Programming 1 (C#)

Week 2

inholland hogeschool

Program term 1.1 (Programming 1)

01 (wk 36)	Introduction C# / Visual Studio 2022 (Community), basic problem solving
02 (wk 37)	selection, methods
03 (wk 38)	loops, basic version control setup
04 (wk 39)	classes, enums, arrays
05 (wk 40)	public/private, fields/properties, values & references
06 (wk 41)	inheritance version control
07 (wk 42)	Repetition / practice exam
08 (wk-43)	*no classes*
09 (wk-44)	exam *(practical, computer)*
10 (wk-45)	-

Selectie

Selectie types

Selectie → een beslissingsstructuur

2-way (boolean) selectie

- → selectie is afhankelijk van eenboolean-value
- → if-then selectie
- → if-then-else selectie
- → nested selectie

Meerdere (waarde)selectie

- → selectie is afhankelijk van een integer/char/string-waarde

Example ‘if-then selection’

pseudocode

```
PassLimit = 55
read grade
if grade < PassLimit
      display "Failed"
```

```
class Program
{
  const int PassLimit = 55;

  static void Main(string[] args)
  {
    // read grade
    string input = Console.ReadLine();
    int grade = int.Parse(input);

    // grade below limit?
    if (grade < PassLimit)
    {
      Console.WriteLine("Failed");
    }
  }
}
```

‘if-then-else selection’

pseudocode

```
if <condition>
        <statement(s)>
else
        <statement(s)>
```

Example ‘if-then-else selection’

pseudocode

```
PassLimit = 55
read grade
if grade < PassLimit
        display "Failed"
else
        display "Passed"
```

```
class Program
{
  const int PassLimit = 55;

  static void Main(string[] args)
  {
    // read grade
    string input = Console.ReadLine();
    int grade = int.Parse(input);

    // grade below limit?
    if (grade < PassLimit)
    {
      Console.WriteLine("Failed");
    }
    else
    {
      Console.WriteLine("Passed");
    }
  }
}
```

‘nested selection’

pseudocode

```
if <condition>
      <statement(s)>
else
      if <condition>
            <statement(s)>
      else
            <statement(s)>
```

Example ‘nested selection’

pseudocode

```
PassLimit = 55
read grade
if grade < PassLimit
  display "Failed"
else
  if grade < 80
    display "Passed, ok"
  else
    display "Passed, good"
```

```
static void Main(string[] args)
{
  // read grade
  string input = Console.ReadLine();
  int grade = int.Parse(input);

  // grade below limit?
  if (grade < PassLimit)
  {
    Console.WriteLine("Failed");
  }
  else
  {
    if (grade < 80)
    {
      Console.WriteLine("Passed, ok");
    }
    else
    {
      Console.WriteLine("Passed, good");
    }
  }
}
```

Example ‘nested selection’ - alternative

pseudocode

```
PassLimit = 55
read grade
if grade < PassLimit
  display "Failed"
else if grade < 80
  display "Passed, ok"
else
  display "Passed, good"
```

```
static void Main(string[] args)
{
  // read grade
  string input = Console.ReadLine();
  int grade = int.Parse(input);

  // grade below limit?
  if (grade < PassLimit)
  {
    Console.WriteLine("Failed");
  }
  else if (grade < 80)
  {
    Console.WriteLine("Passed, ok");
  }
  else
  {
    Console.WriteLine("Passed, good");
  }
}
```

Exercise 1 – highest/lowest

Read two numbers. Show the highest value and the lowest value.

(highest value = …, lowest value = …)

Exercise – pseudocode

```
read number1, number2
if number1 > number2
     highest = number1
     lowest = number2
else
     highest = number2
     lowest = number1
display "highest: " + highest
display "lowest: " + lowest
```

Selection with multiple conditions

```
PassLimit = 55
read grade
if grade >= PassLimit AND grade < 60
      display "just passed..."
```

Truth table: AND

A	B	A AND B (A ∧ B)
0	0	0
0	1	0
1	0	0
1	1	1

```
// student can start internship?
bool studentInternship = firstYearDone && (totalCredits >= 100);
```

A-side B-side

Truth table: OR

A	B	A OR B (A ∨ B)
0	0	0
0	1	1
1	0	1
1	1	1

```
// student failed Programming 1?
bool failedProgramming1 = failedExam || failedAssignments;
```

A-side B-side

Truth table: NOT

A	NOT A (!A)
0	1
1	0

```
// student passed Programming 1?
bool passedExam = !failedExam;
```

‘multiple selection’

```
switch <expression>
      case <option1>:
            statement(s)
      case <option2>:
            statement(s)
      case <option3>:
            statement(s)
      default:
            statement(s)
```

