4. Отладка, гит
Memory layout

Программирование и алгоритмизация

Практические занятия

БИВТ-24-17

Надежда Анисимова

ms teams m2102039@edu.misis.ru

Проверка себя

1. Ссылка это?
2. Сработает ли такой код? Если да, то что означает?

```
int val = 50;
const int& ref = val;
```

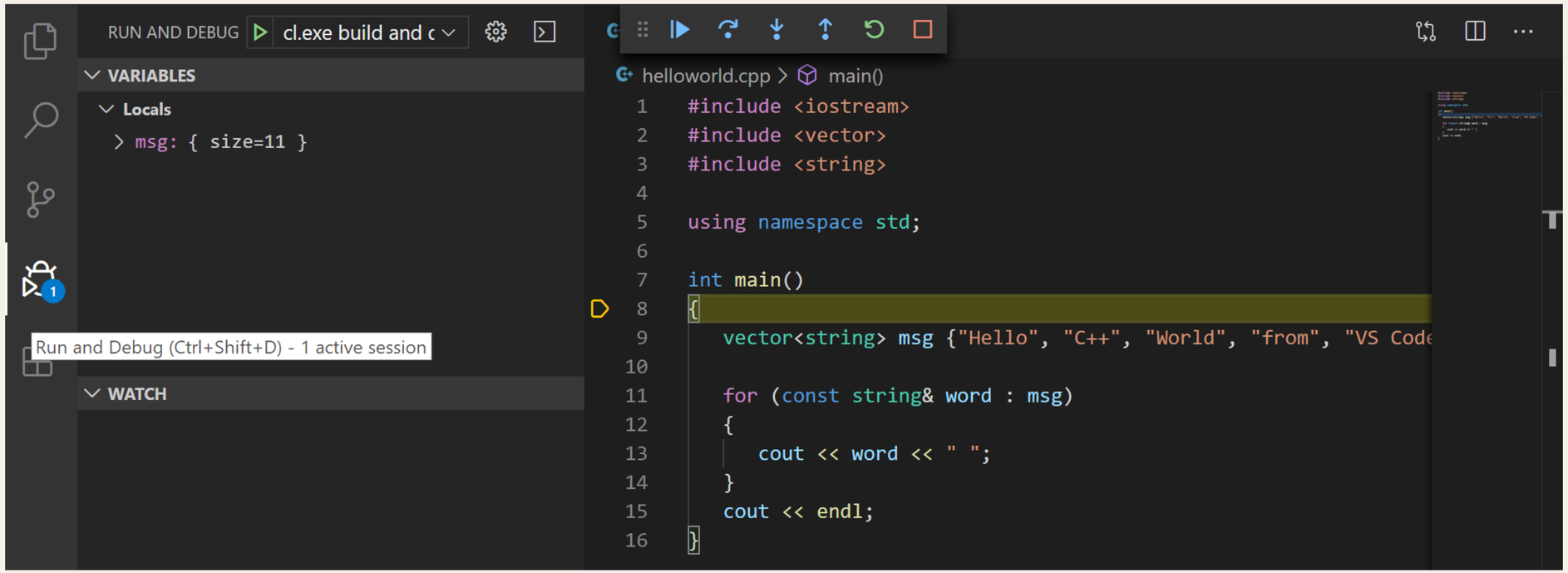
3. Три команды гита по порядку, чтобы изменения в удаленный репозиторий отправить?

Отладка

01

Отладка (или дебаггинг, debugging) — это процесс поиска и исправления ошибок или неполадок в программном коде

Сейчас большинство IDE имеют встроенный отладчик.



Как отлаживать?

- Устанавливать точки остановки (breakpoints) — это места, где программа временно останавливается, чтобы можно былопроверить состояние переменных

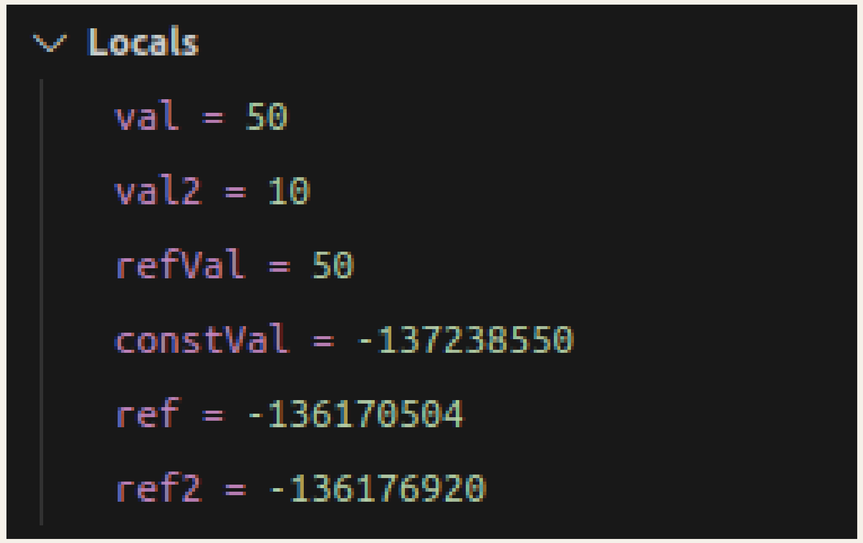
- Пошагово выполнять код

Шаг с обходом (step over)

Шаг с заходом (step into)

Шаг с выходом (step out)

- Просматривать состояние переменных



Git

02

Создание репозитория

Создать репозиторий локально
соединить с удаленным

- git init
- git remote add origin <ссылка .git>

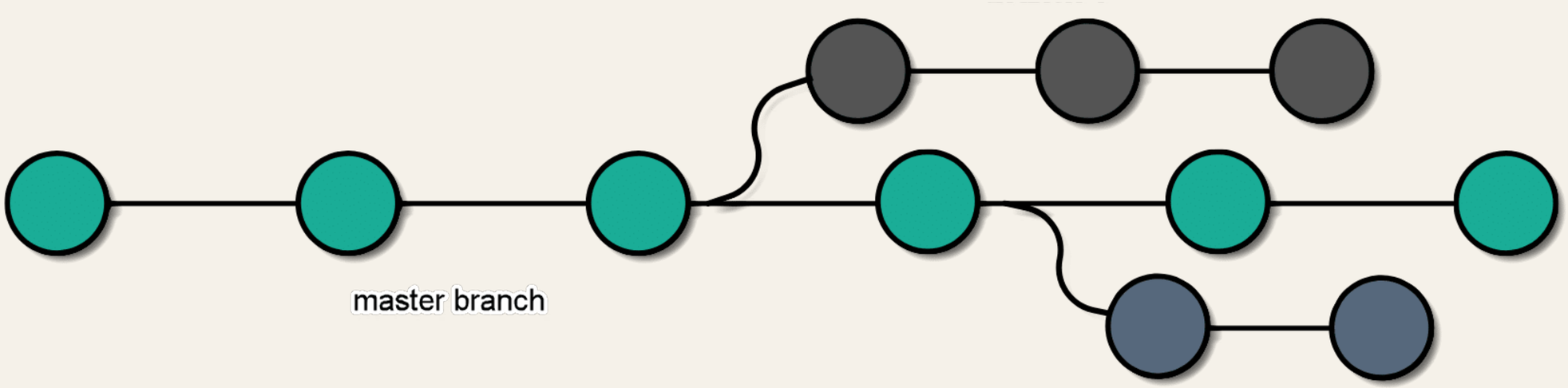
Запушить с флагом -u чтобы соединить локальную ветку и удаленную

