Динамические структуры данных

Владимир Агаджанов, КА «Тор», 2023

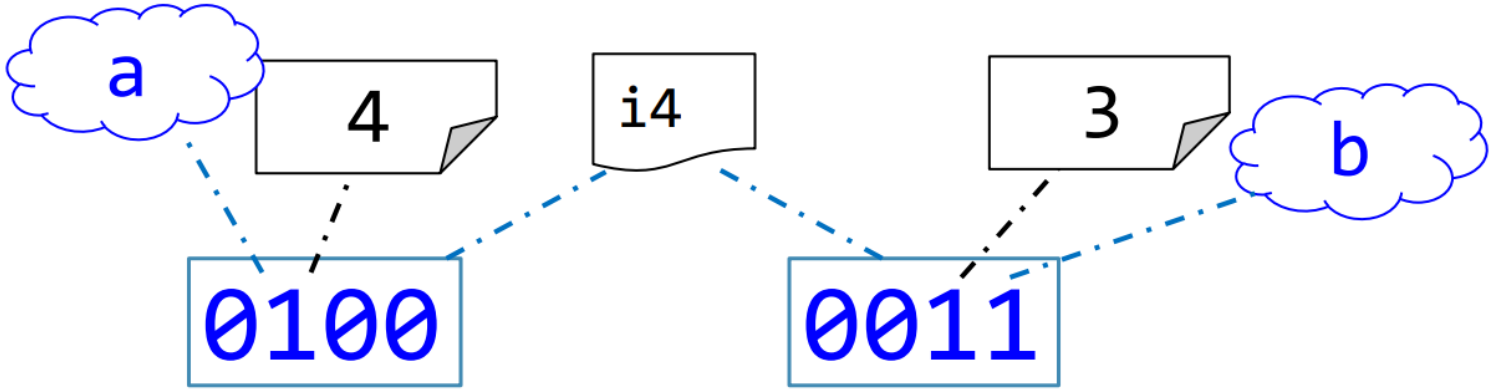
Динамические структуры данных – это **структуры данных, память под которые выделяется и освобождается по мере необходимости в рантайме**. Динамические структуры данных в процессе существования в памяти могут изменять не только число составляющих их элементов, но и характер связей между элементами.

Базовые операции:

- Доступ к элементам
- Добавление элементов
- Удаление элементов
- Предоставление текущего размера коллекции

RAMM

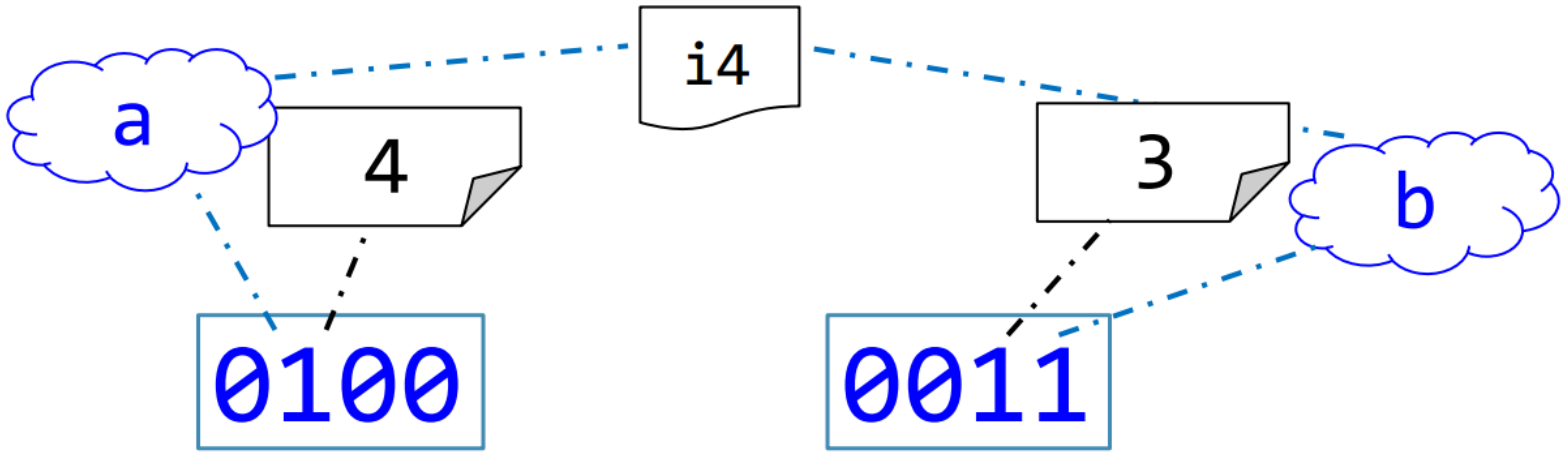
Random Access Memory Model (модель памяти с рандомным доступом) –
принятая в C/C++ модель памяти, представляющая собой непрерывную область памяти
с возможностью адресовать каждую ячейку памяти.



Тип закрепляется за значением?

