1. **Стек (Stack Segment):**

* Служит для хранения локальных переменных и данных, связанных с функциями, такими как параметры функции или адрес возврата.
* Стек управляется автоматически: память выделяется при входе в функцию и освобождается при выходе из нее. Это намного быстрее в сравнении с выделением памяти в куче, но память в стеке сильно ограничена (Размер зависит от машины)

1. **Куча (Heap Segment)**:

* Используется для динамического распределения памяти. Память в куче выделяется и освобождается вручную с помощью таких функций, как new и delete. Память медленная, за   
  счет того, что адреса рандомные, но ограничена лишь свободной ОЗУ на машине.

1. **Сегмент кода (Code Segment или Text Segment)**:

* В этом сегменте хранится исполняемый код программы.
* Этот сегмент обычно имеет доступ только на чтение, чтобы предотвратить изменения кода во время выполнения.

1. **Сегмент констант (RO Data Segment)**:

* В этом сегменте хранятся литералы и другие константные данные, которые не должны изменяться во время выполнения программы. Например, строки, объявленные с помощью const.

**5. Сегмент данных (Data Segment)**:

* В этом сегменте хранятся глобальные и статические переменные, которые инициализированы до запуска программы.
* Этот сегмент обычно разделяется на две части:
  + **Сегмент инициализированных данных**: хранит глобальные и статические переменные, которые были инициализированы явным образом (например, int x = 5;).
  + **Сегмент неинициализированных данных (BSS Segment)**: хранит глобальные и статические переменные, которые были объявлены, но не инициализированы (например, int y;).

**Основные взаимодействия программы с памятью**

-**Объявление массивов строк**:

**-std::string units[], units\_thousands[], teens[], tens[], hundreds[]:**

-Эти массивы являются глобальными переменными и хранятся в сегменте данных (data segment). Сегмент данных используется для хранения глобальных и статических переменных, которые инициализированы до запуска программы.

* Для каждого элемента массива ("один", "два" и т.д.:) создается объект std::string, который хранит указатель на строковый литерал в сегменте констант.
* Объекты std::string также могут выделять дополнительную память в куче (heap) для хранения строки, если длина строки превышает оптимальный размер буфера

**-Прототипы функций**

**std::string convertNumber(int n, bool isMillion, bool isThousand);**

**std::string numToWords(int n);**

* Эти объявления функций размещаются в сегменте кода.
* Они не занимают дополнительной памяти при выполнении программы, так как это просто указания компилятору, что эти функции будут определены позже.

**-Определение функции main()**

При вызове функции main создаётся фрейм (участок стека, выделенный для каждой функции при её вызове.) стека, где хранятся её локальные переменные и параметры. Другими словами, при входе в main создается рамка в стеке, уже в этой рамке будут сохраняться последующие переменные обвяленные внутри main.

**-Ввод/вывод std::cout << и std::cout << ;**  
Вывод сообщения:

* Строка "Введите число до 999999999: " помещается в буфер std::cout.
* Буфер std::cout очищается, когда вы вызывается std::endl (это вызывает flush(смыв) буфера) или когда буфер заполняется.

Ввод числа:

* Данные вводятся с клавиатуры и сначала попадают в буфер std::cin.
* std::cin считывает данные из буфера и присваивает значение переменной number.

Буферы располагаются в сегменте данных и их размер обычно определяется стандартной библиотекой или может быть настроен пользователем.

**-Вызов функции numToWords ( std::cout << numToWords(number) << std::endl; )**

При вызове функции, для нее создается новый фрейм, находящийся над фреймом функции main().

* Аргумент number копируется в новый фрейм стека
* В стеке выделяется память для параметра n, который инициализируется значением number.

**-Проверка условий и возврат строки ( if (n == 0) return "ноль"; )**

Следующее верно для всех условий и возвратов.

* Условные операторы (if) сами по себе не изменяют структуру памяти, но они могут влиять на использование памяти, косвенно контролируя выполнение других операторов и функций.
* Если число n равно нулю, функция возвращает пустую строку. Возврат значения завершает выполнение функции и освобождает локальные переменные из стека.

Так примерно выглядят разделы памяти перед завершением программы:

|  |
| --- |
| **Сегмент кода** |
| - main() |
| - numToWords() |
| - convertNumber() |
| **Сегмент данных** |
| - std::string units[] |
| - std::string unitsFem[] |
| - std::string teens[] |
| - std::string tens[] |
| - std::string hundreds[] |
| **Сегмент постоянных данных** |
| - "один" |
| - "два" |
| - "три" |
| - ... |
| - "девятьсот" |
| **Стек** |
| - Фрейм convertNumber() |
| - std::string result |
| - int n |
| - bool isMillion |
| - bool isThousand |
| - Фрейм numToWords() |
| - std::string result |
| - int n |
| - Фрейм main() |
| - int number |