git push -u origin main

-u
--set-upstream

Создать удаленно
клонировать локально

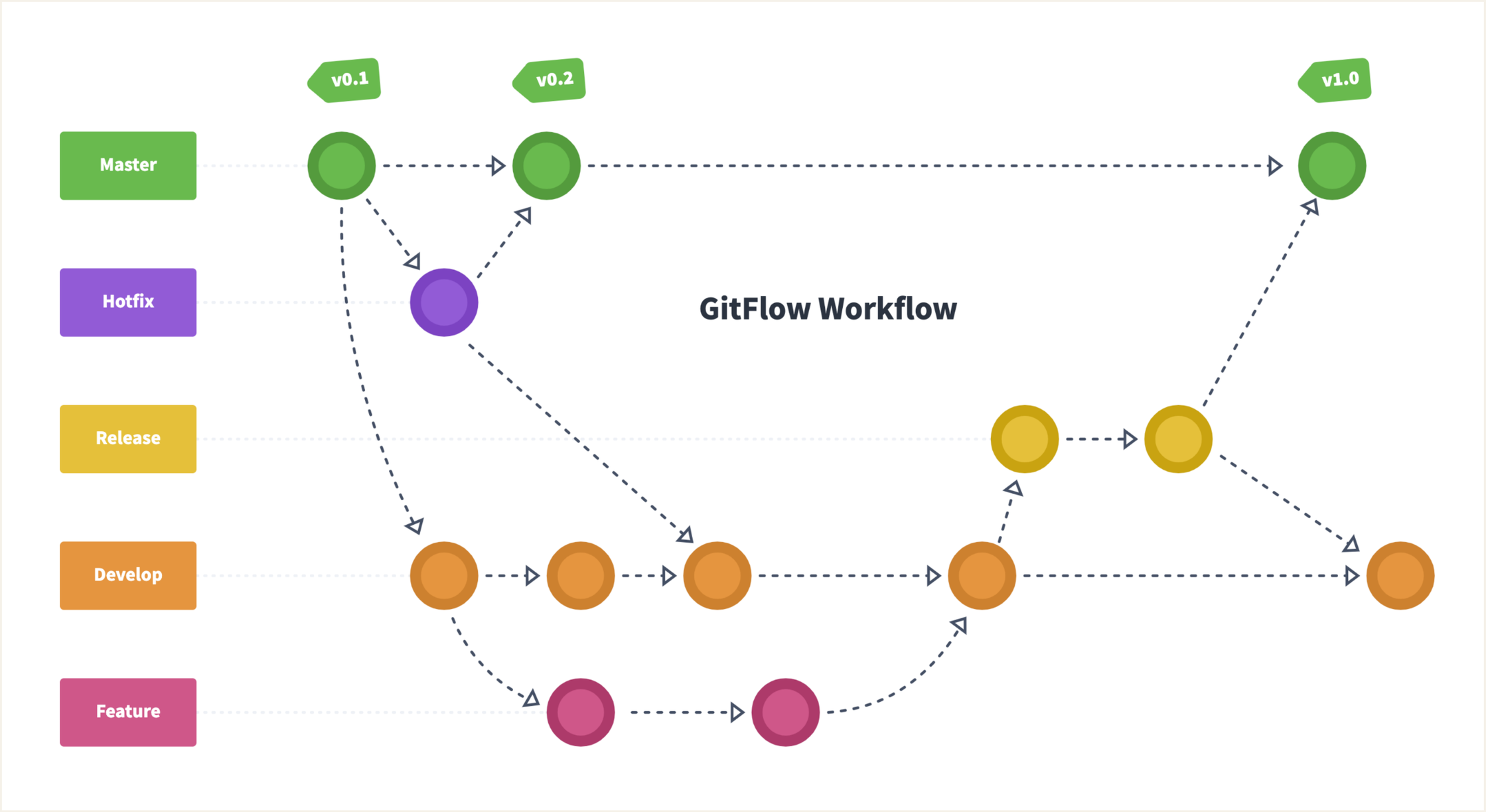
- git clone <ссылка .git>



git branch <branch_name> – создание новой ветки

git checkout <branch_name> – переключение на существующую ветку

git checkout –b <branch_name> – создание новой ветки и переключение на нее


v0.1
v0.2
v1.0
Master
Hotfix
Release
Develop
Feature
GitFlow Workflow

Команды с ветками

git branch

```
* main
```

Локальные ветки

git branch -r

```
origin/HEAD -> origin/main
origin/main
```

Удаленные ветки

git branch -a

```
* main
  remotes/origin/HEAD -> origin/main
  remotes/origin/main
```

Все ветки

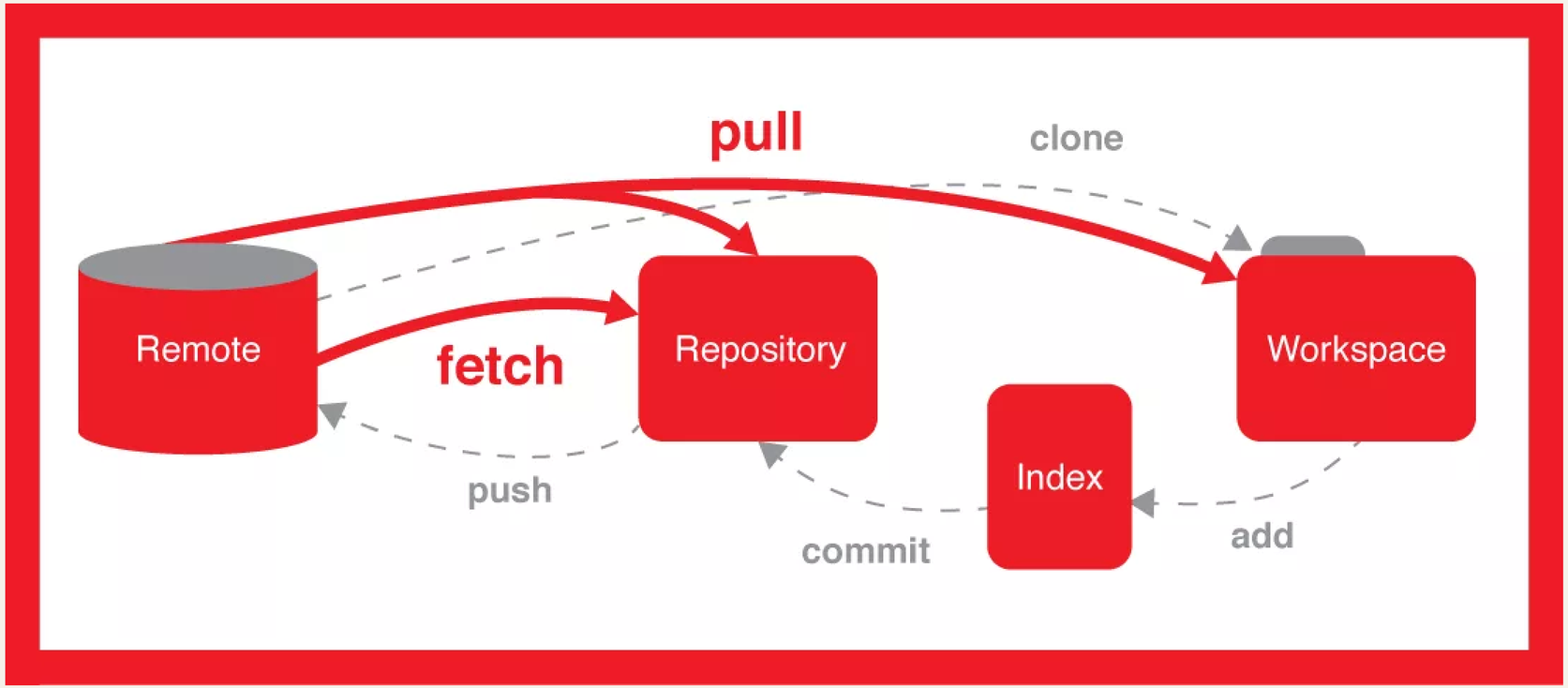
git branch -vv

```
* main 48aa7b7 [origin/main]
```

Связь локальная удаленная

git fetch – загружает обновления с удалённого репозитория без слияния с вашей текущей веткой

git pull — связка команд git fetch и git merge.