RAMM

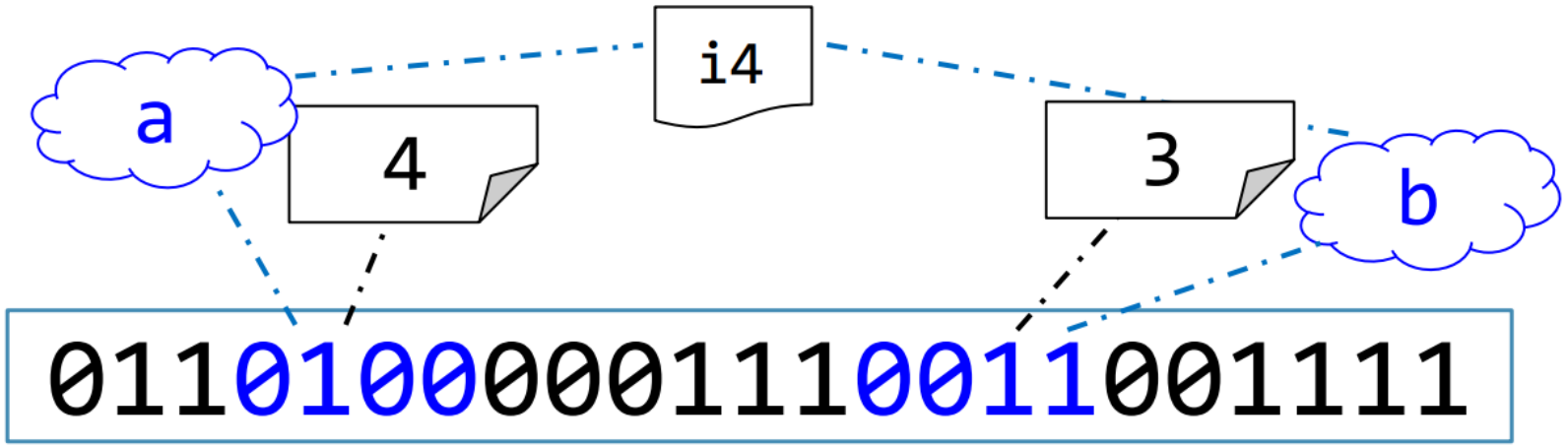
Random Access Memory Model (модель памяти с рандомным доступом) –
принятая в C/C++ модель памяти, представляющая собой непрерывную область памяти
с возможностью адресовать каждую ячейку памяти.



Тип в C++ закреплен за именем, но никак не за памятью под этим именем

RAMM

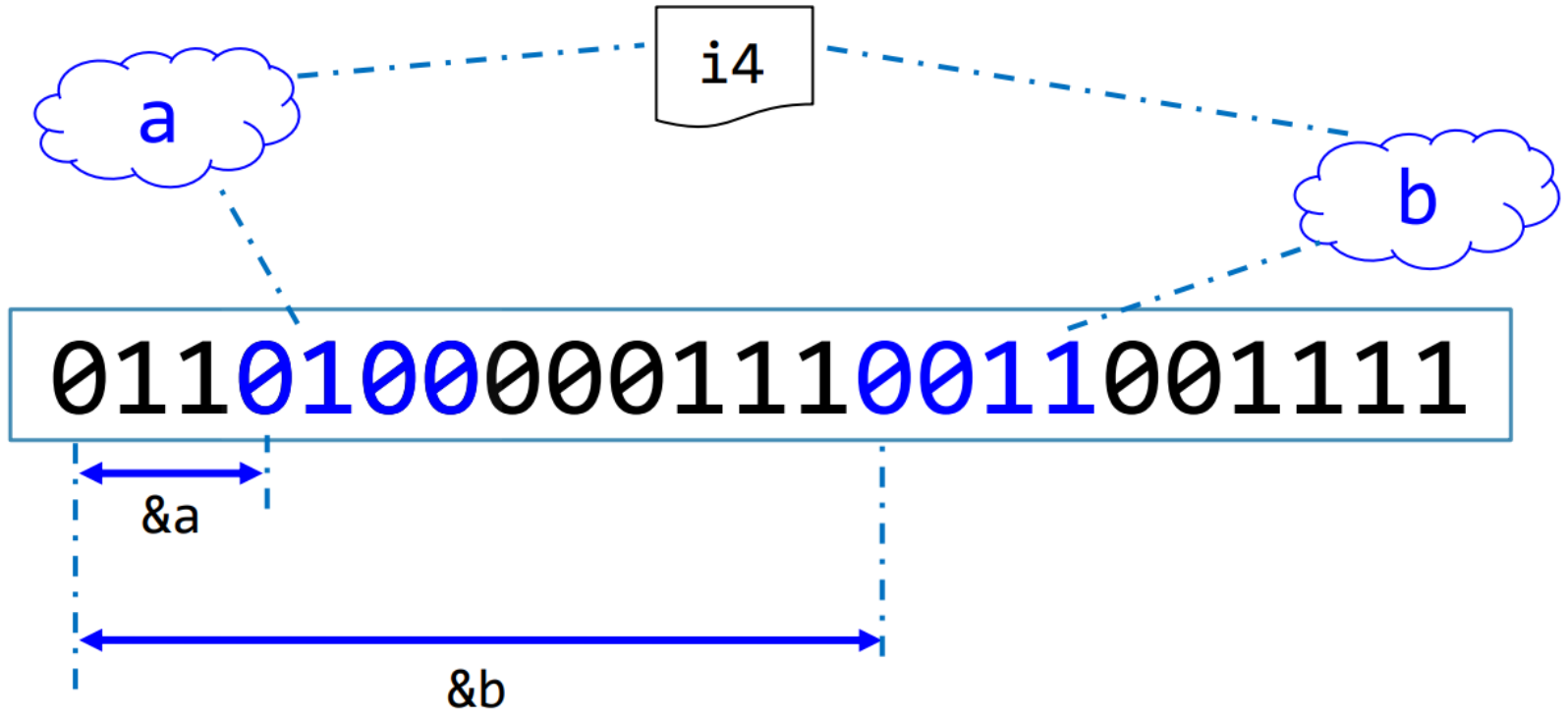
Random Access Memory Model (модель памяти с рандомным доступом) –
принятая в C/C++ модель памяти, представляющая собой непрерывную область памяти
с возможностью адресовать каждую ячейку памяти