Example ‘multiple selection’

```
read grade
switch grade
      case 'A':
            display "Excellent"
      case 'B', 'C':
            display "Well done"
      case 'D':
            display "Passed"
      case 'F':
            display "Failed"
      default:
            display "Invalid grade"
```

```
static void Main(string[] args)
{
  string txt = "";
  string grade = Console.ReadLine().ToUpper();
  switch (grade)
  {
    case "A":
      txt = "Excellent";
      break;
    case "B":
    case "C":
      txt = "Well done";
      break;
    case "D":
      txt = "Passed";
      break;
    case "F":
      txt = "Failed";
      break;
    default:
      txt = "Invalid grade";
      break;
  }

  Console.WriteLine(txt);
}
```

Ranged selection

```
switch <expression>
        case <range1>:
                statement(s)
        case <range2>:
                statement(s)
        default:
                statement(s)
```

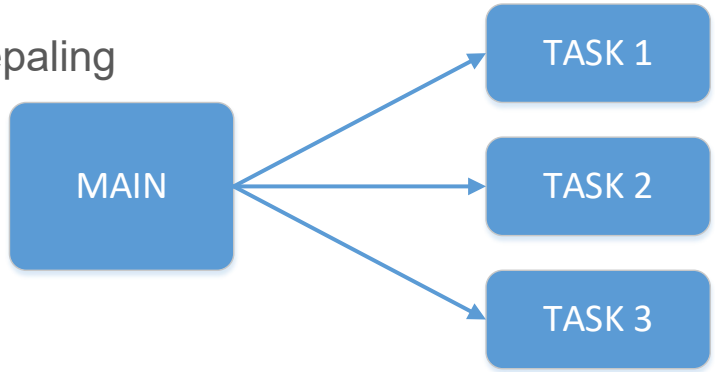
```
static void Main(string[] args)
{
    Console.Write("Enter your age: ");
    int age = int.Parse(Console.ReadLine());

    switch (age)
    {
        case < 0:
            Console.WriteLine("Negative age entered.");
            break;
        case > 120:
            Console.WriteLine("Invalid age entered.");
            break;
        default:
            Console.WriteLine($"Next year your age will be {age + 1}.");
            break;
    }
}
```

Methodes

Methodes – complexiteit van het programma

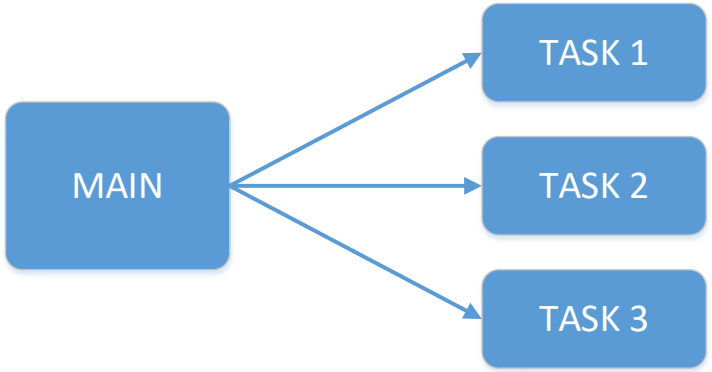
- De taak van een programmeur is om een programma te maken dat een probleem oplost.
- Een belangrijk element bij het oplossen van een probleem is het terugbrengen van de complexiteit van een probleem tot meerdere deelproblemen.
  - → Of om een taak op te splitsen in meerdere subtaken.
- Voorbeeld: Schrijf een programma dat de gebruiker instrueert om een jaar in te voeren, die invoer verwerkt en weergeeft of dat jaar een schrikkeljaar is.
- Taken:
  - → Druk een instructiebericht af, wacht op invoer van de gebruiker, verwerk de invoer van de gebruiker. De taak zal een jaar opleveren.
  - → Bepaal of een jaar een schrikkeljaar is. Deze taak levert een Boolean op.
  - → Geef een bericht weer dat afhankelijk is van de uitkomst van de schrikkeljaarbepaling



Methodes – complexiteit van het programma

- Losse taken vertalen we naar ‘methodes’.
- Een methode kan parameters krijgen
- Een methode kan een waarde (1, 2) retourneren of geen retourwaarde hebben (3). Voorbeeld:

1. int ReadYearFromConsole()
2. bool IsLeapYear(int year)
3. void DisplayLeapYearMessage(bool isLeapYear)



Opbouw van een applicatie

```
internal class Program
{
    static void Main(string[] args)
    {
        Program program = new Program();
        program.Start();
    }

    void Start()
    {
        // your code here...
    }

    // your other methods here...

}
```

Use this code for your Main (from now on)

You start coding here

Methode *(geen return waarde)*

```
Start method
read number
square = number * number
display square
WaitForUser()
```

```
WaitForUser()
read character
```

