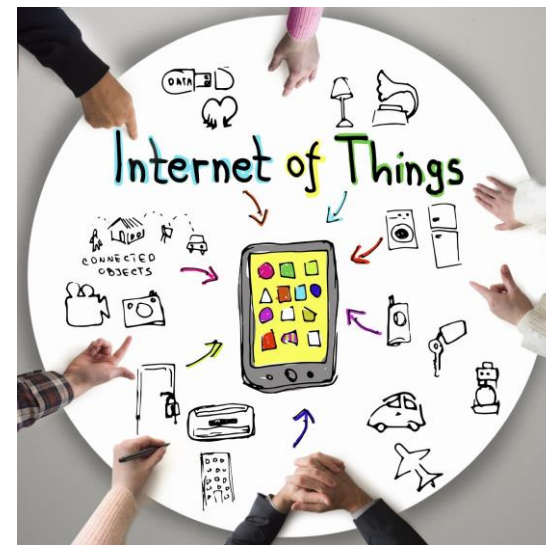


Почему мы учим именно язык «С»



Язык С применяется:

- Микроконтроллеры
- Интернет вещей
- Операционные системы
- Драйверы



Drivers



Парадигмы программирования

Парадигмы программирования

← Императивная

- Структурная
- Процедурная
- Модульная
 - ООП

→ Декларативная

- Функциональная



Парадигмы программирования

main

Настройка порта
Чтение данных
Запись данных в буфер
Чтение данных из буфера
Анализ данных
Открытие текстового файла
Запись данных
Закрытие текстового файла
Отправка сигнала завершения в
порт



Парадигмы программирования

Настройка порта

...

Чтение данных

...

Запись данных в буфер

...

Чтение данных из буфера

...

main

Отправка сигнала завершения

...

Анализ данных

...

Открытие текстового файла

...