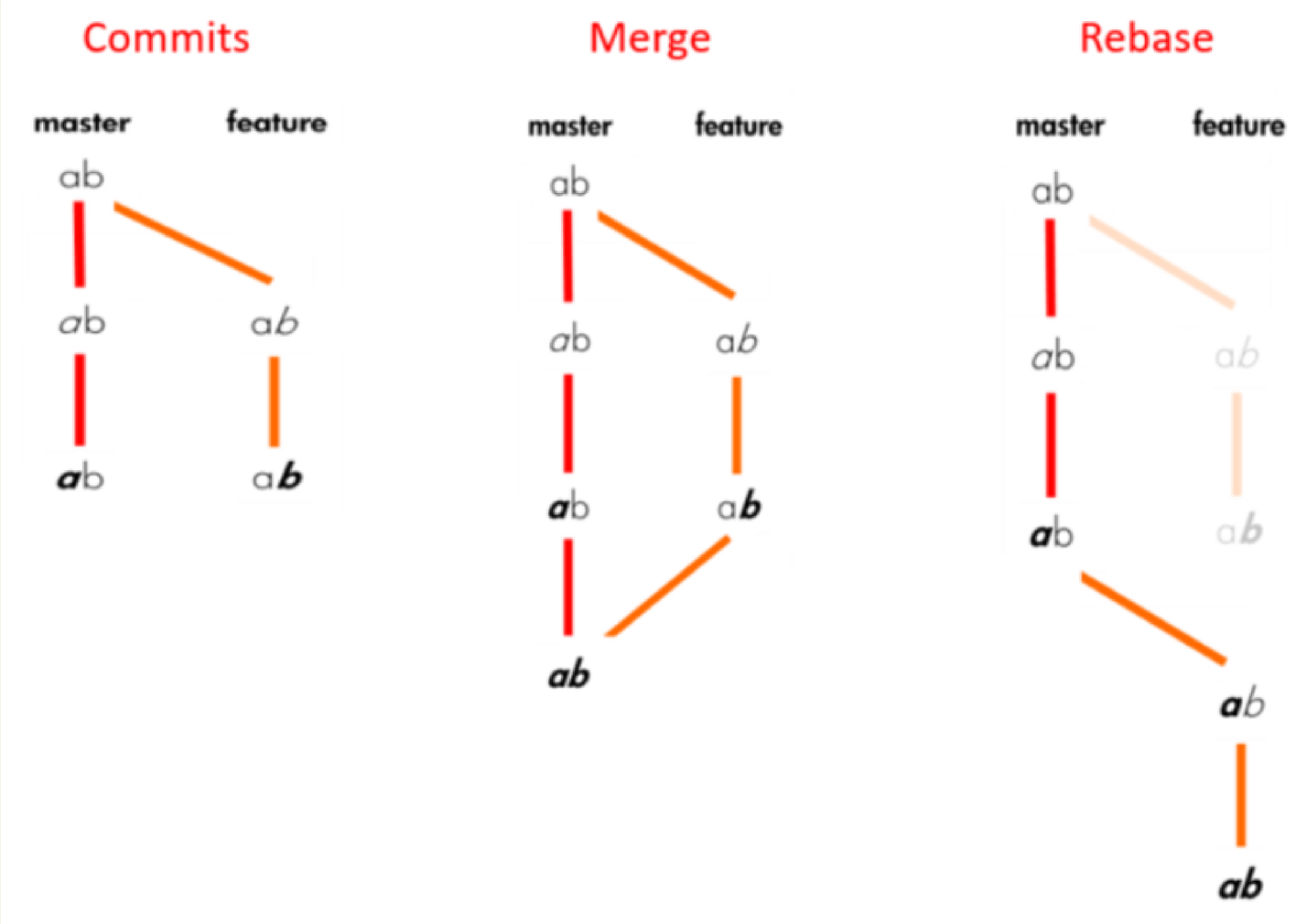
git merge — слияние

Внести изменения из одной ветки в другую

Меняется целевая ветка (куда мержим)

git rebase – перезаписывает историю

сжимает все изменения в один «патч». Затем он интегрирует патч в целевую ветку



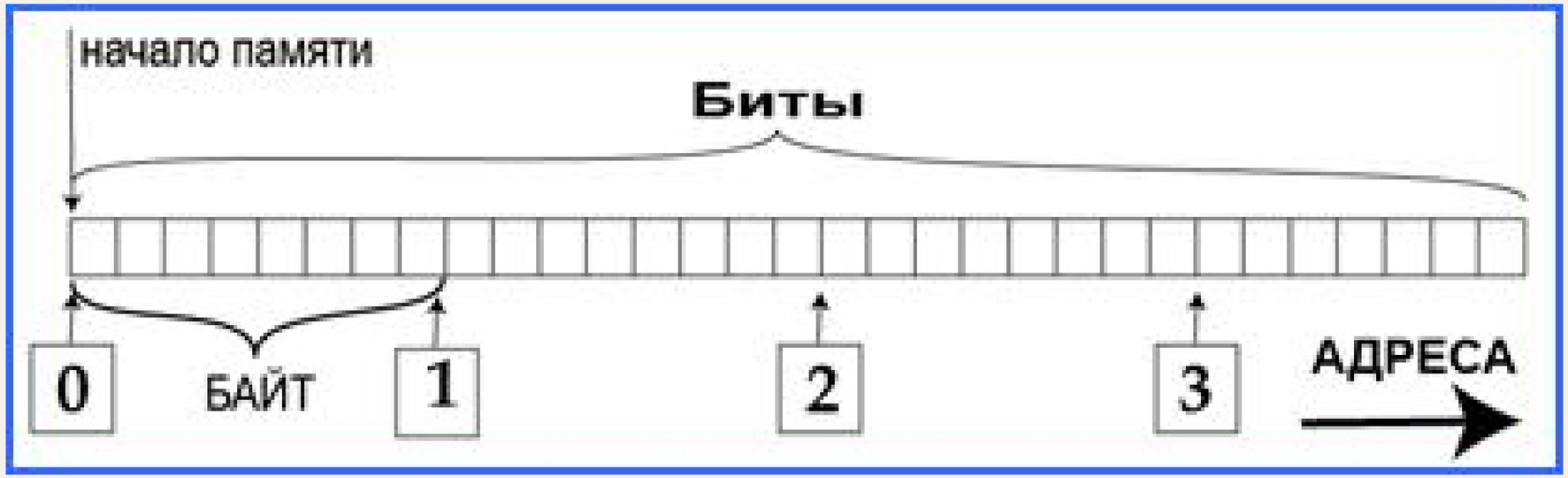
Memory layout

03

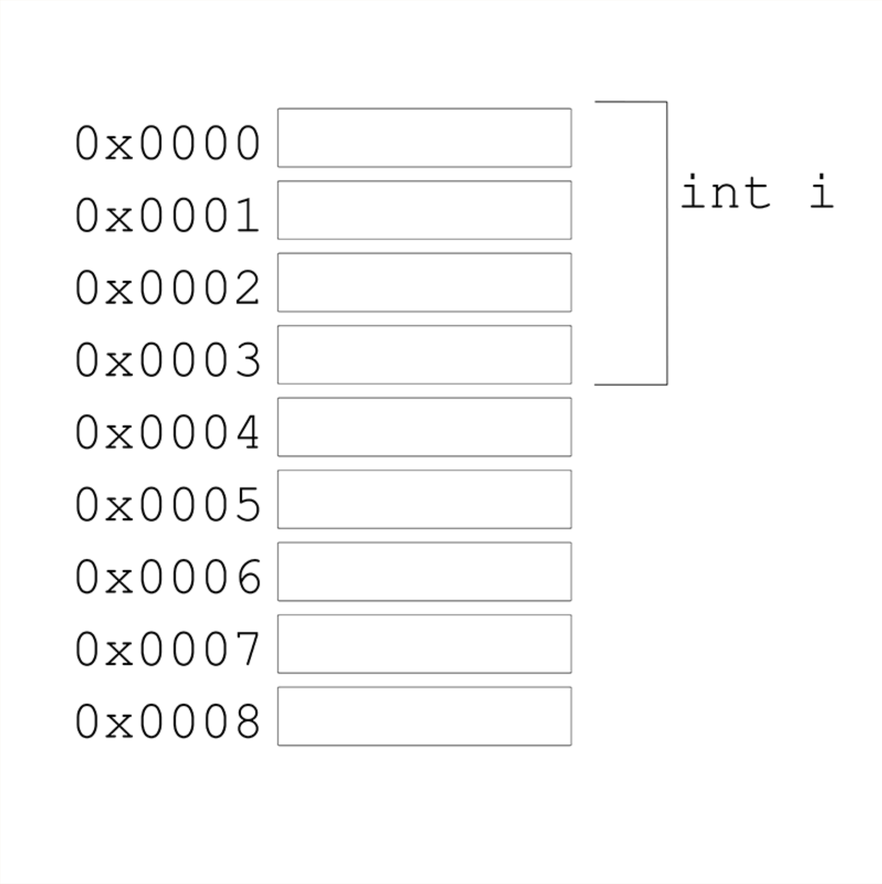
Память

- Память программы представляет собой последовательность байтов, каждый из которых имеет уникальный адрес.

- Все данные, включая переменные, объекты, инструкции программы и т. д., хранятся в памяти в виде последовательности байтов



Байт – это наименьшая адресуемая единица памяти в компьютере.

