202505262350

Проект: Игра со строками с помощью cString

О чём это всё?

Представьте, что вы хотите поиграть со словами на вашем компьютере. Вы хотите хранить слова, объединять их, разрезать и изменять. Этот проект похож на создание собственного специального инструментария именно для этого! Мы создаём нечто, называемое cString, что является нашим собственным способом работы со словами (которые программисты называют "строками").

Почему бы просто не использовать обычные инструменты?

Ваш компьютер уже имеет способ работы со словами (это называется std::string), но мы создаём свой собственный, чтобы узнать, как всё работает "под капотом". Это как разобрать игрушечную машинку, чтобы увидеть, как работает двигатель - возможно, вы не будете использовать свой собственный двигатель в настоящей машине, но вы будете гораздо лучше понимать двигатели! Поскольку сейчас мы используем std::string, это похоже на сборку игрушечной машинки из отдельных, подготовленных частей.

Что в инструментарии? (Файлы проекта)

Наш инструментарий имеет три основные части:

- cString.h: Это как чертёж для нашего контейнера слов. Он описывает, что такое cString и что он может делать. Он сообщает компьютеру все вещи, которые наш контейнер слов сможет делать, такие как хранить слова, разрезать слова или склеивать слова вместе.
- cString.cpp: Здесь мы строим контейнер слов в соответствии с чертежом. Это содержит инструкции о том, как на самом деле выполнять все вещи, перечисленные в чертеже. Это как пошаговые руководства по созданию функций.
- main.cpp: Здесь мы играем с нашим контейнером слов. Это как небольшая игровая площадка, где мы создаём контейнеры слов cString и заставляем их делать разные вещи, чтобы показать, как они работают. Здесь происходит действие!

Контейнер слов cString - что он может делать?

Представьте cString как специальный контейнер, который хранит слова. У него есть множество инструментов для работы с этими словами:

- CString(int len = 0); - Создание пустого контейнера: Это способ создать новый, пустой контейнер слов. Параметр int len указывает контейнеру, сколько места выделить (сколько букв он может содержать), но можно начать с малого (или даже с нуля!). Если вы ничего не укажете $(= 0)$, он начнётся как маленький контейнер.
  - Пример: CString myWordBox; (Создаёт пустой контейнер)
  - Пример: CString bigWordBox(100); (Создаёт контейнер, который может содержать 100 букв)
- CString(const char *str); - Заполнение контейнера сразу: Это способ создать новый контейнер слов и сразу поместить в него слово! const char* str - это просто сложный способ сказать "обычное компьютерное слово".
  - Пример: CString greeting("Hello!"); (Создаёт контейнер со словом "Hello!" внутри)
- CString(const CString& other); - Копирование контейнера: Это создаёт совершенно новый контейнер, который является точной копией другого контейнера. Если вы измените слово в новом контейнере, это не изменит слово в оригинальном контейнере.
  - Пример: CString original("Meow");
  - Пример: CString copy(original); (Теперь copy также содержит "Meow", но это отдельный контейнер)
- ~CString() = default; - Очистка (не беспокойтесь об этом слишком сильно): Это то, что компьютер делает, когда он полностью закончил с контейнером слов. Это как выбросить контейнер и освободить занимаемое им пространство. Часть = default означает, что мы позволяем компьютеру обрабатывать это автоматически. Это означает 'компьютер, пожалуйста, сделай то, что считаешь лучшим, и я тебе доверяю'.
- std::string GetString() const; - Заглядывание внутрь: Это позволяет вам увидеть, какое слово хранится в контейнере. Это даёт вам способ прочитать слово. Это не позволяет вам изменить слово.
  - Пример: CString myWord("Banana");
  - Пример: std::string theWord = myWord.GetString(); (Теперь theWord содержит "Banana")
- void SetString(const char* str); - Изменение слова: Это позволяет вам поместить новое слово в контейнер, заменив то, что там было раньше.
  - Пример: CString myWord("Apple");
  - Пример: myWord.SetString("Orange"); (Теперь myWord содержит "Orange")
- void SetLength(int length); - Изменение размера слова: Это позволяет изменить количество символов внутри слова. Если у вас меньше символов, чем новый размер, пустые места заполняются ' 10 '.

