Підготував Рубанський Сергій, студент групи комп’ютерна математика-2.

Командний рядок

Зробіть парсінг командного рядочку функціями стандартної бібліотеки STL Ci++.

Алгоритм дій:

1. Зчитуємо рядок s з консолі за допомогою команди
getline(cin, s);
або якщо з файлу коли відкритий потік file:
getline(file, s);
Додаємо в кінець рядку s пробіл задля повної коректності роботи нашого наступного алгоритму.
2. Створюємо масив рядків argv. Нехай j - це змінна яка зберігає індекс початку слова, спочатку j = 0. Проходимось по s і якщо бачимо на i-ому місці пробіл додаємо в argv підрядок з j індексу по i-1, переприсвоюємо j.

Записаний алгоритм:

```
s.push_back(' ');
   string argv[s.length()];
   int j = 0;
   int k = 0;
   for (int i=0;i<s.length(); i++){
      if (s[i] == ' '){
         argv[k] = s.substr(j, i-j);
         k++;
         j = i+1;
      }
   }
  int argc = k;
```

Argc - кількість слів в масиві argv.

Протестуємо наш алгоритм:

Вводимо рядок

```
Hello world -filename file 12sa.txt
```

Отримуємо на вихід:

```
Words:
Hello
world
-filename
file
12sa.txt
```

Рядок:

```
python
```

Output:

```
Words:
python
```

Як бачимо, все працює нормально.

Опис бібліотек та функцій використаних в проекті.

1) Бібліотека iostream
Частина стандартної бібліотеки C++. В ній реалізована підтримка для файлового вводу/виводу даних вбудованих типів. Операції введення/виводу виконуються за допомогою класів istream (потокове введення) та ostream (потокове виведення). Третій клас, iostream, є похідним від них і підтримує двонаправлене введення/виведення. Для зручності в бібліотеці визначено три стандартні об'єкти-потоки:

cin – об'єкт класу istream, що відповідає стандартному введенню. Загалом він дозволяє читати дані з терміналу користувача;
cout – об'єкт класу ostream, який відповідає стандартному висновку. У випадку він дозволяє виводити дані на термінал користувача;
Виведення здійснюється, за допомогою перевантаженого оператора зсуву вліво (<<), а введення - за допомогою оператора зсуву вправо (>>): cout << s; або cin >> s; відповідно.
#include<iostream>

2) Бібліотека string
Клас string призначений для роботи з рядками типу char*, які є рядком із завершальним нулем. Клас string був запроваджений як альтернативний варіант для роботи з рядками типу char*. Рядки, які завершуються символом ‘¥0’ ще називаються С-рядками. Оскільки, string є класом, можна оголошувати об'єкти цього класу. Щоб працювати з string підключив бібліотеку <string>.
#include<string>

3) Бібліотека cmath
Бібліотека cmath визначає набір функцій для виконання загальних математичних операцій та перетворень.
#include<cmath>

4) Бібліотека fstream
Бібліотека для роботи з файлами, у якому підключено такі заголовкові файли як <ifstream> — бібліотека для файлового введення, і <ofstream> — бібліотека для файлового виведення. Бібліотека <fstream> містить функції для роботи з файлами. Об'єкти цього класу підтримують об'єкт filebuf як свій внутрішній буфер потоку, який виконує операції введення/виведення у файл, з яким вони пов'язані (якщо такі є). Потоки файлів асоціюються з файлами або під час побудови, або за допомогою виклику члена open. Клас fstream є похідним від iostream
#include <fstream>

Підключив простір імен std:
using namespace std;

Відкриття файлу для запису у кінець:

```
ofstream file;
file.open(output_filename, ios::app);
```

Безпосередньо запис:

```
file << "Error.";
```

Закриття файлу:

```
file.close();
```

Метод з бібліотеки string для перетворення рядка в дійсне число:

```
a[i] = stof(s);
```

Метод eof() класу ios використовується для перевірки, чи потік викликав будь-яку помилку EOF (End Of File). Використав для підрахунку кількості рядків у файлі.

```
while (!base.eof())
{
    base.getline(str, 1024, '\n');
    i++;
}
```

Метод compare() порівнює string об’єкти. Повертає 0 якщо вони рівні.

```
if (t1.compare(argv[0]) != 0 || t2.compare(argv[2]) != 0){
```

Оператор switch для вибору одного з багатьох блоків коду для виконання.
Синтаксис :

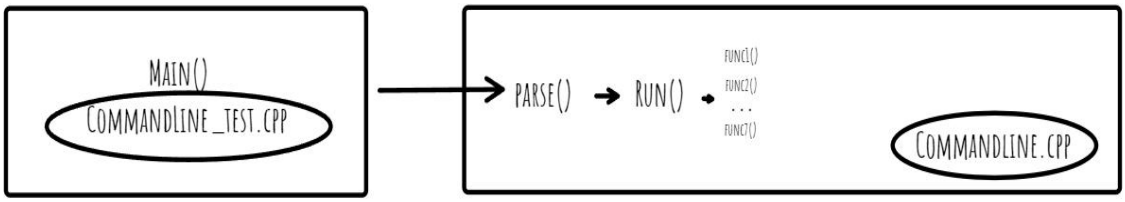
```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

Ось як це працює:
>Вираз switch обчислюється один раз
>Значення виразу порівнюється зі значеннями кожного випадку
>Якщо є збіг, виконується відповідний блок коду
Ключові слова break і default є необов'язковими.

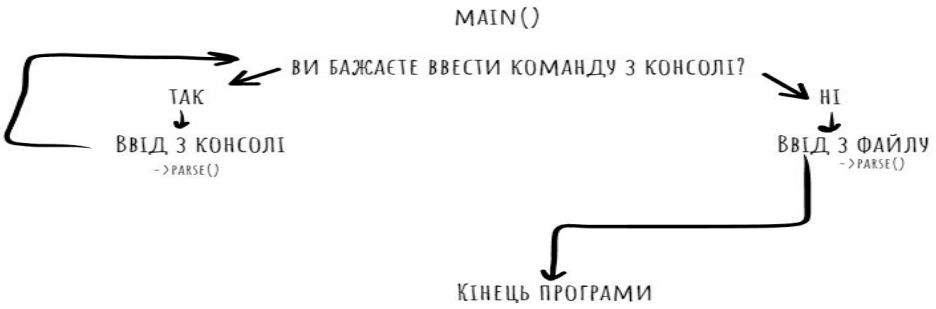
Я використав це для виклику функції відповідної до заданого номеру.

```
switch (numb){
    case 1:
        func1(argc, argv);
        break;
    case 2:
        func2(argc, argv);
        break;
    case 3:
        func3(argc, argv);
        break;
    case 4:
        func4(argc, argv);
        break;
    case 5:
        func5(argc, argv);
        break;
    case 6:
        func6(argc, argv);
        break;
    case 7:
        func7(argc, argv);
        break;
}
```

Структура проекту:



Будова main()



Будова всіх функцій func1, func2, …, func7