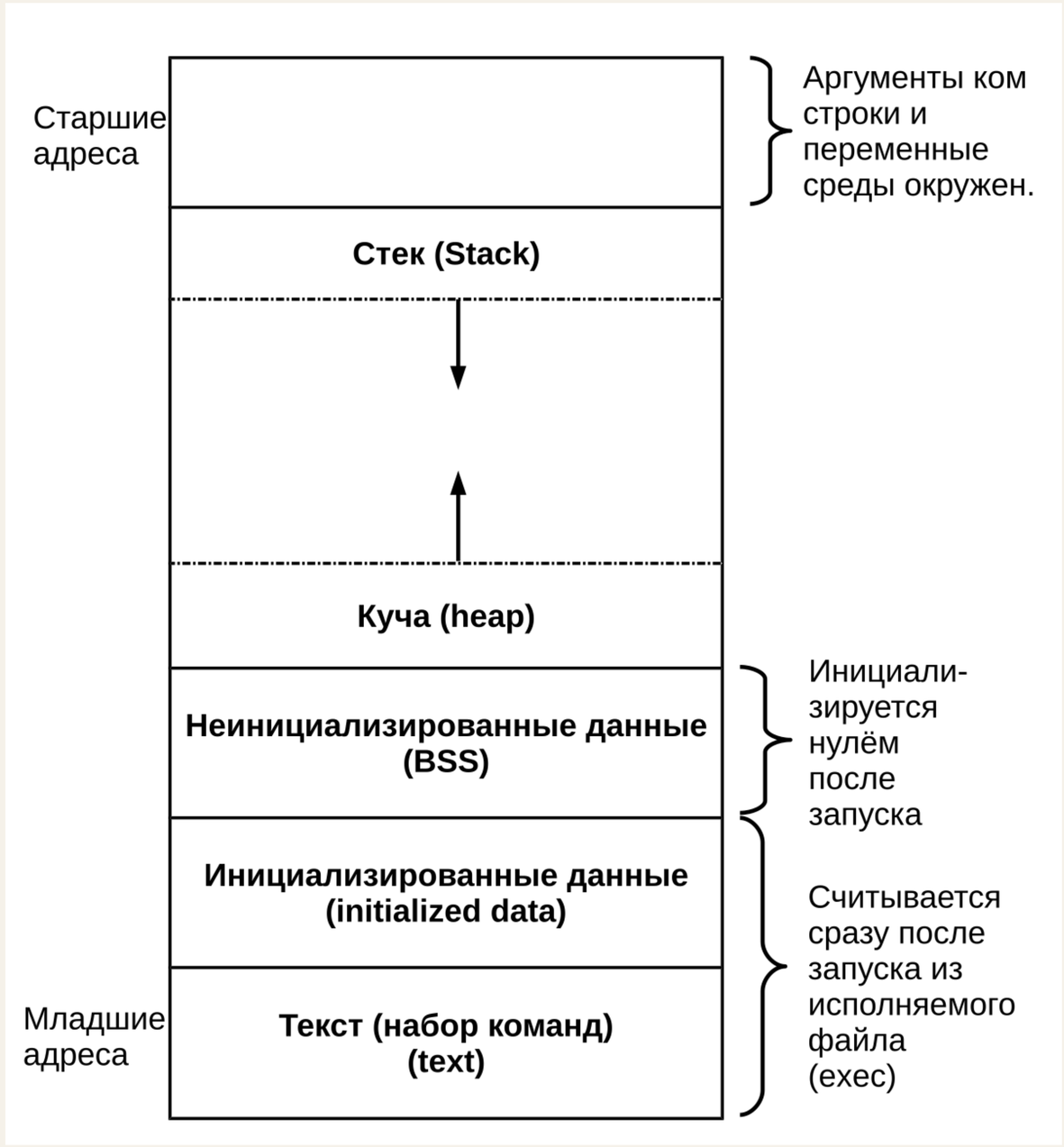


- Он состоит из последовательности битов, обычно 8 бит.
- Байт обычно используется для хранения данных и представления символов.
- В C++ байт может хранить значения любого элемента базового набора символов и любого элемента базового набора символов литералов.
- Стандарт C++ гарантирует, что размер байта составляет не менее 8 бит.

Память, выделяемая под программу

Memory layout – организации памяти для различных компонентов программы.

Когда мы запускаем нашу программу, операционная система **выделяет блок памяти для нашего процесса**. Этот блок памяти содержит различные сегменты

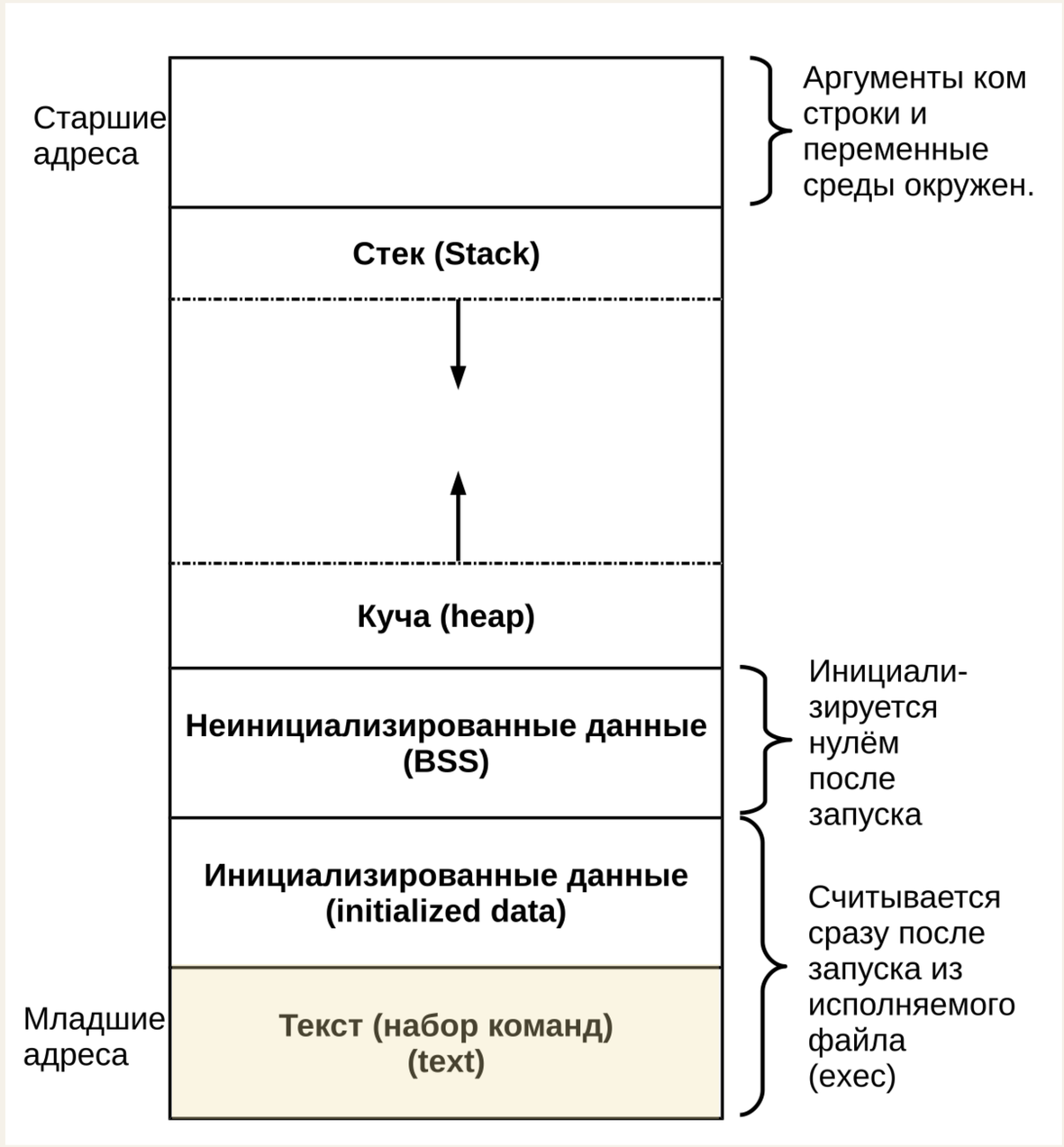


Память, выделяемая под программу

1. Сегмент кода (текстовый сегмент):

Содержит скомпилированные инструкции программы. Обычно представляет собой только для чтения и содержит исполняемый код.

Инструкции из этого сегмента извлекаются процессором для выполнения.