Запись данных

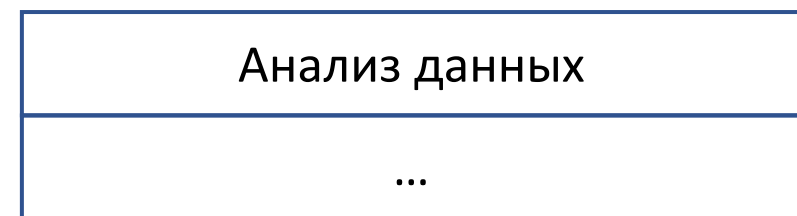
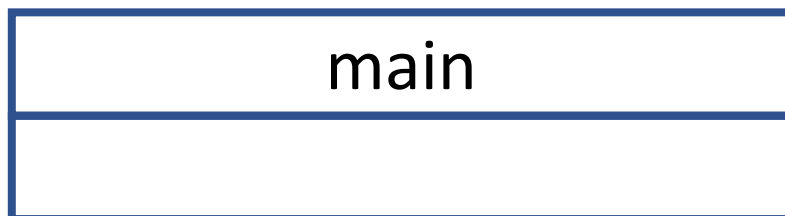
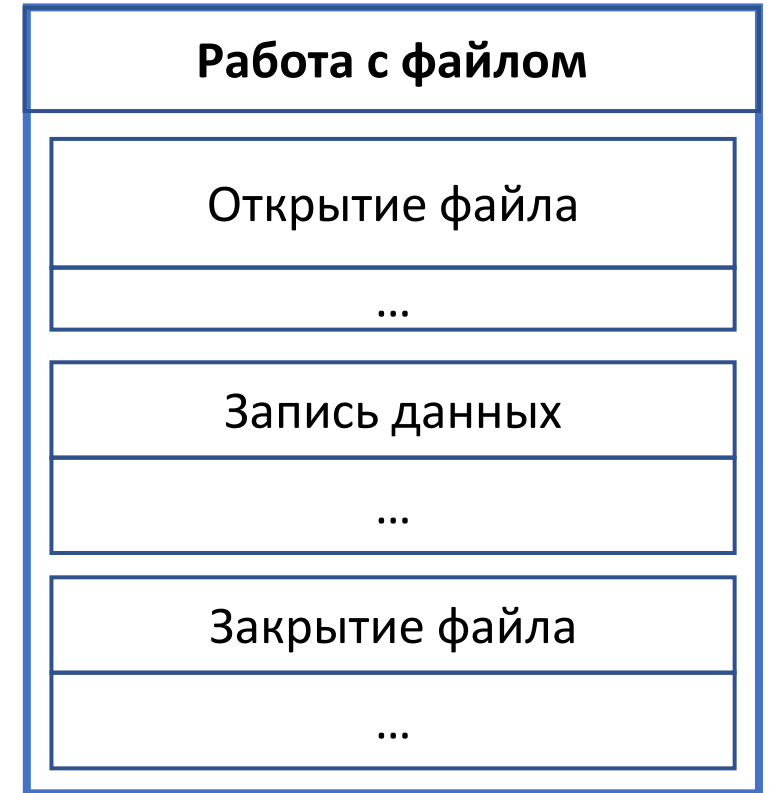
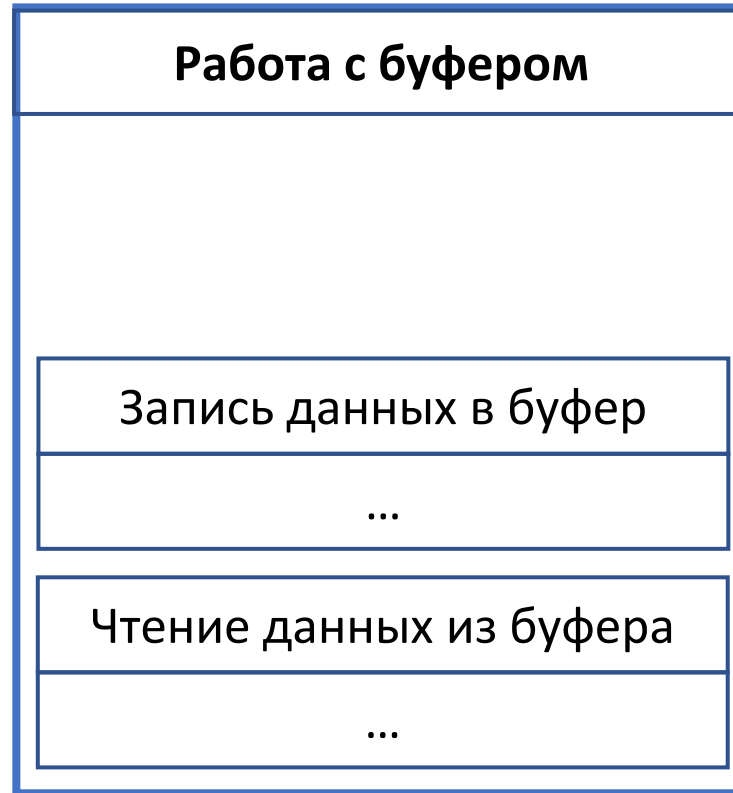
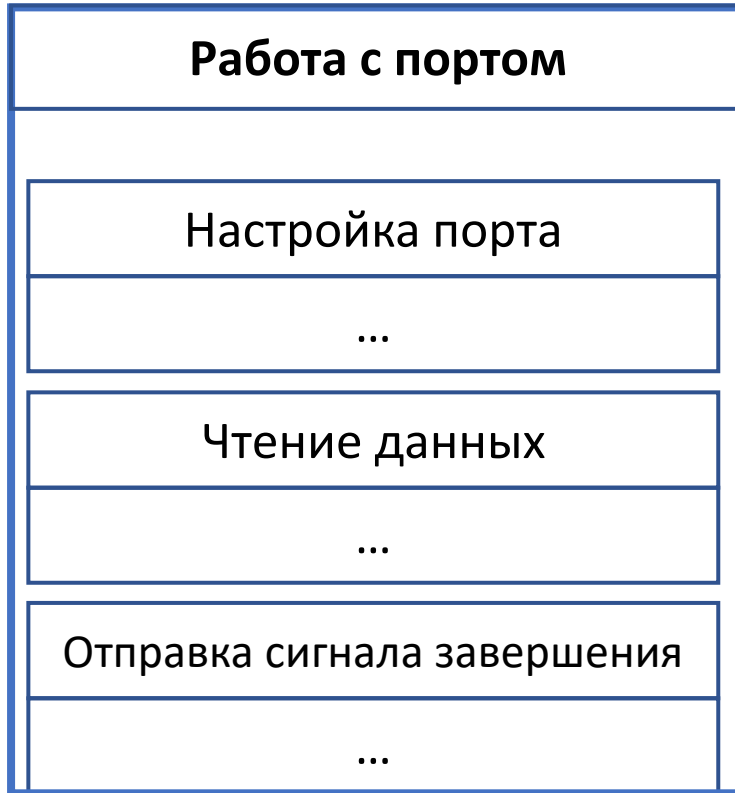
...

