Practicum P1 — Herfst 2006 **Opgave 3**

Leerdoelen: Rijen, C-Strings, Functies

Strings

In C++ zijn er verschillende mogelijkheden om strings, rijtjes van letters, op te slaan. In deze opgave zullen we met zogenaamde C-Strings werken. Deze C-strings danken hun naam aan het feit dat ze ook al bestonden in de taal C. De programmeertaal C is de voorganger van C++.

Een C-strings in niets anders dan een rij van letters. Volgens afspraak zetten we altijd een `'\0'`-teken direct achter de echte letters in de C-string. Als de rij langer mocht zijn dan de gewone letters en het `'\0'`-teken, dan tellen de letters achter het `'\0'`-teken gewoonlijk niet mee. Als de C++-compiler een rij letters voor je vult met een gegeven string, dan zal dat `'\0'`-teken automatisch toegevoegd worden. Indien je zelf een rij vult met letters zul je dat `'\0'`-teken moeten toevoegen. In het computer Nederlands heet het `'\0'`-teken doorgaans een *null-character*.

Een van de gevolgen van deze afspraak is dat een C-string altijd (minstens) een langer moet zijn dan het aantal letters dat je ziet. Een aantal voorbeelden van C-strings:

```
char string1 [6]  = "Hallo";        // rij is precies lang genoeg
char string2 [80] = "Informatica";  // rij is veel langer dan nu nodig is
char string3 [42];                  // geen beginwaarde gegeven
char string4 []   = "algoritme";    // compiler bepaalt zelf de lengte
```

De rij met de naam `string1` zal dus precies 6 tekens lang zijn. De elementen van deze rij hebben de nummers 0 tot en met 5. De inhoud van de elementen is achtereenvolgens `'H'`, `'a'`, `'l'`, `'l'`, `'o'`, en `'\0'`. Als je per sé wil mag je de letters natuurlijk ook een voor een opschrijven (zoals je dat voor getallen gewend bent). In dit geval moet je het het null-character zelf expliciet toevoegen.

```
char moeilijk []  = {'m','o','e','i','l','i','j','k','\0'};
```

Met een twee dimensionale rij kun je een rij van strings maken.

```
const int StrLengte = 80;   // liever via een constante dan via waarde
char strings [3][StrLengte]
    = {"Het nijmeegse informatica instituut heet ICIS."
      ,"Algoritmen en programma's."
      ,""
      };
```

Het afdrukken van een C-string stopt bij een `'\0'`-teken. Zo zal

```
cout << string2 << ".\n";
```

De tekst `"Informatica."` afdrukken. Het `'\n'` teken is een zogenaamde *newline*. Deze newline zorgt ervoor dat het volgende teken links op een nieuwe regel afgedrukt gaat worden.

Net als gewone rijen kun je C-strings meegeven als functie argument, maar niet opleveren als resultaat. In tegenstelling tot normale rijen kun je van C-strings wel de lengte bepalen door te gaan zoeken naar het null-character:

```
int lengte(char rij [])
{
    int len = 0;
    for ( ; rij[len]!='\0'; len=len+1)
        ;
    return len;
}
```

Als je precies bent geeft de functie `lengte` natuurlijk niet de lengte van de rij, maar de lengte van het stuk van de rij dat je gebruikt. Het null-teken wordt hierbij ***niet*** meegeteld.

Rijen die je meegeeft aan een functie worden (in tegenstelling tot normale argumenten) niet gekopieerd naar een lokale variabele. Dat betekent dat je in de functie de meegegeven rij kunt veranderen. Zo kunnen we bijvoorbeeld een functie `void keerom(char rij [])` schrijven die de meegegeven C-string omdraait.

Opdrachten

- Implementeer als oefening de boven besproken functie `void keerom(char rij [])` . Schrijf 'n aantal aanroepen van deze functie om de correcte werking te controleren. Zorg dat je de verschillende mogelijkheden in de tests aan de orde komen (bijvoorbeeld lengte van de string is even, lengte is oneven en lengte is nul).

- Implementeer een functie `void vervangAlle (char string [], char oud [], char nieuw [])` . Deze functie moet in de gegeven `string` alle voorkomens van de tekst gegeven als `oud` vervangen door `nieuw`. Zo zal na de aanroep `vervangAlle(string2,"a","AA")` de gegeven string de inhoud `"InformAAticAA"` hebben.

  Zoals altijd splits je de opdracht natuurlijk in een aantal kleinere onderdelen. Een voor de hand liggende keuze zou hier kunnen zijn een herhaling van het vervangen van een voorkomen van het gegeven patroon. Het vervangen van een zo'n voorkomen kun je ook weer splitsen. Bijvoorbeeld in het opzoeken van het patroon, het opschuiven van de rest van de zin en het kopiëren van de nieuwe tekst naar de juiste plaats.

  Denk van te voren ook over de randvoorwaarden waaronder je functie werkt of zou moeten werken. Bijvoorbeeld mag de `string` leeg zijn (idem voor `oud` en `nieuw`)? Je mag in de functie `vervangAlle` wel aannemen dat de rij waarin de `string` staat voldoende lang is. Zorg er bij de aanroepen voor dat dit ook het geval is.

- Schrijf code om de functie `vervangAlle` te testen. Zorg dat alle 'rare' situaties in de tests aan bod komen. Denk bijvoorbeeld aan:

  - `oud` komt niet voor;
  - `nieuw` is langer dan `oud`;
  - `nieuw` is korter dan `oud`;
  - `oud` staat helemaal aan het begin van `string`;
  - `oud` staat helemaal aan het eind van `string`;
  - `string` is leeg;
  - `nieuw` is leeg;
  - `nieuw` bevat `oud`, bijvoorbeeld `oud` is `"a"` en `nieuw` is `"aa"`;
  - ...

  Doe dit door in een geneste loop verschillende strings, patronen en vervangingswaarden te proberen.

Inleveren

Vóór woensdag 11 oktober, 8:30 uur via Blackboard. De beoordeling van deze opgave telt niet mee voor je practicumcijfer. Als je de opgave op tijd inlevert krijg je feedback op je werk. Vergeet niet om **in** je uitwerking duidelijk je naam en die van je practicumpartner te vermelden. Zet de uitvoer van je programma als commentaar in het programma. Zo kunnen we direct het resultaat van de testen beoordelen.