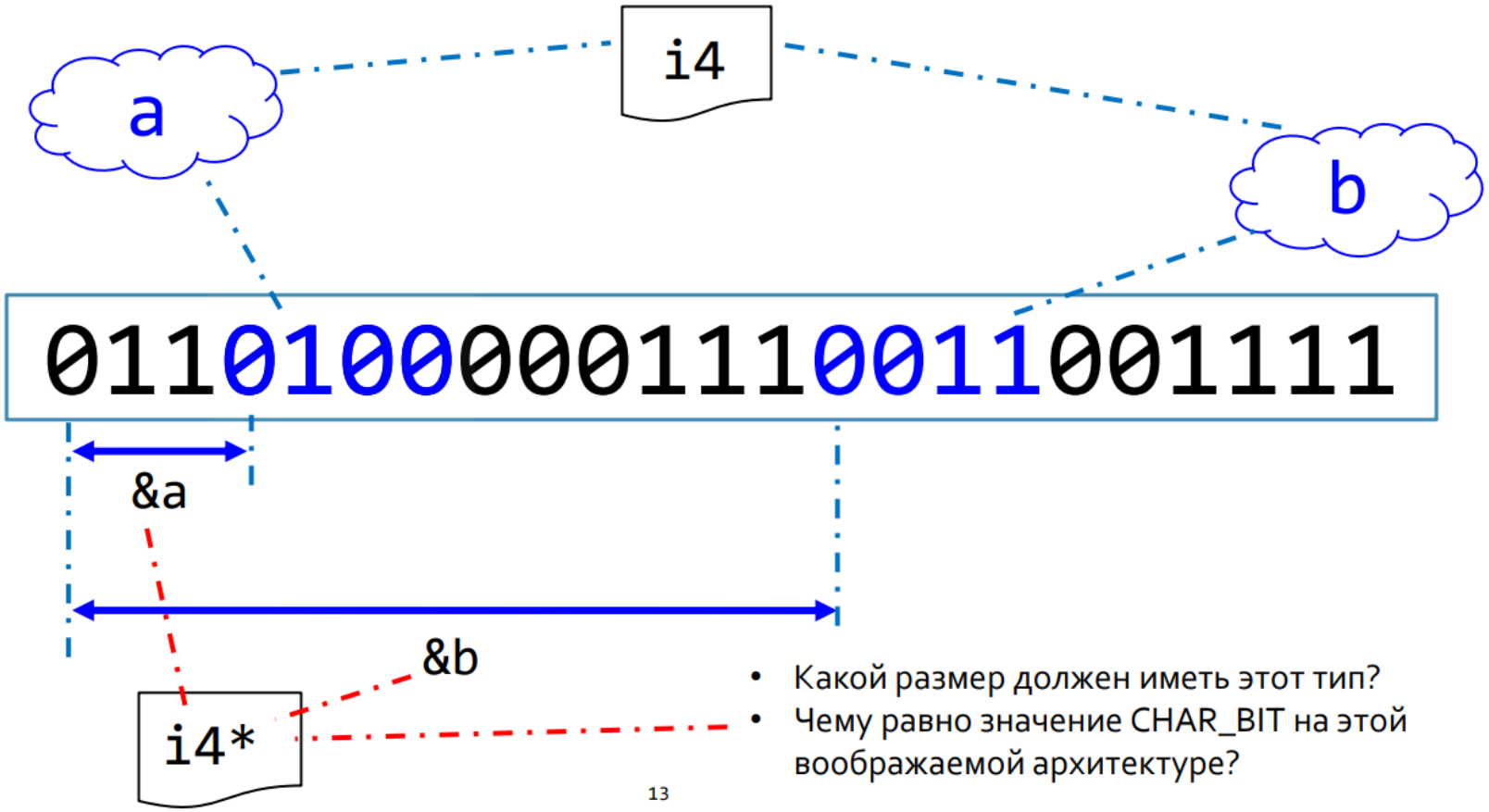
Здесь только нули и единицы, no more…

RAMM

Смещение в памяти от нуля до начала данных под именем называется адресом.