Закрытие текстового файла

...

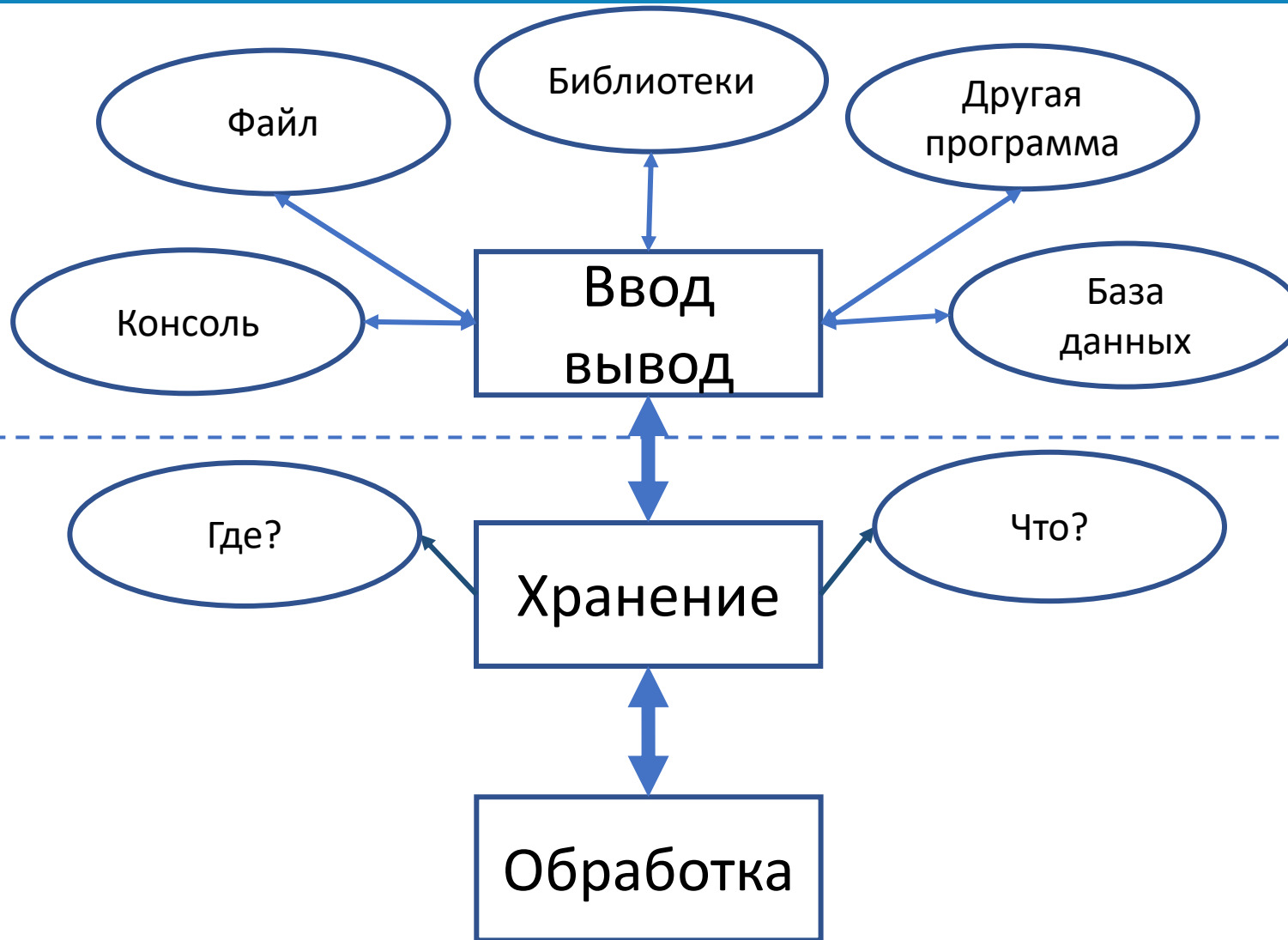


Парадигмы программирования



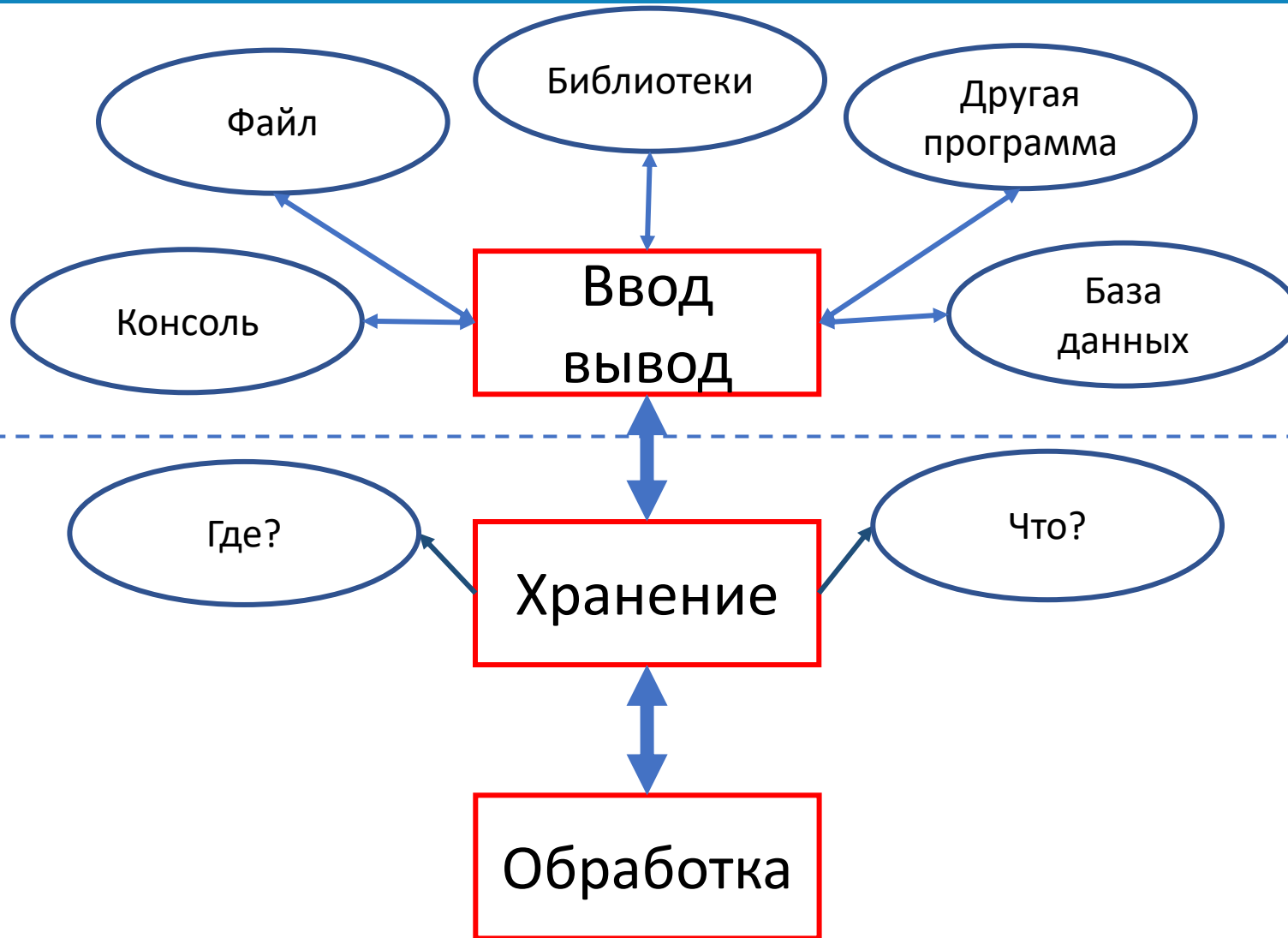


Дерево языка





Дерево языка





Практическая часть



Работа с командной строкой

```
sab@SAB: .../Lesson1$
```

| № | Команда | Описание | Пример |
|----|---|--|--|
| 0 | man | Описание работы команды | man ls |
| 1 | pwd | Показать текущее местонахождение | ~/SAB\$ pwd /home/user/SAB |
| 2 | ls | Позволяет просмотреть содержимое текущего каталога | ~/SAB\$ ls 1 1.txt |
| 3 | cd <путь к директории> | Перейти в другую директорию | ~\$ cd ~ /SAB/2 (полный путь) или ~/SAB\$ cd 2 (короткий путь) |
| 4 | mkdir <название директории> | Создание директории | ~/SAB\$ mkdir 1 |
| 5 | touch <название файла> | Создание файла | ~/SAB\$ touch 1.txt |
| 6 | nano <название файла> | Редактирование файла | ~/SAB\$ nano 1.txt |
| 7 | cp <что_копировать куда_копировать> | Копирование файла | ~/SAB/1\$ cp 1.txt ~/SAB/2 |
| 8 | cp -r <путь_к_папке путь_к_новому_месту> | Копирование директории | ~/SAB/1\$ cp -r 1 ~/SAB/2 |
| 9 | mv <что_переместить куда_переместить> | Переместить файл | ~/SAB/1\$ mv 1.txt ~/SAB/2 |
| 10 | rm <название файла> | Удалить файл | ~/SAB/1\$ rm 1.txt |
| 11 | rm -r <название файла> | Удалить директорию | ~/SAB/1\$ rm -r 1 |