    - Пример: CString myWord("Apple");
    - Пример: myWord.SetLength(10); (Теперь myWord содержит "Apple\0\0\0\0\0")
- int GetLength() const; - Подсчёт букв: Это сообщает вам, сколько букв в слове, которое находится в контейнере.
    - Пример: CString myWord("Dog");
    - Пример: int letterCount = myWord.GetLength(); (Теперь letterCount равен 3)
- std::string::size_type FindSubStr(const char* substr) const; - Поиск меньшего слова: Это как поиск меньшего слова внутри слова в контейнере. Это сообщает вам, где начинается меньшее слово.
    - Пример: CString myWord("Butterfly");
    - Пример: std::string::size_type location = myWord.FindSubStr("fly"); (Теперь location равен 4, потому что "fly" начинается с 4-й буквы)
- CString StrCopy(int start, int byte) const; - Вырезание части: Это как взять часть слова и создать новый контейнер слов с этой частью. start - это место, с которого вы начинаете вырезание, а byte - это сколько букв вырезать.
    - Пример: CString myWord("Hamburger");
    - Пример: CString piece = myWord.StrCopy(4, 3); (Теперь piece содержит "bur")
- void ReplaceSubStr(const char *substr1, const char* substr2); - Замена части: Это как замена части слова на другую часть.
    - Пример: CString myWord("Ice Cream");
    - Пример: myWord.ReplaceSubStr("Cream", "Water"); (Теперь myWord содержит "Ice Water")
- void RemoveSubStr(const char* substr); - Удаление части: Это как удаление части слова полностью.
    - Пример: CString myWord("Hot Dog");
    - Пример: myWord.RemoveSubStr("Hot "); (Теперь myWord содержит "Dog")
- CString operator+(const CString& other) const; - Склеивание контейнеров: Это способ объединить два контейнера слов в один большой контейнер.
    - Пример: CString first("Hello");
    - Пример: CString second(" World");
    - Пример: CString combined = first + second; (Теперь combined содержит "Hello World")
- CString& operator=(const CString& other); - Копирование контейнера (другой способ): Это другой способ скопировать контейнер слов. Это как сказать "сделай слово в этом контейнере таким же, как слово в том контейнере." Если они не один и тот же контейнер!
    - Пример: CString box1("Red");
    - Пример: CString box2;

    - Пример: box2 = box1; (Теперь box2 также содержит "Red")
- CString& operator=(const char* str); - Помещение обычного слова в контейнер: Это как помещение обычного компьютерного слова (строки в стиле C) в контейнер слов.
    - Пример: CString myBox;
    - Пример: myBox = "Blue"; (Теперь myBox содержит "Blue")
- char& operator[](int index); - Получение одной буквы: Это позволяет вам получить доступ к конкретной букве в контейнере слов. Параметр int index - это номер буквы (начиная с 0). Таким образом, myWord[0] - это первая буква, myWord[1] - вторая буква и так далее. Эта версия позволяет вам изменить букву.
- const char& operator[](int index) const; - Получение одной буквы, безопасно: То же, что и выше, но const означает, что это позволяет вам только смотреть на букву. Вы не можете её изменить. Это используется, когда у вас есть объект const CString.
    - Пример: CString myWord("Cat");
    - Пример: char firstLetter = myWord[0]; (Теперь firstLetter содержит 'C')
    - Пример: myWord[1] = 'o'; (Теперь myWord содержит "Cot")
- friend std::ostream& operator<<(std::ostream& os, const CString& string); - Показ слова: Это позволяет вам вывести контейнер слов прямо на экран с помощью cout. Это то, что делает cout << myWord; работающим.
- friend std::istream& operator>>(std::istream& is, CString& string); - Ввод слова: Это позволяет вам ввести слово прямо с клавиатуры (используя cin) в контейнер слов. Это то, что делает cin >> myWord; работающим.

Важный секрет:

Мы используем специальный, сверхбезопасный встроенный контейнер слов внутри нашего CString, называемый std::string. Он заботится обо всех сложных деталях хранения букв, поэтому нам не нужно беспокоиться о совершении ошибок.

Почему это важно (общая картина)

Этот проект помогает вам понять:

- Как компьютеры хранят и манипулируют словами.
- Как создавать свои собственные инструменты (классы) для выполнения конкретных задач.
- Как избежать распространённых ошибок программирования (например, утечек памяти).

Даже несмотря на то, что мы используем std::string внутри, понимание того, как все части соединяются вместе, является очень ценным!