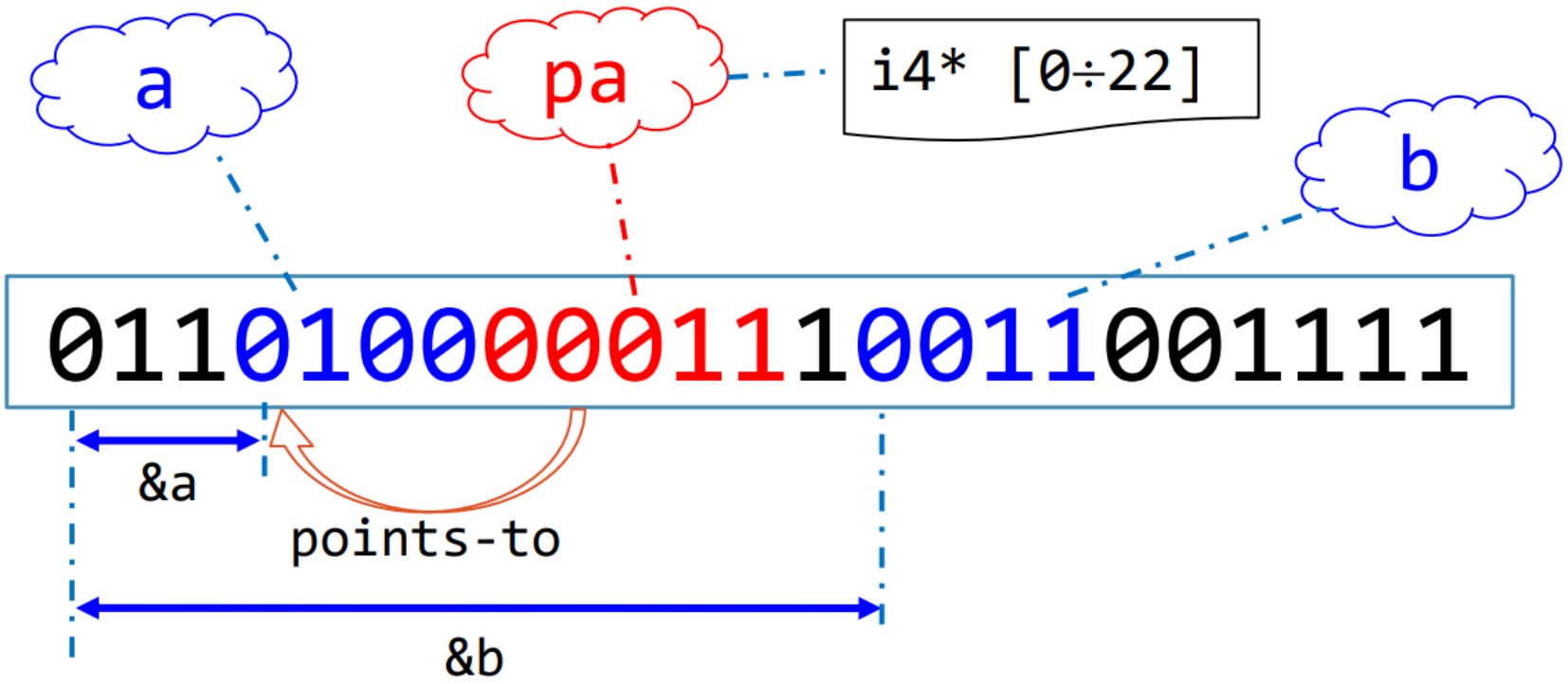
Указатель как и любая другая переменная имеет место в памяти,
в которой и располагается адрес куда он указывает.



Какова разрядность архитектуры у машины с картинки учитывая побитовую адресацию и последний адрес 23?

RAMM

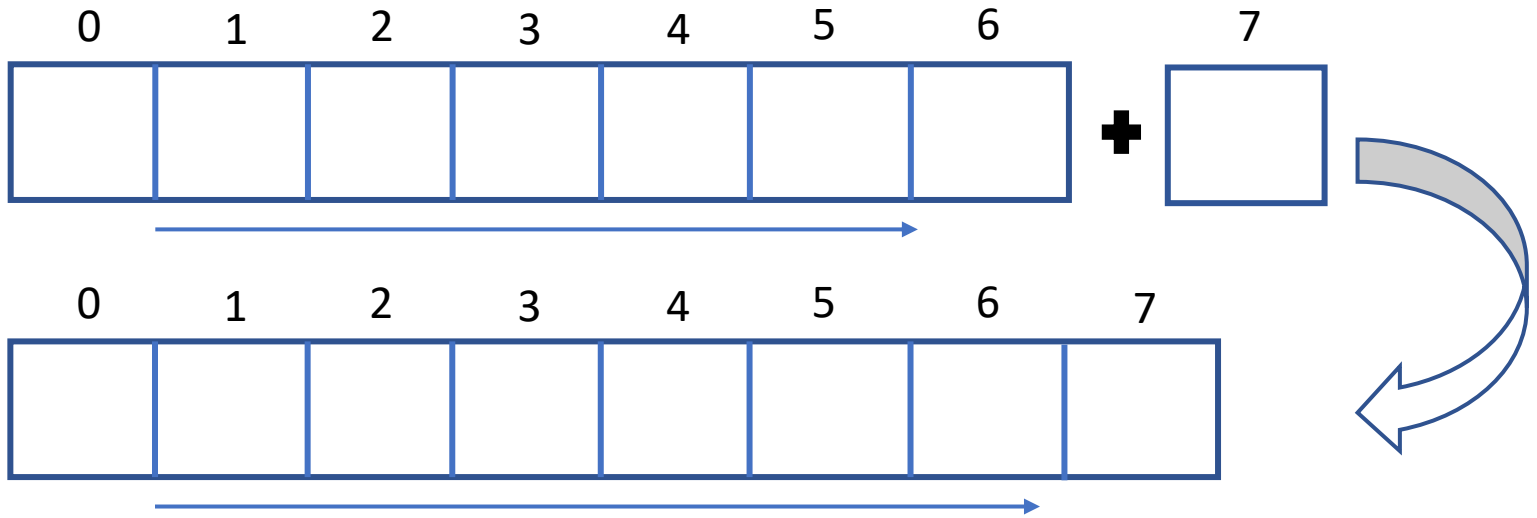
Тип i4* имеет размер 5 бит, поскольку именно столько нужно, чтобы закодировать число от 0 до 23.



