Institute of Automation

C++ Basics and Applications in technical Systems

Lecture 2 - Flow control and user defined data-types

Institute of Automation
University of Bremen

02nd November 2012 / Bremen

WiSe 2012/2013

VAK 01-036

Universität Bremen

Overview

Lecture schedule

Institute of Automation

Time schedule

HK	**26. Oct.** - Introduction / Simple Program / Datatypes ...
HK	**02. Nov.** - Flow control / User-Defined Data types ...
CF	**09. Nov.** - Simple IO / Functions/ Modular Design ...
CF	**16. Nov.** - C++ Pointer
CF	**23. Nov.** - Object oriented Programming / Constructors
AL	**30. Nov.** - UML / Inheritance / Design principles
AL	**07. Dec.** - Namespace / Operators
AL	**14. Dec.** - Polymorphism / Template Classes / Exceptions
HK	**11. Jan.** - Design pattern examples

Universität Bremen

Important dates

Institute of Automation

Submission of exercises

1-3 **16. Nov.** - Deadline for submission of Exercise I, 13:00

4-6 **07. Dec.** - Deadline for submission of Exercise II, 13:00

For admission to final exam you need at least 50% of every exercise sheet.

Final project

1-9 **15. Feb.** - Deadline for submission of final project, 13:00

Final exam

1-9 **06. Feb.** - Final exam, 10:00-12:00, H3

Universität Bremen

Program structure

Institute of Automation

Statement to include function declarations

```
#include <iostream>
```

Structure of a simple program

```
int main()
{
  return 0;
}
```

Declaration statements

```
int iXValue;
unsigned int iYValue = 12;
```

Universität Bremen

Expressions

Statement with operator (expression)

```
iYValue += 10;
```

Simple data-types

int, **unsigned**, **bool**, **float**, **char**, ...

Statement block (one or more statements)

```
{
  Statement1;
  Statement2;
}
```

Functions and references

Function to get type-limits and sizes

```
sizeof(int);
numeric_limits<int>::max();
```

Type conversion by cast

```
static_cast<int>(bMyBoolean);
```

References of variables (a kind of alias)

```
int iMyInt;
int & iReferenceToMyInt = iMyInt;
```

Constant values

```
const int iMY_CONST_VALUE = 13;
```

if, else - Statement

If-Statement

Statement is executed if booleanExpression is true:

```
if (booleanExpression)
  Statement;
```

If-Else-Statement

Statement1 is executed if booleanExpression is true, otherwise Statement2:

```
if (booleanExpression)
  Statement1;
else
  Statement2;
```

If-Else-Statement with blocks

Statement1 and Statement2 are executed if booleanExpression is true, otherwise Statement3:

```
if (booleanExpression)
{
  Statement1;
  Statement2;
}
else
{
  Statement3;
}
```

Small exercise

Comparism of two natural numbers

Create a program that:

- asks the user to input two natural numbers
- compares both numbers
- displays which one is the bigger one

Example

Input	Output
„Please input value 1: “ 2 „Please input value 2: “ 4	4 is bigger than 2

Case selection with switch

- expression is evaluated, the result has to be of type integer or char
- constValueX is compared to the result of expression; if equal: statements are executed
- break has to be used to finish a case; without break the execution continues
- the statements after the label default are executed if no case fits the result of the evaluated expression

Example

```
switch (expression)
{
case constValue1:
  Statements1;
  break;

case constValue2:
  Statements2;
  break;

default:
  Statements;
}
```

Institute of Automation

Small exercise

A simple console menu using switch

Create a program that:

- Prompts the user to input one of the following keys: <c>, <s> or <t> (case insensitive)
- Displays the following string depending on the user intput:
  - c „Start calculation...“
  - s „Start program...“
  - t „Terminate program...“
  - all other keys will produce a „Unspecified input!“

Example

Input	Output
„Please choose <c>, <s> or <t>: “ s	Start calculation...