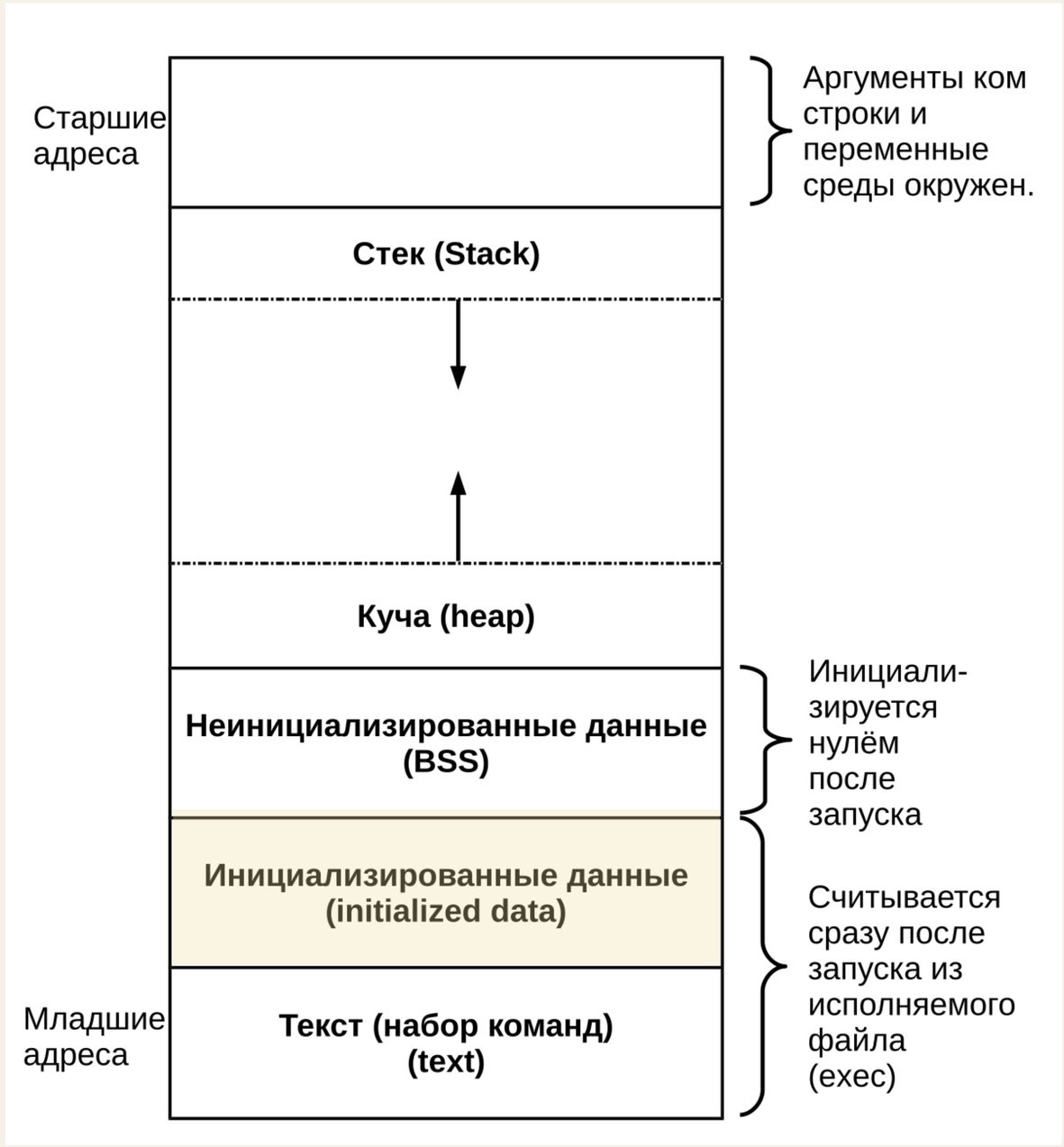


Память, выделяемая под программу

2. Сегмент данных (сегмент инициализированных данных)

Хранит инициализированные глобальные и статические переменные.

Инициализированные переменные имеют определенные значения, присвоенные на этапе компиляции.

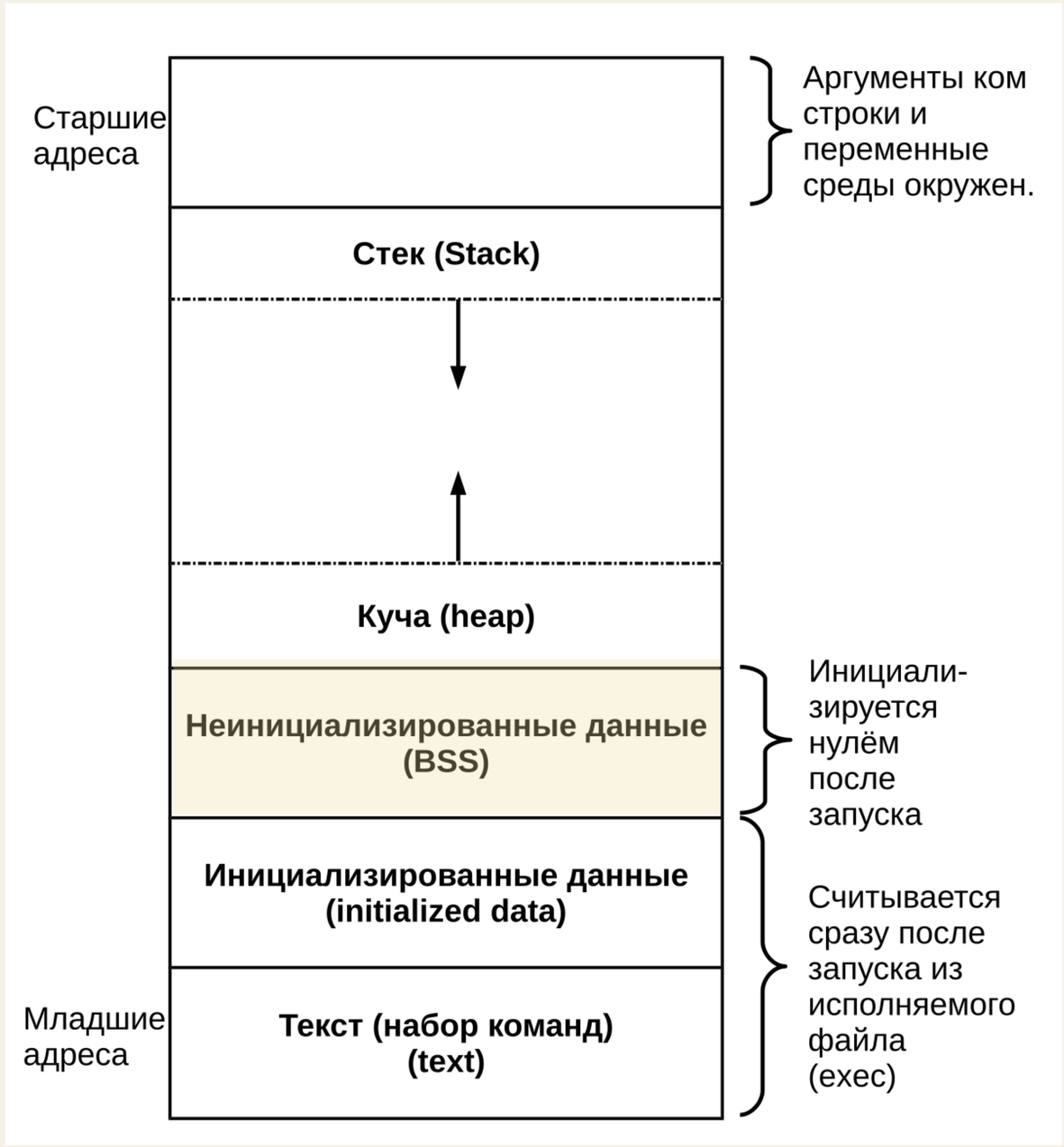


Память, выделяемая под программу

3. Сегмент BSS (сегмент неинициализированных данных)

Хранит глобальные и статические переменные, которые не инициализированы

Эти переменные **по умолчанию инициализируются нулями**.