Где на картинке находится i4 * pb?

Dynamic Array

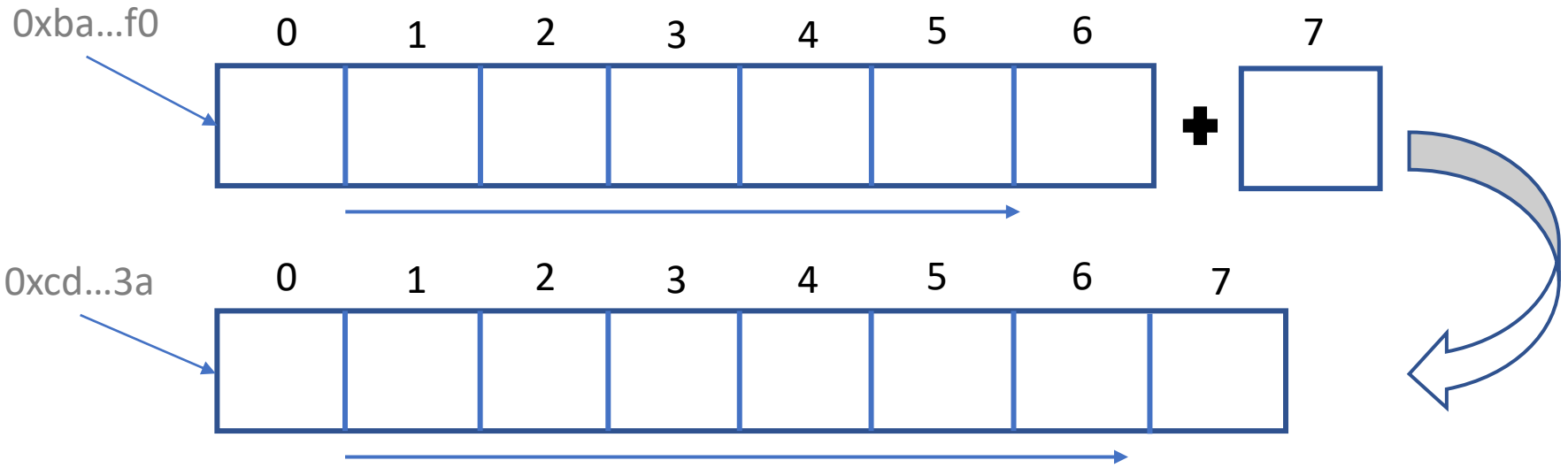
Структура данных переменной длинны, данные которой хранятся в непрерывной области памяти.
Инкапсулирует логику изменения размера внутри себя и защищает инварианты этой логики



Какие операции самые дешевые, а какие самые дорогие?

Dynamic Array

Структура данных переменной длинны, данные которой хранятся в непрерывной области памяти.
Инкапсулирует логику изменения размера внутри себя и защищает инварианты этой логики.



Какие операции самые дешевые, а какие самые дорогие?

Реаллоцирует данные в выделенной области при переполнении в другую область памяти большего размера!

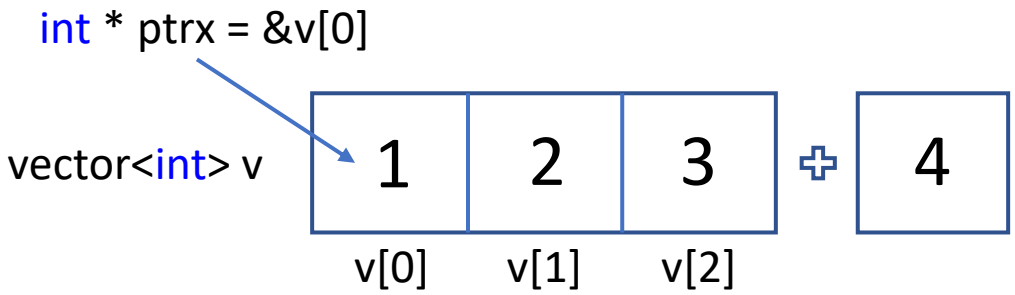
Видите ли вы тут ошибку?

```
#include <vector>
#include <iostream>
using namespace std;

void print(int * p, int size) {
    for(;size > 0; --size) {
        cout << *(p++) << " ";
    }
}

int main() {
    vector<int> v {1, 2, 3};
    int * ptrx = &v[0];

    v.push_back(4);

    print(ptrx, v.size());

    return 0;
}
```

godbolt.org

Видите ли вы тут ошибку?