Перенаправление ввода и вывода

Ввод и вывод распределяется между тремя стандартными потоками:

- **stdin** — стандартный ввод (клавиатура), - 0
- **stdout** — стандартный вывод (экран), - 1
- **stderr** — стандартная ошибка (вывод ошибок на экран). - 2

< file — использовать файл как источник данных для стандартного потока ввода.

> file — направить стандартный поток вывода в файл (перезапись)

2> file — направить стандартный поток ошибок в файл (перезапись)

>>file — направить стандартный поток вывода в файл (добавление)

2>>file — направить стандартный поток ошибок в файл. (добавление)

&>file или **>&file** — направить с.п. вывода и с.п. ошибок в файл.



Перенаправление ввода и вывода

- < file** — использовать файл как источник данных для стандартного потока ввода.
- > file** — направить стандартный поток вывода в файл (перезапись)
- 2> file** — направить стандартный поток ошибок в файл (перезапись)
- >>file** — направить стандартный поток вывода в файл (добавление)
- 2>>file** — направить стандартный поток ошибок в файл. (добавление)
- &>file** или **>&file** — направить с.п. вывода и с.п. ошибок в файл.

```
sab@SAB: /$ ps > 1.txt  
sab@SAB: /$ cat 1.txt  
sab@SAB: /$ ps >> 1.txt  
sab@SAB: /$ ps qq > 1.txt  
sab@SAB: /$ ps qq 2> 1.txt
```




find

