1 Pointers begrijpen

Typ of kopieer het volgende programma naar een nieuw project in je ontwikkelomgeving.

1. Zet in commentaar achter elke toekenning wat je denkt dat de waarden zijn van de 4 variabelen na uitvoering van die toekenning.
2. Controleer je antwoorden door te single-steppen met de debugger, en daarbij de waarden te inspecteren. Verklaar eventuele verschillen.

```
int main ()
{
    int var1, var2;
    int *ptr1, *ptr2;

    ptr1 = &var1;
    ptr2 = &var2;

    *ptr1 = 63;
    *ptr2 = 9;

    *ptr1 = *ptr2;
    ptr1 = ptr2;
    *ptr1 = 78;

    return 0;
}
```

2 We value your call

Typ of kopieer onderstaand programma naar een nieuw project in je ontwikkelomgeving. Welke waarde van `a` wordt door het programma afgedrukt? Verklaar dit resultaat.

```
#include <iostream>

//call by value
void cbv(int param)
{
    param += 10;
}

int main()
{
    int a = 5;
```

```
    cbv(a);
    std::cout << "a: " << a << std::endl;
    return 0;
}
```

3 A good reference

Welke waarde van `a` wordt door dit programma afgedrukt?

```
#include <iostream>

//call by reference
void cbr(int &param)
{
    param += 10;
}

int main()
{
    int a = 5;
    cbv(a);
    std::cout << "a: " << a << std::endl;
    return 0;
}
```

4 These are a few of my favourite things

> Cream colored ponies
>
> And crisp apple strudels
>
> Door bells and sleigh bells
>
> And schnitzel with noodles
>
> Wild geese that fly with the moon on their wings
>
> These are a few of my favorite things

My favorite things - The Sound of Music

Als je meerdere waardes van verschillende types als een object wilt doorgeven kun je een `struct` gebruikt. In het volgende fragment wordt de struct `Favourites` gedefinieerd. In objecten van dit type kun je je favoriete letter, priemgetal en cijfer opslaan.

```
#include <iostream>

struct Favourites
{
    char letter;
    int prime;
    char digit;
};

int main()
{
    Favourites my_favourites;
    my_favourites.letter = 'J';
    my_favourites.prime = 11;
    my_favourites.digit = '9';

    std::cout << "Hey, my favourite letter is " << my_favourites.letter << std::endl;
}
```

De laatste regel in de functie main laat zien hoe je een struct-member uitleest.

4.1 Opdrachten

1. Implementeer onderstaande varianten van de functie `show_favourites` die een object van het type `Favourites` accepteert en volgende uitvoer weergeeft:

   ```
   Favourite letter: J
   Favourite prime: 11
   Favourite digit: 9
   ```

   De eerste variant accepteert een Favourite by value, de tweede by reference en de derde als pointer

   ```
   // pass by value
   void show_favourites_by_value(Favourites f)
   {
   }

   // pass by reference
   void show_favourites_by_reference(Favourites &f)
   {
   }

   // pass by pointer
   void show_favourites_by_pointer(Favourites *p)
   {
   ```

```
}
```

2. Schrijf nu een functie `set_prime` die de `prime`-member van de struct verandert. Beargumenteer op welke wijze je de struct doorgeeft aan de functie.

5 It's a brand new Day

Maak in onderstaand programma de main-functie af door een object van het type `Day` op de *heap* aan te maken en geheugen na gebruik weer vrij te geven.

```
#include <string>
#include <iostream>
using namespace std;

class Week
{
private:
    int _week_num;
public:
    Week(int num) : _week_num { num } {};
    int week_number() { return _week_num; }
};

class Day
{
private:
    string _name;
    Week *_week;
public:
    Day(string name, Week *week) : _name { name }
                                 , _week { week } {};
    string& name_of_day() { return _name; }
    Week* week() { return _week; }
};

int main() {
    Day *day = nullptr;

    // your code here
    // end of your code

    cout << "It's the " << day->name_of_day()
         << " in week " << day->week()->week_number()
         << "!" << endl;
```

```
    // your code here
    // end of your code

    return 0;
}
```

6 Don't leak

Bestudeer onderstaand voorbeeld:

```
#include <iostream>

int* multiply(int x, int y)
{
    int *result = new int;
    *result = x * y;
    return result;
}

int main(int argc, char **argv)
{
    int *two_times_three = multiply(2, 3);
    std::cout << "2 x 3 = " << *two_times_three << std::endl;
    return 0;
}
```

6.1 Opdrachten

1. Leg uit wat de programmeur van bovenstaande code is vergeten.
2. Voeg dit toe aan de code.
3. Denk na over het gebruik van de functie `multiply`. Kunnen andere programmeurs zonder uitleg met deze functie aan de slag?

7 I forgot my name!

Bestudeer onderstaand voorbeeld:

```
#include <iostream>

char *my_name()
{
    char myname[6] = { 'J', 'e', 'r', 'o', 'e', 'n' };
```

```
    return myname;
}

int main(int argc, char **argv)
{
    char *your_name = my_name();
    for(int i = 0; i < 6; ++i)
        std::cout << your_name[i];
    std::cout << std::endl;
}
```

7.1 Opdrachten

1. Beschrijf wat je verwacht wanneer je deze code uitvoert.
2. Compileer de code maar voer hem nog niet uit. Controlleer of je compiler waarschuwingen geeft. Zo ja, noteer deze en probeer ze te verklaren.
3. Voer de code nu daadwerkelijk uit. Gebeurt er wat je verwacht?
4. Hoe zou jij een array van characters vanuit een functie naar een andere kunnen overbrengen? Dit betekent dat de letters van de naam Jeroen moeten bruikbaar zijn in `main`.

