3 Strings und reguläre Ausdrücke

Aufgaben

23. Binärwerte in Strings konvertieren

Schreiben Sie eine Funktion, die einen Bereich von 8-Bit-Integern entgegennimmt (etwa als Array oder als Vektor) und einen String mit den hexadezimalen Darstellungen der Eingabedaten ausgibt. Die Funktion sollte in der Lage sein, Ausgaben in Groß- und in Kleinbuchstaben zu machen. Betrachten Sie dazu die folgenden Beispiele:

Eingabe: `{ 0xBA, 0xAD, 0xF0, 0x0D }`, Ausgabe: `"BAADF00D"` oder `"baadf00d"`

Eingabe: `{ 1,2,3,4,5,6 }`, Ausgabe: `"010203040506"`

24. Strings in Binärwerte konvertieren

Schreiben Sie eine Funktion, die einen String mit hexadezimalen Ziffern entgegennimmt und einen Vektor aus 8-Bit-Integern als numerische Deserialisierung des Stringinhalts zurückgibt. Schauen Sie sich dazu die folgenden Beispiele an:

Eingabe: `"BAADF00D"` oder `"baadf00d"`, Ausgabe: `{ 0xBA, 0xAD, 0xF0, 0x0D }`

Eingabe: `"010203040506"`, Ausgabe: `{ 1,2,3,4,5,6 }`

25. Großschreibung eines englischen Artikeltitels

Schreiben Sie eine Funktion, die eine Texteingabe in eine Version umwandelt, in der jedes Wort mit einem Großbuchstaben beginnt, alle andere Buchstaben aber kleingeschrieben sind (wie es bei Titeln im englischen Sprachraum üblich ist). Beispielsweise soll der Text `"the c++ challenger"` in `"The C++ Challenger"` umgewandelt werden.

26. Strings verknüpfen

Schreiben Sie eine Funktion, die eine Liste von Strings und ein Trennzeichen entgegennimmt und daraus einen neuen String erstellt, indem sie alle Eingabestrings verknüpft und dazwischen jeweils das Trennzeichen stellt. Hinter dem letzten Einzelstring darf das Trennzeichen nicht mehr erscheinen, und wenn kein Eingabestring bereitgestellt wird, muss die Funktion einen leeren String zurückgeben.

Beispiel:

Eingabe: `{ "this","is","an","example" }` und Trennzeichen `' '` (Leerzeichen)

Ausgabe: `"this is an example"`

27. Strings aufteilen

Schreiben Sie eine Funktion, die einen String und eine Liste möglicher Trennzeichen entgegennimmt, den String jeweils an den Trennzeichen in einzelne Token aufteilt und diese Token in einem Vektor ausgibt.

Beispiel:

Eingabe: `"this,is.a sample!!"` mit Trennzeichen `",.! "`

Ausgabe: `{"this", "is", "a", "sample"}`

28. Längster Palindrom-Teilstring

Schreiben Sie eine Funktion, die einen String entgegennimmt, darin die längste Zeichenfolge ermittelt, die ein Palindrom ist, und sie ausgibt. Befinden sich in dem String mehrere Palindrome derselben Länge, soll nur das erste ausgegeben werden.

29. Kfz-Kennzeichen validieren

Wir gehen hier von Kfz-Kennzeichen im Format `LLL-LL DDD` oder `LLL-LL DDDD` aus, wobei `L` ein Großbuchstabe von A bis Z und `D` eine Ziffer ist. Schreiben Sie folgende Funktionen:

- Eine Funktion, die prüft, ob ein Kfz-Kennzeichen das korrekte Format aufweist.
- Eine Funktion, die einen Text entgegennimmt und alle darin vorhandenen Kfz-Kennzeichen findet und ausgibt.

30. URL-Bestandteile extrahieren

Schreiben Sie eine Funktion, die einen URL-String entgegennimmt und daraus die einzelnen Teile der URL (Protokoll, Domäne, Port, Pfad, Abfrage und Fragment) entnimmt.

(...)

27. Strings aufteilen

Im Folgenden finden Sie zwei Versionen der Aufteilungsfunktion:

- Die erste Version nutzt ein einzelnes Zeichen als Trennzeichen. Um den Eingabestring aufzuteilen, verwendet sie einen Stringstream, der mit dem Inhalt des Eingabestrings initialisiert wird. Sie liest mit `std::getline()` Stücke aus dem String, bis sie das nächste Trennzeichen oder ein Zeilenendezeichen erreicht.
- Die zweite Version verwendet eine Liste möglicher Trennzeichen, die als String angegeben sind. Mit `std::string::find_first_of()` findet sie, beginnend an der gegebenen Position, die erste Stelle, an der irgendeines der Trennzeichen vorkommt. Dieser Vorgang wird in einer Schleife ausgeführt, bis der gesamte Eingabestring verarbeitet ist. Die extrahierten Teilstrings werden zu dem Ergebnisvektor hinzugefügt.

```
template <class Elem>
using tstring = std::basic_string<Elem, std::char_traits<Elem>,
                                  std::allocator<Elem>>;

template <class Elem>
using tstringstream = std::basic_stringstream<
   Elem, std::char_traits<Elem>, std::allocator<Elem>>;
template<typename Elem>
inline std::vector<tstring<Elem>> split(tstring<Elem> text,
                                        Elem const delimiter)
{
   auto sstr = tstringstream<Elem>{ text };
   auto tokens = std::vector<tstring<Elem>>{};
   auto token = tstring<Elem>{};
   while (std::getline(sstr, token, delimiter))
   {
      if (!token.empty()) tokens.push_back(token);
   }
   return tokens;
}

template<typename Elem>
inline std::vector<tstring<Elem>> split(tstring<Elem> text,
                                        tstring<Elem> const & delimiters)
{
   auto tokens = std::vector<tstring<Elem>>{};
   size_t pos, prev_pos = 0;
```

```
    while ((pos = text.find_first_of(delimiters, prev_pos)) !=
    std::string::npos)
    {
       if (pos > prev_pos)
       tokens.push_back(text.substr(prev_pos, pos - prev_pos));
       prev_pos = pos + 1;
    }
    if (prev_pos < text.length())
    tokens.push_back(text.substr(prev_pos, std::string::npos));
    return tokens;
}
```

Die beiden folgenden Codebeispiele zeigen, wie verschiedene Strings an einem einzigen oder mehreren Trennzeichen aufgeteilt werden können:

```
int main()
{
   using namespace std::string_literals;
   std::vector<std::string> expected{"this", "is", "a", "sample"};
   assert(expected == split("this is a sample"s, ' '));
   assert(expected == split("this,is a.sample!!"s, ",.! "s));
}
```