find – утилита, с помощью которой возможно найти файл по его имени

```
find [адрес начала поиска] ... [выражение]
```

Вывести все файлы с информацией о них

```
sab@SAB: /$ find . -printf '%M %n %s %Tb %p\n'
```

Вывести все файлы, размером более 100 МБ, но менее 2ГБ

```
sab@SAB: /$ find . -size +100M -size -2G
```

Вывести все файлы, изменённых не ранее 30 и не позднее 20 минут назад

```
sab@SAB: /$ find . -type f -mmin +20 -mmin -30
```



Grep

Grep (*global regular expression printer*) – утилита командной строки, позволяющая производить поиск строки в файле.

```
grep [ключи] шаблон [ имя_файла ... ]
```

| Ключ | Описание |
|------|---|
| -c | Выдает только количество строк, содержащих выражение. |
| -h | Скрывает вывод названия файла, в котором было обнаружено вхождение. Используется при поиске по нескольким файлам. |
| -i | Игнорирует регистр символов при поиске. |
| -l | Выдает только имена файлов, содержащих сопоставившиеся строки. |
| -n | Выдает перед каждой строкой ее номер в файле (строки нумеруются с 1). |
| -s | Скрывает выдачу сообщений о не существующих или недоступных для чтения файлах. |
| -v | Выдает все строки, за исключением содержащих выражение. |
| -E | Поиск с использованием регулярных выражений |
| -o | Вывод только обнаруженных символов |
| -r | Рекурсивный поиск |