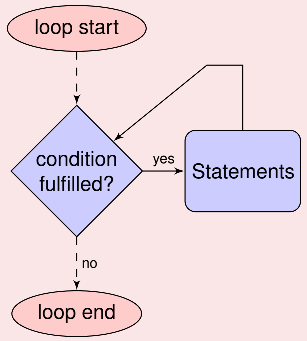
While-Do

The condition is checked before the first execution of the statements.

Example

```
while (condition)
{
  Statements;
}
```

Flow chart



Example

```
#include <iostream>
using std::cout;
using std::cin;
using std::endl;

void main()
{

  bool bFinish = false;
  char cInput;

  while (!bFinish)
  {

    cout << "Program termination with (T)" << endl;
    cin >> cInput;
    if (cInput == 'T' || cInput == 't')
      bFinish = true;

  }

}
```

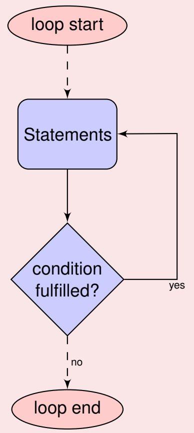
Do-While

The statements are executed once before the first condition check is performed.

Example

```
do
{
  Statements;
} while (condition);
```

Flow chart



Institute of Automation

Example

```
#include <iostream>
using std::cout;
using std::cin;
using std::endl;

void main()
{

  char cInput;

  do
  {

    cout << "Program termination with (T)" << endl;
    cin >> cInput;

  }
  while (cInput != 'T' && cInput != 't');

}
```

Universität Bremen

Loops with for

Institute of Automation

Structure

```
for (Initialization; Condition; Modification)
{
  Statements;
}
```

Example

```
const unsigned int iLIMIT = 1000;
double dArray[iLIMIT];
for (int nI = 0; nI < iLIMIT; nI++)
{
  std::cout << "Value [" << nI << "] ?";
  std::cin >> dArray[nI];
}
```

Universität Bremen

Small exercise

Institute of Automation

A simple console menu in a loop

Enhance the menu from exercise on slide 11:

- Use a loop to call the menu again and again until the user exits the application with <t>.

Universität Bremen

Non numerical ranges

Example

- A set of predefined colour values should be offered (red, green, blue).
- Possible auxiliary construction ⇒ usage of int variable:

```
int iColour;   // red = 0, green = 1, blue = 2
```

Disadvantage

- The meaning of the values has to be described as comment
- Bad implicit documentation:

```
if (iColour == 1)  // which colour is meant here?
if (iColour == 5)  // 5 is an undefined value!
```

Solution: Enumeration-types

Institute of Automation

Structure

```
enum typename {
  enumerations
} listOfVariables;
```

Example

```
enum Colour_T {
  RED,
  GREEN,
  BLUE
} colorValue1;
Colour_T colorValue2 = BLUE;
```

Example

```
switch (colorValue) {
case RED:
  Statements1;
  break;

case GREEN:
  Statements2;
  break;

case BLUE:
  Statements3;
  break;
}
```

Universität Bremen

More examples

Declaration of data-type and a variable

```
enum Colour_T {RED, GREEN, BLUE} value;
```

Definition of further variables

```
Colour_T can, bucket = BLUE;
```

Anonymous declaration

```
enum {BICYCLE, LORRY, CAR} vehicle;
```

Declaration with integrated definition

```
enum Palette_T {
WHITE = 0, GREY = 1, BROWN = 4, PURPLE = 8
} mixture;
```

Operations

Operations

Only the assignment operation is allowed for enum data-types. In other cases the value will be casted implicitly to `int`.