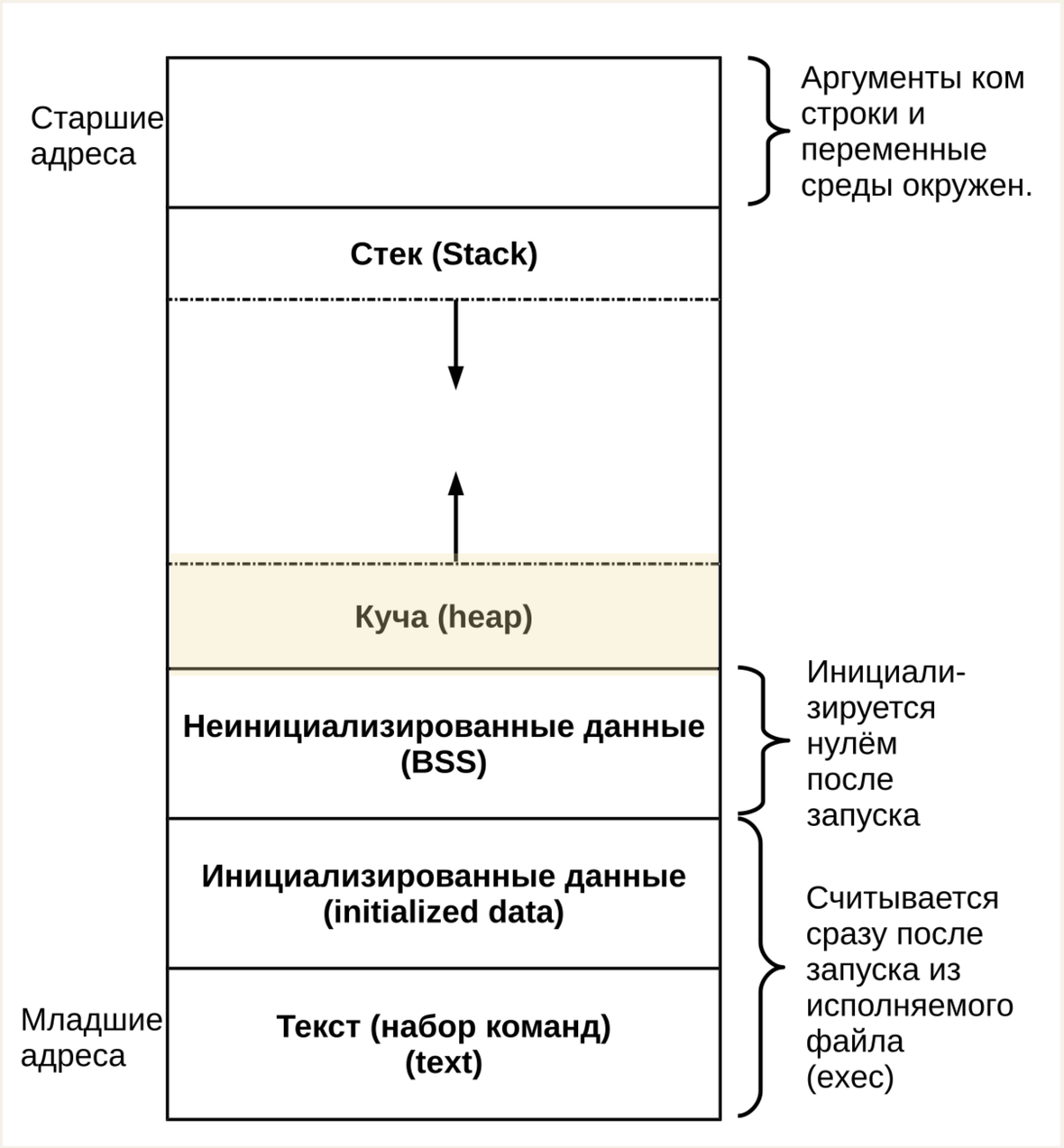
Память, выделяемая под программу

4. Куча (heap)

Область памяти, используемая для **динамического выделения памяти**.

Переменные, выделяемые в куче, определяются только во время выполнения программы и могут быть изменены в размере по мере необходимости.

Управление памятью на куче обычно осуществляется функциями, такими как malloc() и free() в C, или **операторами new и delete** в C++



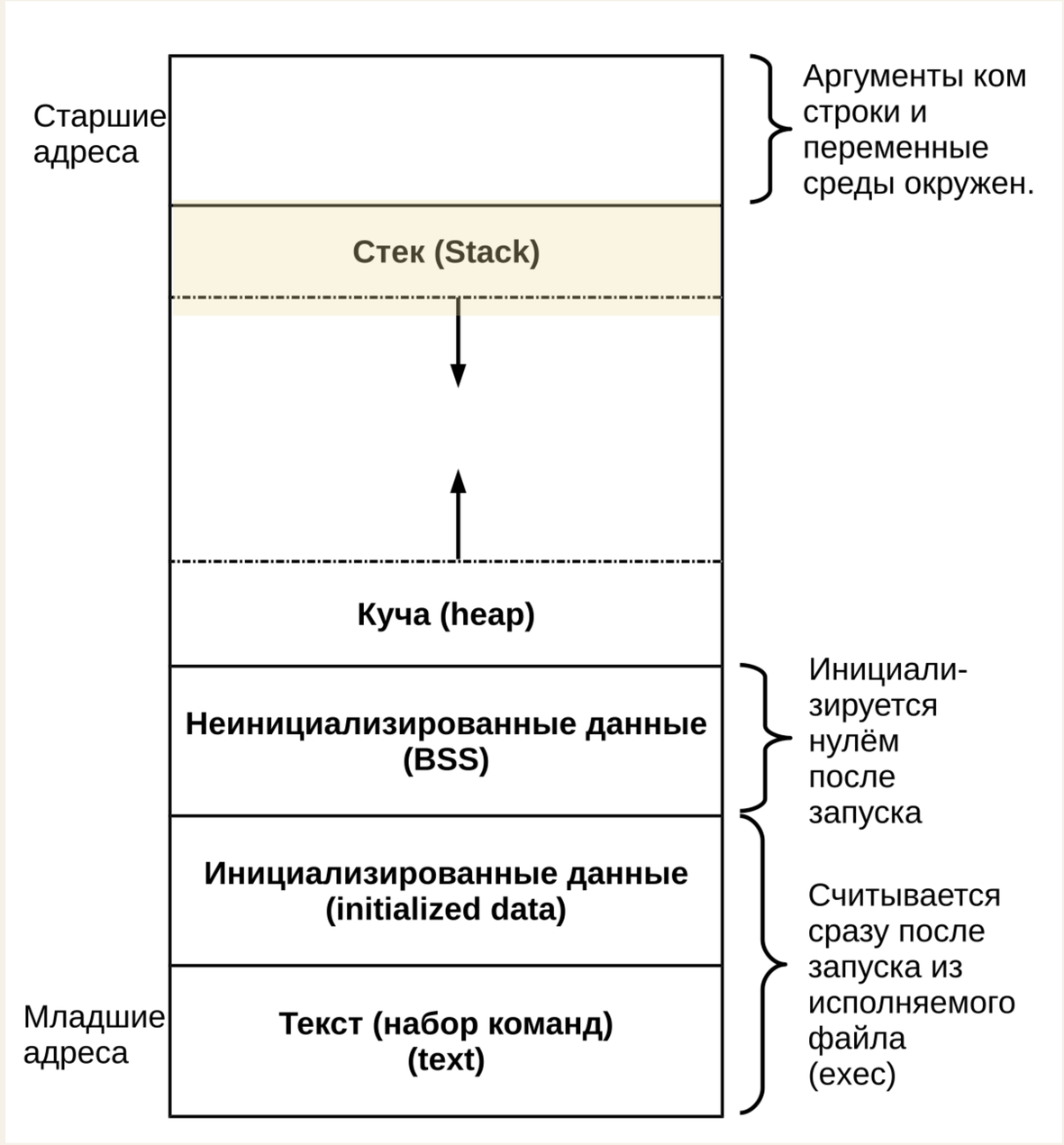
Память, выделяемая под программу

4. Стек (stack)

Хранит информацию о вызовах функций:

- локальные переменные
- параметры функций
- адреса возврата

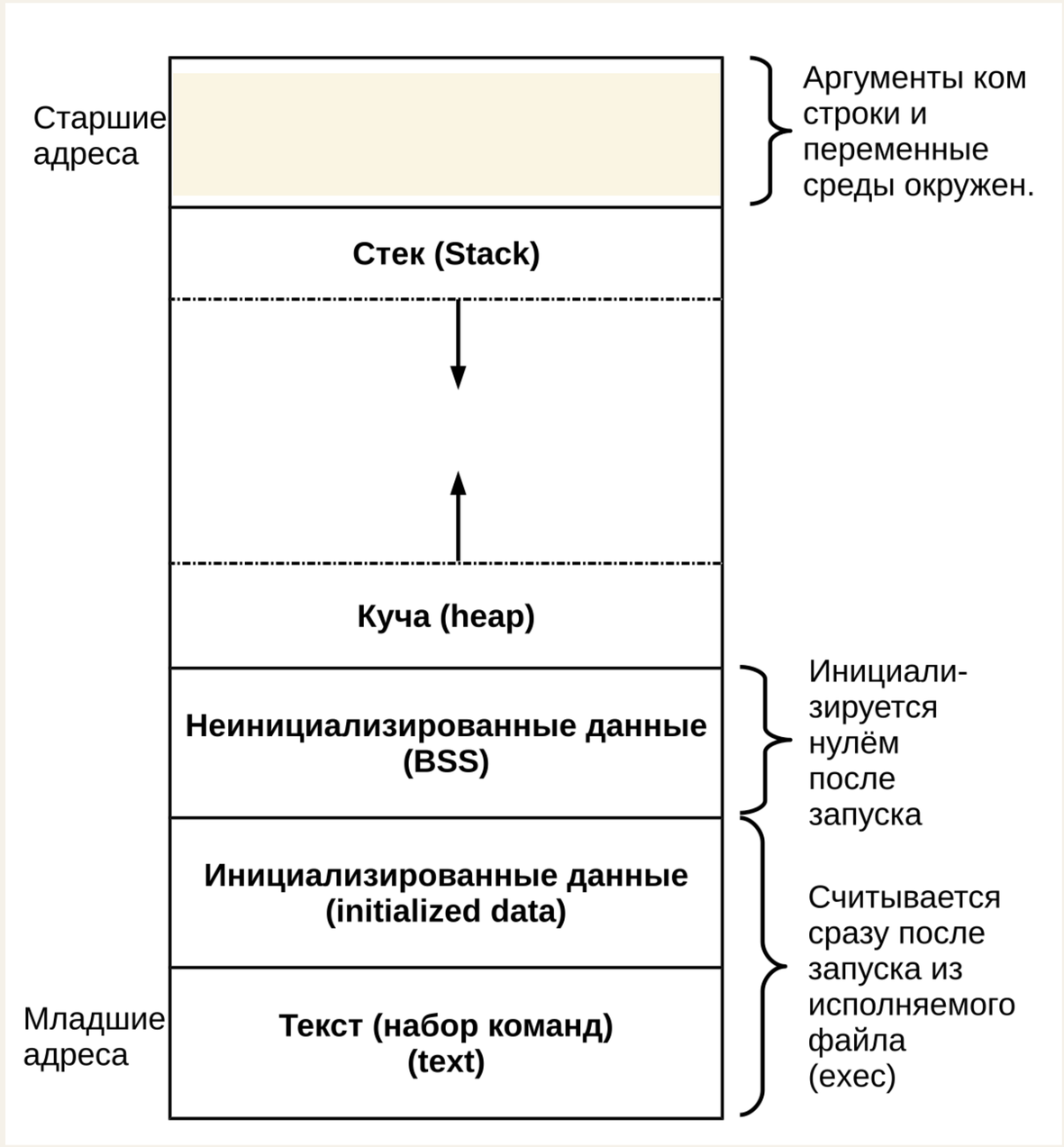
При каждом вызове функции создается новый **фрейм стека**, который добавляется в вершину стека. При возврате из функции ее фрейм удаляется из стека.