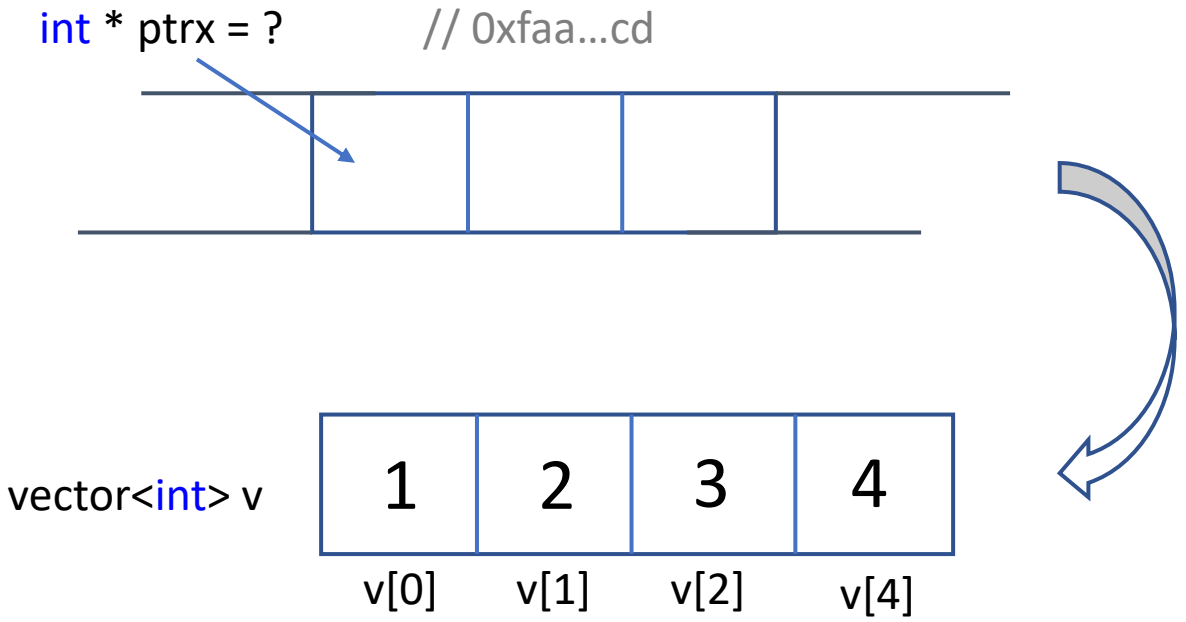
```
#include <vector>
#include <iostream>
using namespace std;

void print(int * p, int size) {
    for(;size > 0; --size) {
        cout << *(p++) << " ";
    }
}

int main() {
    vector<int> v {1, 2, 3};
    int * ptrx = &v[0];

    v.push_back(4);

    print(ptrx, v.size());

    return 0;
}
```



godbolt.org

Видите ли вы тут ошибку?

```
#include <vector>
#include <iostream>
using namespace std;

void print(int * p, int size) {
    for(;size > 0; --size) {
        cout << *(p++) << " ";
    }
}

int main() {
    vector<int> v {1, 2, 3};
    int * ptrx = &v[0];

    v.push_back(4);

    print(ptrx, v.size());

    return 0;
}
```



