Перегрузка бинарных операторов для класса Path

https://godbolt.org/z/v9Gfz87G8

```
class Path {
public:
    Path() noexcept;
    Path(const Path& p);
    Path(const char* input);
    Path& operator=(const char* input);
    Path& operator=(const Path& input);

    friend std::ostream& operator<<(std::ostream& os, const Path& path);
    friend std::istream& operator>>(std::istream& is, Path& path);

    friend Path operator/(const Path& lhs, const Path& rhs);
    friend Path operator/(const Path& lhs, const char* rhs);

    friend bool operator==(const Path& lhs, const Path& rhs) noexcept;
    friend bool operator<(const Path& lhs, const Path& rhs) noexcept;
    friend bool operator>(const Path& lhs, const Path& rhs) noexcept;
    friend bool operator<=(const Path& lhs, const Path& rhs) noexcept;
    friend bool operator>=(const Path& lhs, const Path& rhs) noexcept;
    friend bool operator!=(const Path& lhs, const Path& rhs) noexcept;
private:
    std::unique_ptr<char[]> value = nullptr;
};
```

Конструкторы

```
1.  Path::Path() noexcept{
2.      value = std::make_unique<char[]>(1);
3.  }
4.
    Path::Path(const Path& p) {
5.      size_t length = std::strlen(p.value.get()) + 1;
6.      value = std::make_unique<char[]>(length);
7.
        for (size_t i = 0; i < length; ++i) {
8.          value[i] = p.value[i];
9.      }
10. }
11.
    Path::Path(const char* input) {
12.     size_t length = std::strlen(input) + 1;
13.     value = std::make_unique<char[]>(length);
14.     for (size_t i = 0; i < length; ++i) {
15.         value[i] = input[i];
16.     }
17. }
```

```
Path home_directory = “”;
```

Оператор присваивания

```
1.  Path& Path::operator=(const char* input) {
2.      size_t length = std::strlen(input) + 1;
3.      value = std::make_unique<char[]>(length);
4.      for (size_t i = 0; i < length; ++i) {
5.          value[i] = input[i];
6.      }
7.      return *this;
8.  }
9.
    Path& Path::operator=(const Path& p) {
10.     if (this != &p) {
11.         value = std::make_unique<char[]>(std::strlen(p.value.get()) + 1);
12.         std::strcpy(value.get(), p.value.get());
13.     }
14.     return *this;
15. }
```

```
1.      Path diy_path = "/home/user/diy_path";
2.      Path diy_path2;
3.      diy_path2 = "/home/user/diy_path2";
4.      Path empty_path;
5.      Path copy_of_diy2 = diy_path2;
```

```
inbuilt filesystem: "/home/user/stp/filesystem"
out diy: "/home/user/diy_path"
out diy2: "/home/user/diy_path2"
out empty_path: ""
copy path: "/home/user/diy_path2"
```

Оператор /

```
1.  Path operator/(const Path& lhs, const char* rhs) {
2.      Path result;
3.      size_t lhsLength = std::strlen(lhs.value.get());
4.      size_t rhsLength = std::strlen(rhs);
5.
        result.value = std::make_unique<char[]>(lhsLength + rhsLength + 2);
6.      std::memcpy(result.value.get(), lhs.value.get(), lhsLength);
7.
        if (lhsLength > 0 && lhs.value.get()[lhsLength - 1] != '/') {
8.          result.value.get()[lhsLength] = '/';
9.          ++lhsLength;
10.     }
11.
        std::memcpy(result.value.get() + lhsLength, rhs, rhsLength + 1);
12.     return result;
13. }
```

Оператор ==

```
1. bool operator==(const Path& lhs, const Path& rhs) noexcept
   {
2.     const char* str1 = lhs.value.get();
3.     const char* str2 = rhs.value.get();
4.     while (*str1 != '\0' && *str2 != '\0'){
5.         if (*str1 != *str2){
6.             return false;
7.         }
8.     }
9.     return true;
10.}
```

Оператор <

```
1.  bool operator<(const Path& lhs, const Path& rhs) noexcept {
2.      const char* str1 = lhs.value.get();
3.      const char* str2 = rhs.value.get();
4.
        while (*str1 != '\0' && *str2 != '\0') {
5.          if (*str1 < *str2) {
6.              return true;
7.          } else if (*str1 > *str2) {
8.              return false;
9.          }
10.         ++str1;
11.         ++str2;
12.     }
13.
        return *str1 == '\0' && *str2 != '\0';
14. }
```

```
1.      Path oleg = "/home/aaaaaleg/";
2.      Path ivan = "/home/iiiiiivan/";
3.      bool f = oleg < ivan;
```

```
Program returned: 0
Program stdout
1
```

Операторы ввода/ввывода

```
1. std::ostream& operator<<(std::ostream& os, const Path& path) {
2.     return os <<'"'<< path.value.get() << '"';
3. }
4. 
   std::istream& operator>>(std::istream& is, Path& path) {
5.     char buffer[256];
6.     is >> buffer;
7.     path = buffer;
8. 
       return is;
9. }
```

Производные

```
1.  bool operator>(const Path& lhs, const Path& rhs) noexcept {
2.      return rhs < lhs;
3.  }
4.  bool operator<=(const Path& lhs, const Path& rhs) noexcept {
5.      return !(rhs < lhs);
6.  }
7.  bool operator>=(const Path& lhs, const Path& rhs) noexcept {
8.      return !(lhs < rhs);
9.  }
10. bool operator!=(const Path& lhs, const Path& rhs) noexcept {
11.     return !(lhs == rhs);
12. }
```