C# method call

```
void Start()
{
    Console.WriteLine("Enter an integer number: ");
    string input = Console.ReadLine();
    int number = int.Parse(input);

    int square = number * number;
    Console.WriteLine($"The square of {number} is {square}.");
    WaitForUser();
}

void WaitForUser()
{
    Console.ReadKey();
}
```

void (empty): no return value

Methode *(returns een waarde)*

```
Start method
squareOfFive = GetSquareOfFive()
display squareOfFive
```

```
GetSquareOfFive()
result = 5 * 5
return result
```

– Start method ‘takes’ the return value

C# method call

```
void Start()
{
    int squareOfFive = GetSquareOfFive();
    Console.WriteLine($"Result is {squareOfFive}.");
}

int GetSquareOfFive()
{
    int result = 5 * 5;
    return result;
}
```

```
Mi...
Result is 25.
```

C# method call

```
void Start()
{
    int squareOfFive = GetSquareOfFive();
    Console.WriteLine($"Result is {squareOfFive}.");
}

int GetSquareOfFive()
{
    int result = 5 * 5;
    return result;
}
```

Return value is an integer value

Formal/actual parameters

actual parameters (arguments)

formal parameters

```
Start method
read number1
read number2
product = GetProduct(number1, number2)
display product
```

```
GetProduct(num1, num2)
result = num1 * num2
return result
```

C# formal/actual parameters

```
void Start()
{
    Console.Write("Enter two integer numbers: ");
    int number1 = int.Parse(Console.ReadLine());
    int number2 = int.Parse(Console.ReadLine());

    int product = GetProduct(number1, number2);
    Console.WriteLine($"Product of {number1} and {number2} is {product}.");
}

int GetProduct(int num1, int num2)
{
    int result = num1 * num2;
    return result;
}
```

Microsoft Visu...

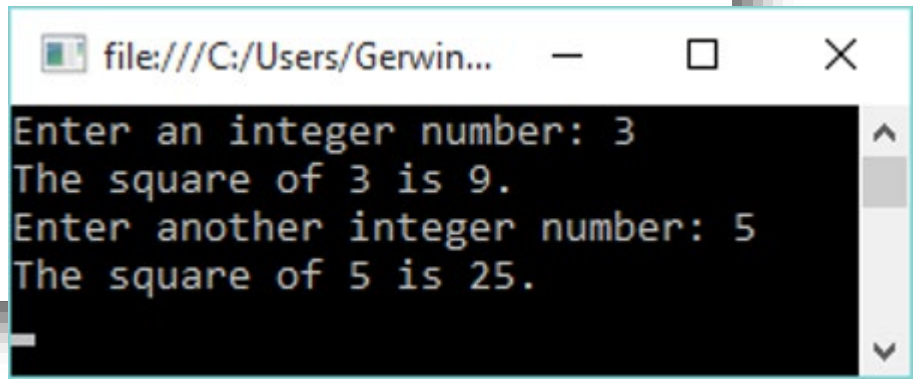
```
Enter two integer numbers:
5
7
Product of 5 and 7 is 35.
```

Code reuse

```
void Start()
{
    Console.Write("Enter an integer number: ");
    int number = int.Parse(Console.ReadLine());
    int square = GetSquare(number);
    Console.WriteLine($"The square of {number} is {square}.");

    Console.Write("Enter another integer number: ");
    number = int.Parse(Console.ReadLine());
    square = GetSquare(number);
    Console.WriteLine($"The square of {number} is {square}.");
}

int GetSquare(int number)
{
    return number * number;
}
```



Nested method call

```
void Start()
{
    Console.WriteLine("begin Start");
    ProcessNumbers();
    Console.WriteLine("end Start");
}

void ProcessNumbers()
{
    Console.WriteLine("..begin ProcessNumbers");
    ProcessNumber(5);
    ProcessNumber(7);
    Console.WriteLine("..end ProcessNumbers");
}

void ProcessNumber(int number)
{
    Console.WriteLine($"....begin ProcessNumber {number}");
    Console.WriteLine($"....end ProcessNumber {number}");
}
```

Microsoft ...

```
begin Start
..begin ProcessNumbers
....begin ProcessNumber 5
....end ProcessNumber 5
....begin ProcessNumber 7
....end ProcessNumber 7
..end ProcessNumbers
end Start
```

Scope / visibility

```
void Start()
{
    Console.Write("Enter two integer numbers: ");
    int number1 = int.Parse(Console.ReadLine());
    int number2 = int.Parse(Console.ReadLine());

    int product = GetProduct(number1, number2);
    Console.WriteLine($"Product of {number1} and {number2} is {product}.");
}

int GetProduct(int num1, int num2)
{
    int result = num1 * num2;
    return result;
}
```

number1, number2, product are local (not accessible by GetProduct)

result, num1, num2 are local (not accessible by Start)

Homework

– (practical class) Programming 1

- week 2 assignments → Moodle

- Please sign up for the Github acadamic program

inholland

hogeschool