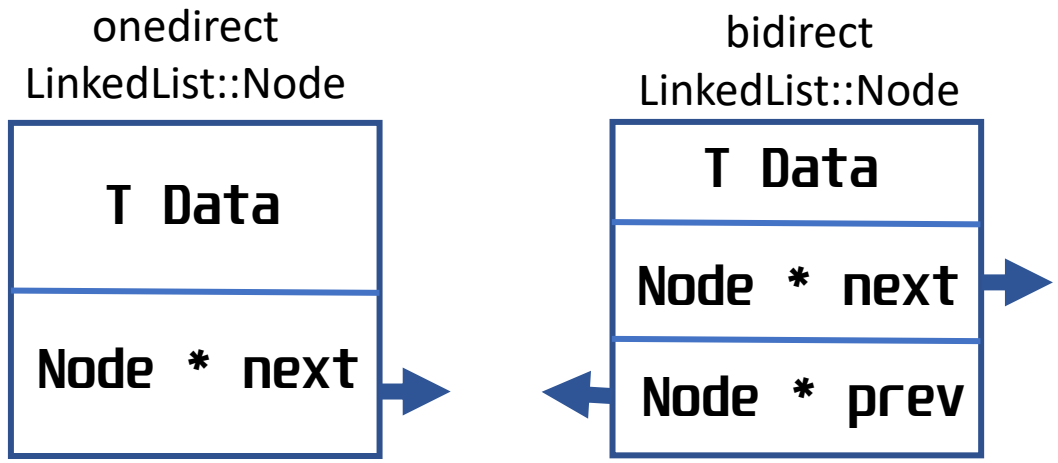
godbolt.org

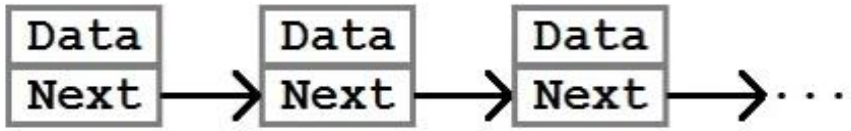
Linked List

Динамическая структура данных, состоящая из элементов, содержащих помимо собственных данных ссылки (links) на следующий (и) предыдущий элемент списка.

Выделяют однонаправленные и двунаправленные списки, подразумевая направление адресации только в одну или в обе стороны.

Хранит внутри себя только указатель на первый последний элемент, которые расположены в памяти в хаотичном порядке, в отличие от массива.





Какие преимущества и недостатки списка по сравнению с динамическим массивом?

godbolt.org

Stack

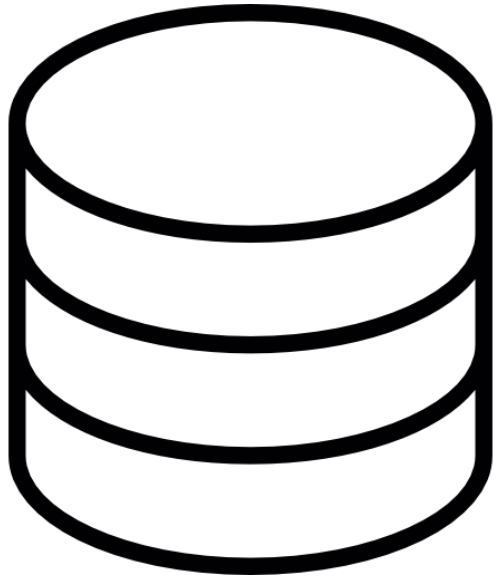
Динамическая структура данных, представляющая из себя упорядоченный набор элементов,
в которой добавление новых элементов и удаление существующих производится с одного конца,
называемого вершиной стека.

LIFO (Last In First Out), т.е. последним пришёл - первым ушёл.

Запрещенные операции:

- Добавление элемента в произвольное место
- Удаление произвольного элемента
- Просмотр произвольного элемента

Типичные примеры стека в жизни?



godbolt.org

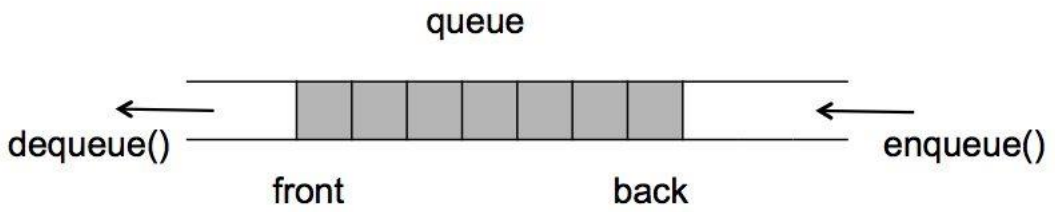
Queue

Последовательный набор элементов с переменной длиной. Добавление элементов в очередь происходит с одной стороны, а удаление — с другой.

FIFO (First In First Out), т.е. первым пришёл - первым ушёл.

Запрещенные операции:

- Добавление элемента в произвольное место в очереди
- Удаление произвольного элемента
- Просмотр произвольного элемента



Такое ощущение, что очередь концептуально похожа на список, но чем?

```
class Queue : public List {
public:
    Queue() : List()
    { }

    void at(int i) = delete;          // нельзя брать произвольный элемент
    void push_front() = delete;       // нельзя добавлять в начало
    void pop_back() = delete;         // нельзя выбирать с конца

    void enqueue(char c)
    {
        push_back(c);
    }

    char dequeue()
    {
        return pop_front();
    }
};
```

…А как реализовать стэк через список?

godbolt.org

```
class Stack : public List {
public:
    Stack() : List()
    { }

    void at(int i) = delete;          // нельзя брать произвольный элемент
    void push_back() = delete;        // нельзя добавлять в конец
    void pop_back() = delete;         // нельзя выбирать с конца

    void push(char c)
    {
        push_front(c);
    }

    char pop()
    {
        return pop_front();
    }
};
```

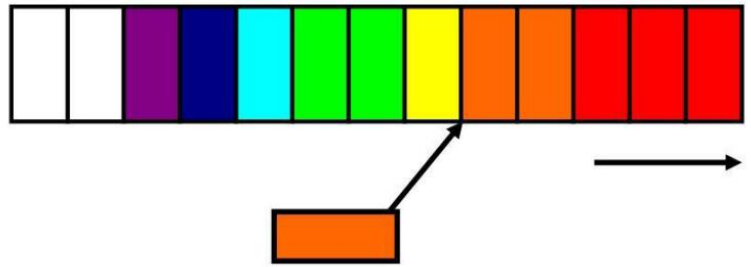
godbolt.org

Priority Queue

Очередь с приоритетным извлечением элементов, где приоритетом может выступать некоторая величина, заданная при создании элемента.