Память, выделяемая под программу

5. Аргументы командной строки (comand line arguments segment)

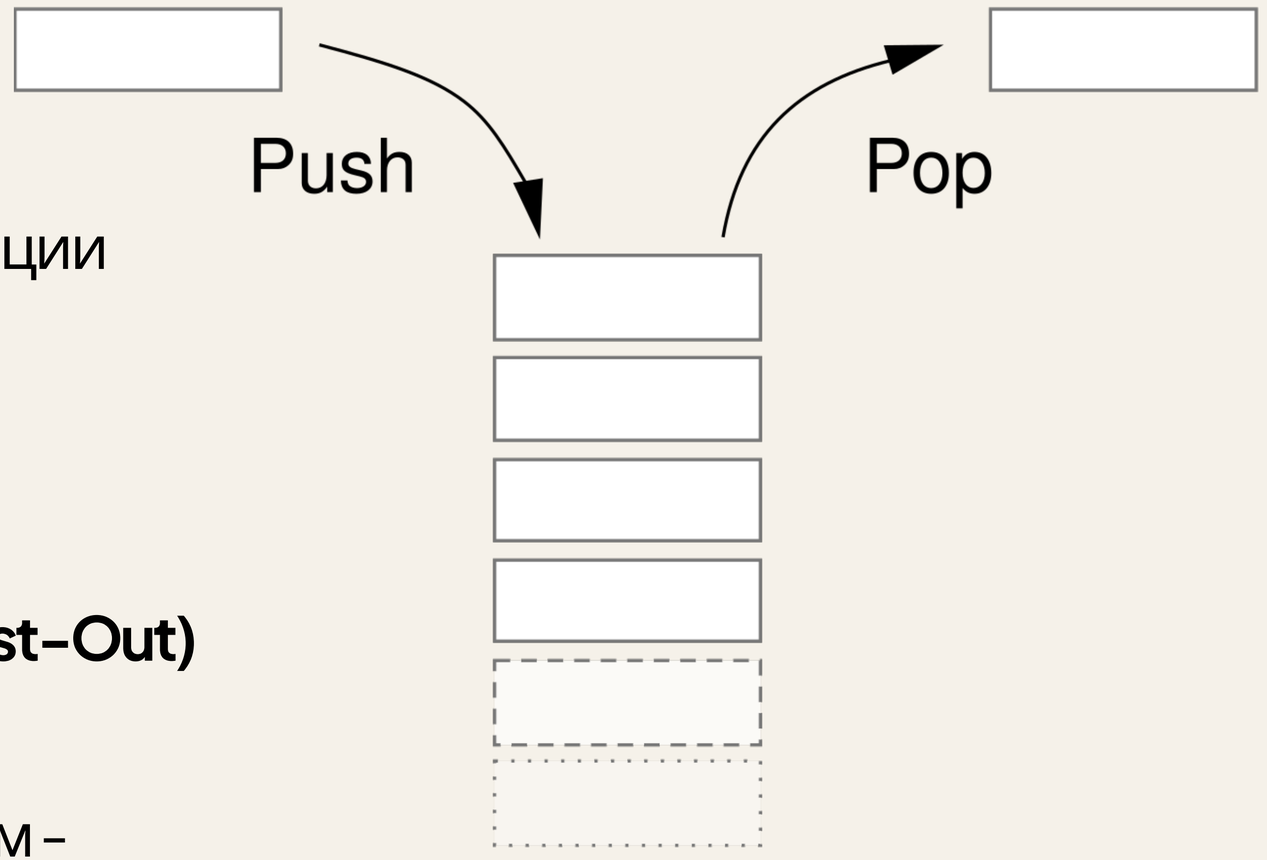
6. Переменные среды окружения (Environment Segment)



Стек vs Куча

04

Стек, как структура данных

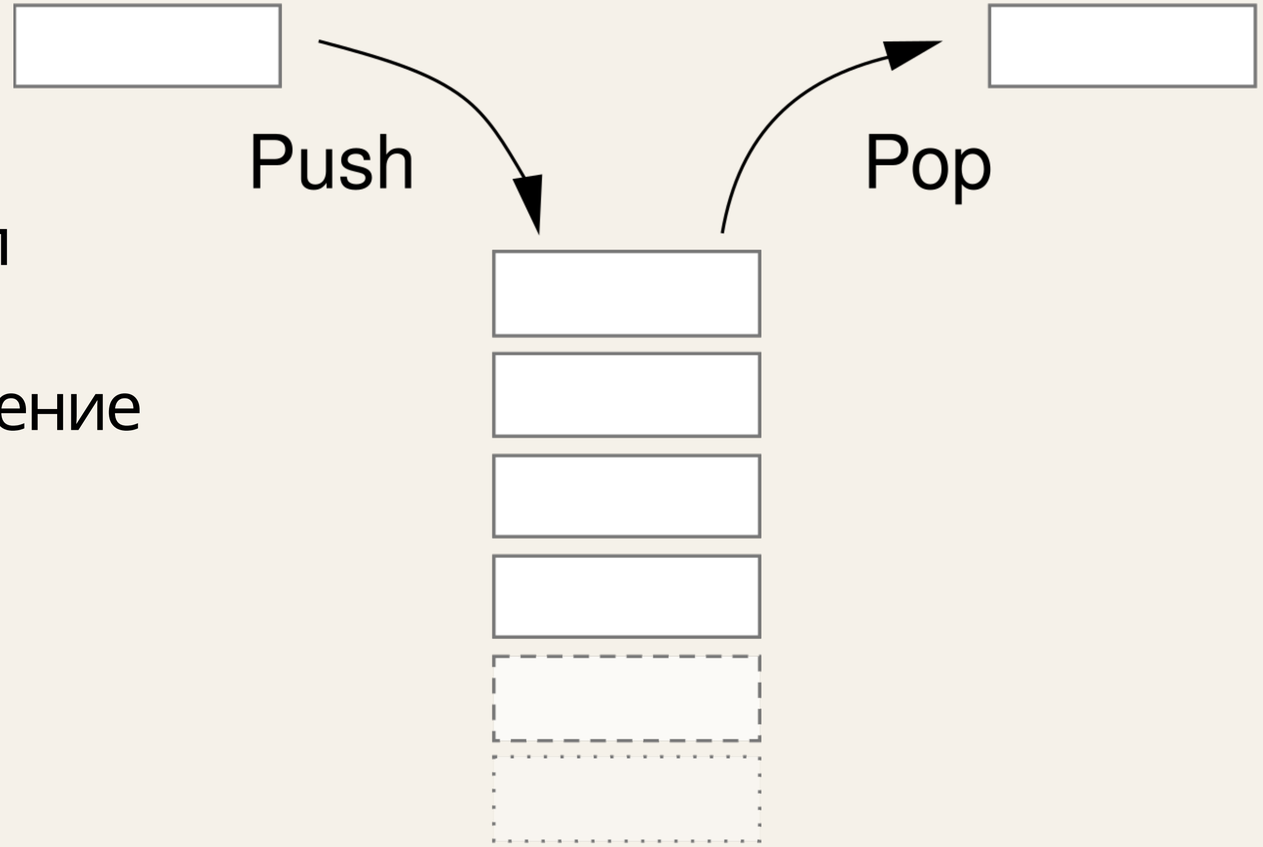


Структура данных – это способ организации данных в программировании для их эффективного использования.