Grep (примеры)

```
sab@SAB: /$ grep Hello file.txt
```

Ищем строку ***Hello*** в файле ***file.txt***

```
sab@SAB: /$ grep -r Hello .
```

Ищем строку ***Hello*** во ***всех*** файлах текущей директории

```
sab@SAB: /$ grep -c Hello file.txt
```

Ищем число вхождений строки ***Hello*** в файле ***file.txt***



Grep (пример с директориями)

Задача:

Необходимо быстро найти пароль от телефона среди множества других паролей.

```
sab@SAB: /$ ./script.sh 5 5 5 100
```

Генерируем 5 директорий, в каждой 5 поддиректорий, в каждой 5 файлов, в каждом из которых 100 строк. В одной из них содержится строка: Password_phone:XXXXXX

```
sab@SAB: /$ grep -R "Password_phone" ./Files
```

Производим поиск по директории *Files*

```
sab@SAB: /$ rm -rf Files/
```

Удаляем созданные директории



Регулярные выражения

Регулярные выражения - инструмент для поиска текста по шаблону.

| Метасимвол | Описание работы |
|------------|--|
| \ | начало буквенного спецсимвола |
| ^ | указывает на начало строки |
| \$ | указывает на конец строки |
| * | указывает, что предыдущий символ может повторяться 0 или больше раз |
| + | указывает, что предыдущий символ должен повториться больше один или больше раз |
| ? | предыдущий символ может встречаться ноль или один раз |
| {n} | указывает сколько раз (n) нужно повторить предыдущий символ |
| {N,n} | предыдущий символ может повторяться от N до n раз |
| . | любой символ кроме перевода строки |
| [az] | любой символ, указанный в скобках |
| x y | символ x или символ y |
| [^az] | любой символ, кроме тех, что указаны в скобках |
| [a-z] | любой символ из указанного диапазона |
| [^a-z] | любой символ, которого нет в диапазоне |
| [:alpha:] | является алфавитным символом |
| [:digit:] | является числом |



Регулярные выражения (примеры)

Поиск содержимого файлов, начинающихся с символов А или В

```
sab@SAB: /$ grep -re '^[AB]' .
```

Поиск количества строк в каждом файле, содержащие подряд две буквы «в»

```
sab@SAB: /$ egrep -rc "[v]{2}" .
```

Поиск содержимого файлов, содержащих слова «Вы или вы»

```
sab@SAB: /$ grep -re '[Bb]ы' .
```

Поиск в файле IPv4 адресов *(для любителей реальной практики)*

```
sab@SAB: /$ grep -E '[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}' ip.txt
```



Задача на подумать...

+2 бонусных
балла

Поиск в файле IPv4 адресов (*для любителей реальной практики*)

```
sab@SAB: /$ grep -E '[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}' ip.txt
```

Необходимо ввести ограничение на *ip* адреса

Адреса:

| | | |
|-----------------|---|--------------------------|
| 123.321.234.712 | - | (>255) |
| 999.999.999.999 | - | |
| 192.168.5.5 | + | |
| 10.0.0.4 | + | |
| 10.000.000.04 | - | (не более 1 нуля подряд) |



cut

cut – утилита, с помощью которой возможно вырезать символы из каждой переданной ей строки

```
cut [ключи] ... [ имя_файла ... ]
```

| Ключ | Описание |
|------|---|
| -c | Вырезать указанную последовательность символов |
| -d | Указать символ разделения (по умолчанию TAB) |
| -f | Выбрать символы, разделенные определенным символом |
| -s | Указание на пропуск любой строки, в которой нет разделителя |

Форматы для задания списка полей или колонок:

- A-B Поля или колонки от A до B включительно
- A- От поля или колонки A до конца строки
- B С начала строки до поля или колонки B
- A, B Поля или колонки A и B



cut (примеры)

Вывести первые пять символов файла

```
sab@SAB: /$ cut -c 1-5 1.txt
```

Выделить второй столбец из строки, символ разделения - двоеточие

```
sab@SAB: /$ cut -d ':' -f 2 1.txt
```