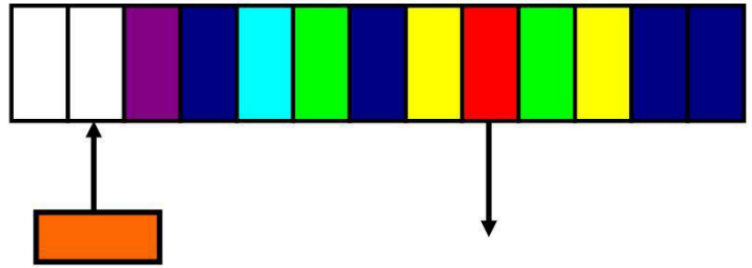
Элемент добавляется в очередь согласно своему приоритету (наибольшим к началу очереди).
При извлечении выбирается первый элемент из начала.

Приоритетное включение



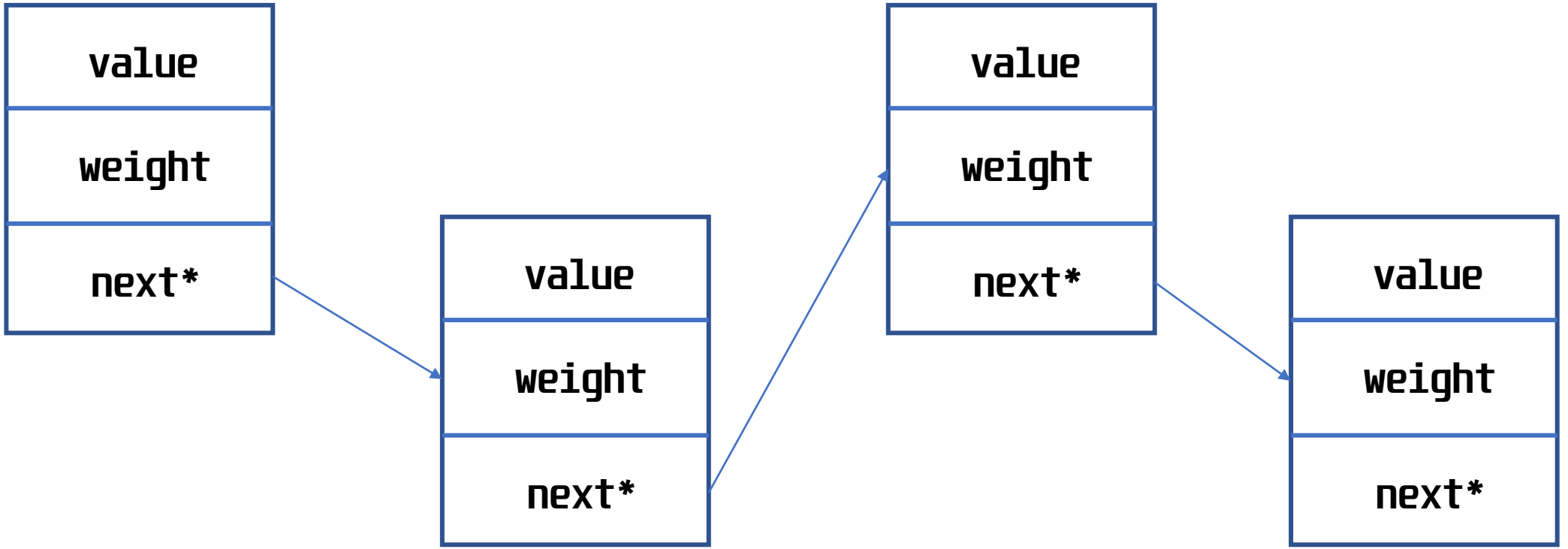
Элемент добавляется в конец очереди независимо от приоритета.
При извлечении выбирается наиболее приоритетный элемент.

Приоритетное исключение



Какая из видов очередей с приоритетом будет быстрее в общем случае?
Как будет выглядеть узел для такой очереди?

Узел очереди с приоритетом



Удачно ли спроектированы узлы очереди?

C++ за 21 день?

Представьте себе хирурга, который прочитал книжку «Хирургия за 21 день», представили?
А теперь он вас оперировать будет, хотите так?

Erik Meijer

- Стоит понимать, что мы находимся в начале **долгого** пути.
- Настраивайте себя на длинную дистанцию.
- Наверное можно механически выучить язык не научившись программированию на этом языке.
  Но это бессмысленно.

Мы только попробовали суп на запах

- Язык C++ выразителен и предлагает много возможностей.
- Увы, сколь C++ богат, столь он и сложен.
- Использовать его, не обладая глубоким пониманием происходящего часто бывает больно.
- В изучении упорство и усидчивость – ваши лучшие друзья,но не стесняйтесь обращаться за советом.