Работает по принципу **LIFO (Last-In-First-Out)**

Элемент, добавленный в стек последним – первым покинет стек

Стек, как структура данных



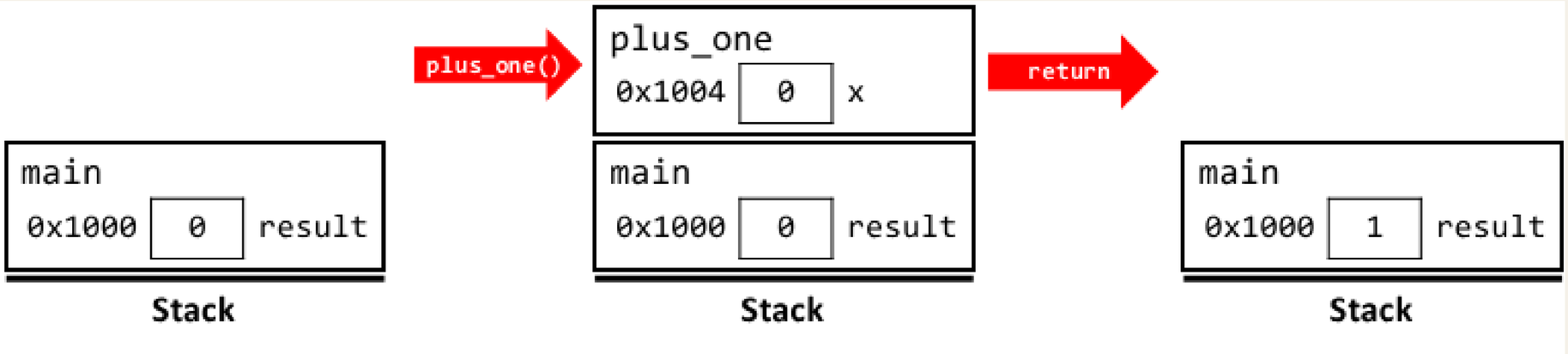
- Массив – произвольный доступ
- Стек – только добавление/удаление с одного конца

push – добавить элемент в конец
pop – удалить элемент с конца

Стек вызова функций

```
int main() {
  int result = 0;
  result = plus_one(0);
  result = plus_two(result);
  cout << result;
}
```

```
int plus_one(int x) {
  return x + 1;
}
```

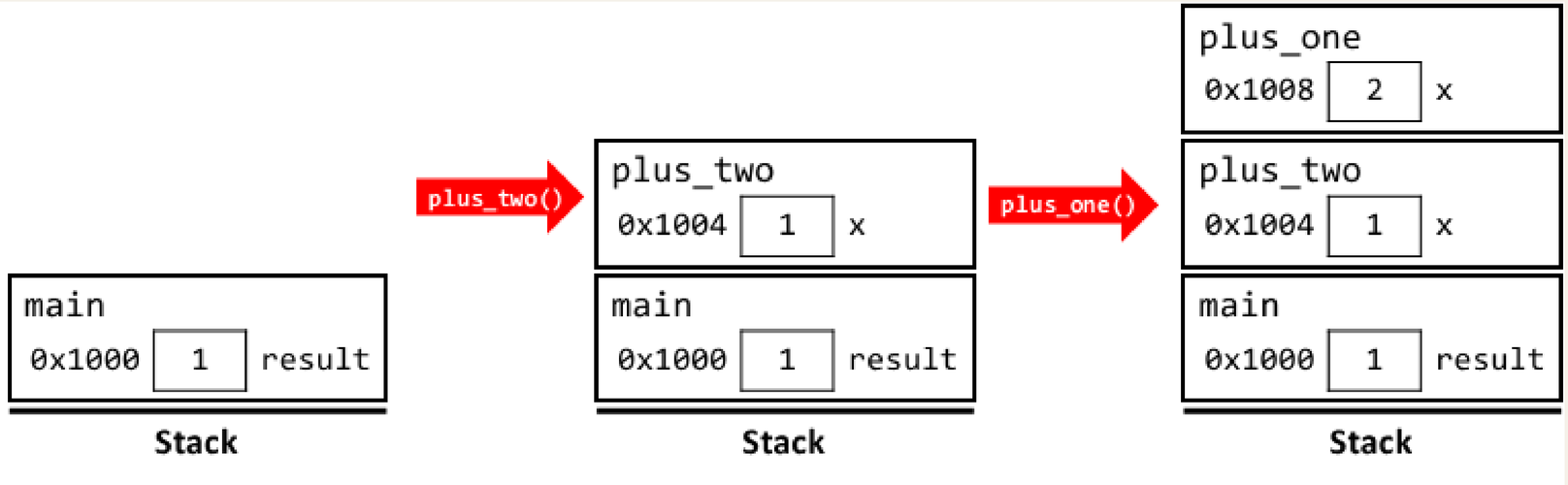


Стек вызова функций

```
int main() {
  int result = 0;
  result = plus_one(0);
  result = plus_two(result);
  cout << result;
}
```

```
int plus_one(int x) {
  return x + 1;
}
```

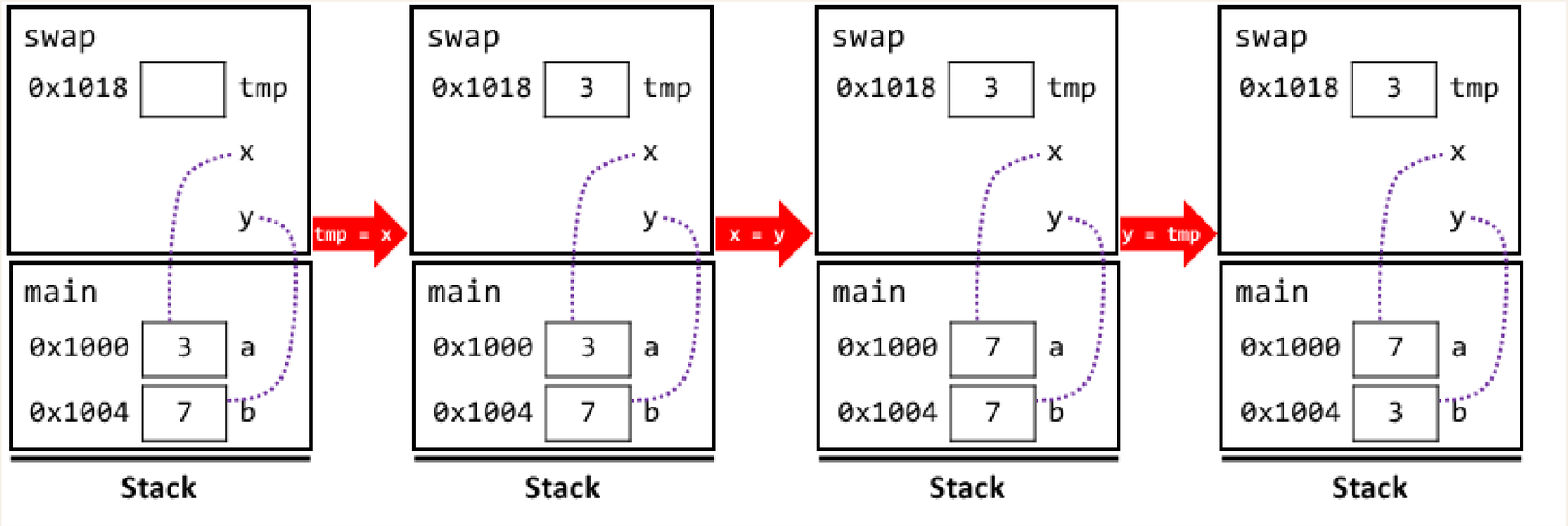
```
int plus_two(int x) {
  return plus_one(x + 1);
}
```



С передачей по ссылке

```
int main() {
  int a = 3;
  int b = 7;
  cout << a << ", " << b << endl;  // prints 3, 7
  swap(a, b);
  cout << a << ", " << b << endl;  // prints 7, 3
}
```

```
void swap(int &x, int &y) {
  int tmp = x;
  x = y;
  y = tmp;
}
```



Стек vs Куча

Свойства	Стек	Куча
Скорость выделения	Быстрое	Медленное
Область видимости	Ограничена блоком кода	Глобальная
Время жизни	Ограничено временем жизни блока кода	Остается выделенным до явного освобождения
Известность на этапе компиляции	Известна	Неизвестна
Метод доступа	Прямой доступ через переменную	Требуется доступ через указатель
Размер	Ограничен, обычно небольшой	Неограничен, можно выделять большие объемы памяти
Использование	Для хранения локальных переменных и данных, связанных с вызовами функций	Для хранения динамически выделяемых данных и больших структур данных
Подверженность переполнению	Да (ограниченный размер)	Нет (можно выделять память по мере необходимости)

Домашнее задание

- Проверить, что скинули ник гита
- Дождаться инвайта в организацию на гите
- Создать ПРИВАТНЫЙ репозиторий в орагинизации
- Создать README с ФИО и группой

формат названия misis2024f-24-XX-secname-i-o,

где XX – **последние цифры номера группы**, secname-i-o – **латиницей маленькими буквами транслитерация фамилии и инициалов автора**