Выделить первое слово из текста (символ разделения - пробел), если их нет в строке – строка пропускается

```
sab@SAB: /$ cut -d ' ' -f1 -s 1.txt
```



tr

tr (translate) – утилита командной строки, позволяющая посимвольно обрабатывать текст

tr [ключи]... набор1 [набор2]

| Ключ | Описание |
|-----------|---|
| -c | Оставить только символы из первого набора, остальные заменить на значение из второго набора |
| -d | Удалить символы, входящие в первый набор |
| -s | Заменяет все символы из первого набора, встречающиеся несколько раз подряд на одиночное вхождение |



tr (примеры)

Заменить все строчные символы на прописные

```
sab@SAB: /$ tr "a-z" "A-Z" < 1.txt
```

```
sab@SAB: /$ tr "[[:lower:]]" "[[:upper:]]" < 1.txt
```

Заменить все символы, кроме a-z, \n на символ X

```
sab@SAB: /$ tr -c "a-z\n" X < 1.txt
```

Удалить все строчные символы

```
sab@SAB: /$ tr -d "\n" < 1.txt
```

Заменить все повторяющиеся пробелы на один

```
sab@SAB: /$ tr -s " " < 1.txt
```



WC

wc (word count) – утилита командной строки, выводящая число переводов строк, слов и байт для каждого указанного файла и итоговую строку, если было задано несколько файлов.

wc [ключи]... [имя_файла ...]

| Ключ | Описание |
|------|-------------------------------------|
| -c | Вывести размер файла в байтах |
| -m | Вывести количество символов в файле |
| -l | Вывести количество строк в файле |
| -w | Вывести количество слов в файле |

Вызов без параметров вернёт все значения количества строк, слов и символов в файле

Вывести число строк в файле

sab@SAB: /\$ wc -l file.txt



uniq

uniq – утилита командной строки, предназначена для поиска одинаковых строк в массивах текста.

uniq [OPTION]... [INPUT [OUTPUT]]

| Ключ | Описание |
|-----------|---|
| -u | Вывести исключительно те строки, у которых нет повторов. |
| -d | Вывести все строки, удаляя дубликаты |
| -D | Вывести только повторяющиеся строки. |
| -c | В начале каждой строки вывести число, которое обозначает количество повторов. |

Вывести число уникальных строк в файле

```
sab@SAB: /$ uniq -u file.txt
```



awk

awk – скриптовый язык построчного разбора и обработки входного потока

awk [ключи] '[шаблон] {действие}'

Утилита awk последовательно применяется к каждой из строчек. Для фильтрации строк применяется шаблон, накладывающий ограничение на отбираемые строки

Расширенный вариант *awk*

BEGIN {действие} -> Выполняется до обработки строк

шаблон {действие}

шаблон {действие}

END {действие} -> Выполняется после обработки строк

| Ключ | Описание |
|-------------------|---|
| -F fs | Указать символ разделения (по умолчанию пробел) |
| -v var=val | Задать значение переменной |



awk

Пользователь может использовать встроенные переменные или создавать свои.

Awk рассматривает переменную как строковую. По умолчанию строка инициализируется "0" или пустой строкой

Типы переменных:

- позиционные
- числа с плавающей точкой
- строка символов
- массив

Утилита awk поддерживает стандартные конструкции языка C – ветвления, циклы

| Основные встроенные переменные | Описание |
|--------------------------------|--|
| \$0 | Значение обрабатываемой строки |
| \$1 - \$N | Значение определенного столбца, полученных в результате деления строки сепаратором |
| FS | Разделитель полей записи на вводе |
| NF | Число полей в текущей записи |
| NR | Номер записи (общее число считанных записей) |

| Некоторые команды | Описание |
|-------------------|------------------------|
| print | Вывести строку целиком |
| length(N) | Длина N в символах |



awk (примеры)

Все примеры производятся над файлом, содержащим значения ФИО студентов, их дату рождения и оценку за экзамен

```
Ivanov Ivan Ivanovich 15-04-1995 87
Petrov Sergey Aleksandrovich 23-09-1992 78
Smirnov Olga Nikolaevna 07-06-1990 92
Kuznetsov Dmitry Anatolyevich 12-11-1997 89
Popov Elena Ivanovna 02-03-1988 94
Sokolova Anna Alekseevna 19-07-1992 76
.....
```




awk (примеры)

Напечатать весь текст

```
sab@SAB: /$ awk '{print}' test.txt
```

Напечатать сообщение приветствия и прощания между текстом

```
sab@SAB: /$ awk 'BEGIN {print "Hello"} {print} END {print "Good Bye"}' test.txt
```