8 C Strings

Voor C++, was er C. In deze taal werden strings als een array van karakters opgeslagen. Het einde van een string werd aangegeven door de waarde 0. Zo bestaat het woord "Llama" uit vijf letters. Maar om dit woord als een string op te slaan, hebben we zes `char`s nodig! De zesde `char` heeft als waarde 0. Dit NUL-character geeft het einde van de string weer. De volgende twee arrays bevatten dus precies dezelfde string:

```
char my_favourite_animal[] = "Llama"; // an array of 6 elements!
char harrys_favourite_animal[] = { 'L', 'l', 'a', 'm', 'a', 0 };
```

8.1 Opdrachten

1. Maak de while-loop in onderstaande functie `int string_length(char *str)` af. Deze functie retourneert de lengte van een string (exclusief het NUL-character). **Je mag de conditie in de while-loop niet aanpassen**.

```
int string_length(char *str) {
    int result = 0;
    while(*str) { // don't change this line!
```

```
        // your code here
    }
    return result;
}
```

Voorbeeld:

```
    string_length("Llama") == 5
    string_length("Guanaco") == 7
```

2. Gebruik jouw implementatie van `string_length` in volgend voorbeeld en verklaar de uitvoer. Is deze zoals je verwacht? Waarom wel, waarom niet?

```
void not_safe() {
    char lowercase[16];
    char uppercase[16];
    for(int i = 0; i < 16; ++i)
        lowercase[i] = 'a' + i;
    for(int i = 0; i < 16; ++i)
        uppercase[i] = 'A' + i;
    cout << string_length(lowercase) << endl;
    cout << string_length(uppercase) << endl;
}
```

3. Pas de functie `not_safe` aan opdat de uitvoer juist is, dus:

```
16
16
```

9 Buffer overflow

1. Schrijf een functie `void copy_string(char *from, char *to)` die de string `from` kopieert naar `to` en gebruik die in volgend voorbeeld:

```
int main(int argc, char **argv) {
    char llama[6];
    copy_string("Llama", llama);
    cout << llama << endl;
    return 0;
}
```

De gewenste uitvoer van dit programma is:

```
Llama
```

2. Wat gebeurt er wanneer de array `to` te klein is voor alle karakters van `from`?

3. **Optionele uitdaging:** De header `<cstring>` bevat de functie `strcpy( char* dest, const char* src )` om C strings te kopiëren. De ontwikkelaars van het besturingssysteem OpenBSD vonden deze functie niet veilig genoeg en ontwikkelden daarom volgend alternatief:

   ```
   strlcpy(char *dst, const char *src, size_t dstsize);
   ```

   Hier bevat `dstsize` de grootte van de array `dst`. Waarom is deze functie veiliger?

4. **Optionele uitdaging:** Waarom is `dstsize` van het type `size_t` en niet `int`?

10 Bring on the big uns

Je gaat nu een programma schrijven dat een number van standard input leest en deze in grote cijfers naar standard output schrijft. Hieronder zie je een voorbeeld van de uitvoer:

```
Enter a number (0 exits): 1337
  #   ####  ####  #####
 ##       #     #     #
  #   ####  ####     #
  #       #     #   #
 ###  ####  ####   #
Enter a number (0 exits): 0
```

10.1 Opdrachten

1. Maak een nieuw project aan in je favoriete IDE
2. Download op blackboard `big_uns.zip`.
3. Importeer of kopieer de code uit de bestanden `read_int.h`, `read_int.cpp` en `big_uns.cpp` in je project.
4. Vervang de implementatie van de functie `write_big` in `big_uns.cpp`. Deze functie ontvangt een getal `a` en moet vervolgens dat getal in grote cijfers naar het scherm schrijven. De grote cijfers staan in de globale array `big_digits`.

10.2 Code

read_int.h

```
#ifndef _READ_INT_H
#define _READ_INT_H
```

```
// Writes prompt to standard output and then
// waits for a user to enter a number.
//
// Returns the number read as an int, or 0
// if something went wrong.
int read_int(const char *prompt);

#endif
```

read_int.cpp

```
#include "read_int.h"
#include <iostream>

int read_int(const char *prompt) {
    int n = 0;
    std::cout << prompt;
    std::cin >> n;
    std::cin.get();
    return n;
}
```

big_uns.cpp

```
#include "read_int.h"
#include <iostream>
using namespace std;

// use this array to obtain the big digits
const char big_digits[10][5][6] =
{
    {
    " ### ",
    "#   #",
    "#   #",
    "#   #",
    " ### ",
    },
    {
    "  #  ",
    " ##  ",
    "  #  ",
    "  #  ",
    " ### ",
    },
    {
    "#### ",
    "    #",
```

```
" ### ",
"#    ",
"#####",
},
{
"#### ",
"    #",
"#### ",
"    #",
"#### ",
},
{
"#   #",
"#   #",
"#####",
"    #",
"    #",
},
{
"#####",
"#    ",
"#### ",
"    #",
"#### ",
},
{
" ####",
"#    ",
"#####",
"#   #",
" ### ",
},
{
"#####",
"    #",
"   # ",
"  #  ",
" #   ",
},
{
" ### ",
"#   #",
" ### ",
"#   #",
" ### ",
},
```

```
    {
    " ### ",
    "#   #",
    "#####",
    "    #",
    " ### ",
    },
};

//*************************************************************************
// Writes the number a in large digits to standard output
// Uses the global array big_digits
//
// Example: If a = 82, this function will output:
//
//    ###   ####
//   #   #      #
//    ###    ###
//   #   #  #
//    ###   #####
//
//*************************************************************************
void write_big(int a)
{
    std::cout << a << std::endl;
}

//*************************************************************************
int main()
{
    while(int a = read_int("Enter a number (0 exits): "))
    {
        write_big(a);
    }
    return 0;
}
```