Напечатать первый и второй столбец текста (Фамилию и имя)

```
sab@SAB: /$ awk '{print $1, $2}' test.txt
```

Напечатать целиком все строки, содержащие символ F

```
sab@SAB: /$ awk '/F/{print $0}' test.txt
```

Напечатать второй столбец текста, символ разделитель - тире

```
sab@SAB: /$ awk -F- '{print $2}' test.txt
```

Вывести число строк в файле

```
sab@SAB: /$ awk 'END{print NR}' test.txt
```



awk (примеры)

Вывести сумму значений 5го столбца

```
sab@SAB: /$ awk '{sum+=$5} END{print sum}' test.txt
```

Вывести все строки, где пятый столбец равен 90

```
sab@SAB: /$ awk '{if ($5==90) print $0}' test.txt
```

Вывести все строки, где пятый столбец равен 90 и вывести их количество

```
sab@SAB: /$ awk '{if ($5==90) {L+=1; print $0}} END{print L}' test.txt
```

Вывести сообщение о результате аттестации в случае получения оценки

```
sab@SAB: /$ awk '{if ($5>=90) {print $1 " " $2 ": Zachet"} else {print $1 " " $2 ": Peresdacha"}}' test.txt
```

Посчитать количество символов в каждом слове файла

```
sab@SAB: /$ awk '{ for(i=1; i<=NF; i++) {L+=length($i)}} END {print L}' test.txt
```

Пояснение: `for(i=1; i<=NF; i++)` – стандартный цикл. При обращении к `$i` получаем значения всех столбцов по очереди с первого по последний.



Pipes (Каналы)

Каналы используются для перенаправления потока из одной программы в другую.

```
sab@SAB: /$ ps | grep p
```

```
sab@SAB: /$ ps > 1.txt; ls >> 1.txt; cat 1.txt | grep a
```



Несколько полезных команд Linux

hexdump — показывает шестнадцатеричное представление данных, поступающих на стандартный поток ввода.

cat — считывает данные со стандартного потока ввода и передает их на стандартный поток вывода.

sudo - запуск программы от имени других пользователей, а также от имени суперпользователя.

bc — калькулятор

python3 — среда разработки Python

sort — сортировка переданных значений (-r — обратный порядок)

```
sab@SAB: /$ echo "10*10" | bc
```

```
sab@SAB: /$ hexdump 1.txt
```



Потренируемся

1. Откройте терминал Linux
2. Создайте директорию. Назовите ее «Task1»
3. Войдите внутрь директории
4. Создайте внутри нее еще две директории – «Task1_1» и «Task1_2»
5. Войдите внутрь директории Task1_1
6. Создайте внутри ее файл с названием «File_1» и еще одну директорию «Task1_1_1»
7. Откройте созданный файл и запишите туда любую информацию. Закройте его
8. Скопируйте «File_1» и «Task1_1_1» в директорию «Task1_2»
9. Опуститесь на уровень ниже и удалите директорию Task1_1
10. Из данной директории переместите «File_1» в директорию «Task1»



Потренируемся

Задание 1.

+3 бонусных
балла

1. Перейдите в директорию lab1.
2. Оставаясь в директории lab1, создайте в каталоге poems/English файл, содержащий текст вашего любимого стихотворения отечественного автора. Название файла должно соответствовать названию стихотворения. Внутри файла, перед текстом произведения укажите название и автора.
3. Перенесите созданный файл из директории English в Russian.
4. Создайте в директории English каталоги, содержащие названия веков и распределите по ним расположенные в ней стихотворения.

Задание 2.

1. Произведите поиск всех стихотворений, названия которых содержит только кириллицу
2. Найдите все файлы с расширением jpeg
3. Найдите все файлы, которые были изменены за последние 20 минут
4. Найдите все файлы, объемом больше 500 Кб

Задание 3.

1. Вычислите у скольких стихотворений вместо названия стоят “***”
2. Выведите напротив каждого файла сообщение о том, содержит ли он восклицательный знак
3. Рассчитайте, сколько раз в тексте стихотворений встречается предлог «на»
4. Вычислите самое часто встречающееся слово в монологе Гамлета



Спасибо за внимание!