Example

```
int iI = RED;                              // possible (casting to int)
vehicle = CAR;                                                // correct
iI = RED + BLUE;                                             // possible
Colour_T bucket = iI;               // error, incompatible data type
BLUE = bucket;                         // error, BLUE is const value
mixture = WHITE + PURPLE;              // error, re-casting from int
                                    // to type Palette is impossible
mixture++;                                     // error, same reason
if (mixture > GREY)                                           // correct
  mixture = PURPLE;
```

User defined structured data-type

Created as composition of

- other user defined data-types
- standard data-types

Definition

- general declaration

```
struct [typename] {structure} [list of variables];
```

- recommended declaration

```
struct <typename> {structure};
```

Declaration, instantiation and access

Institute of Automation

Example

- general declaration

```
struct Point_T {
  int m_iXcoord;                      // Internal data elements
  int m_iYcoord;
  bool m_bVisible;
  Colour_T m_Colour;
};

Point_T newPoint;          // Instance of data-type Point_T
```

- Access on elements

```
newPoint.m_iXcoord = 270;
newPoint.m_iYcoord = 209;
newPoint.m_bVisible = true;
newPoint.m_Colour = BLUE;
```

Universität Bremen

Table of elements with unique data-type

Datatype

```
vector<dataType>
```

STL Header (Standard Template Library)

```
#include <vector>
```

Declaration (Example for int)

```
vector<int> myVector1(10);

vector<int> myVector2;
```

Data Access

Posibillity 1

```
vector<int> myVector(10);
myVector[0];
```

- no check for range under- or overflow
- crash during run-time upon access to non-existing element

Posibillity 2

```
vector<int> myVector(10);
myVector.at(0);
```

- check for range under- or overflow
- error message during runtime upon access to non-existing element

Index is zero based! A std::vector of size n is accessed by index in the range $[0, ..., n-1]$.

Reducing possiblity of faulty access

Institute of Automation

Use if-statement to check ranges of std::vector manually.

Example

```
std::vector<int> myVector(10);
int iPos = ...                    // An arbitrary assignment
                                  // for the position

if (iPos >= 0 && iPos < myVector.size())
{
  myVector[iPos] = ...            // Assignment only if iPos fits
                                  // current range
}
```

Universität Bremen

Initialization and assignment

Example

```
vector<float> costs(12);
```

Declaration and initialization of the vector size.

Example

```
vector<float> newCosts = costs;
```

Declaration of 2_{nd} vector and initialization (size and value) with first vector.

Example

```
vector<float> sortedCosts;
sortedCosts = costs;
```

Separate declaration and initialization.

Institute of Automation

Vectors are dynamic data-types

```
#include <iostream>                          // Inclusion of header files
#include <vector>
using std::cout;
using std::cin;
using std::vector;

int main() {

  vector<int> data;  // declaration of a vector variable (size 0)

  int iValue;
  do {
    cout << "Value (0=End): ";
    cin >> iValue;

    if (iValue != 0)
      data.push_back(iValue);     // append new value to vector

  } while(iValue != 0);

}
```

Universität Bremen

One rowed table of char elements

Datatype

```
string
```

STL Header (Standard Template Library)

```
#include <string>
```

Declaration

```
string sString1;
string sString2("Hello World!");
string sString3(sString2);
```

Examples for operations

```cpp
#include <iostream>
#include <string>
using std::cout;
using std::endl;
using std::string;
```

```cpp
void main() {
```

```cpp
  string sStr1("String1");
  for (int nI=0; nI < sStr1.size(); nI++) // characterwise output
    cout << sStr1.at(nI);                  // using checked access
  cout << endl;
  string sStr2(sStr1);                        // copy whole string
  sStr1 += sStr1;                    // concatenate strings with +=
  sStr2 = sStr1 + "Addition";            // addition and assignment
  sStr1 = 'A';                            // assign single character
  cout << sStr1 << sStr2 << endl;
```

```cpp
}
```

Exercise

Institute of Automation

A simple sorting program using vector

Create a program that:

- prompts the user to input three float values and stores them in a std::vector
- print the values in the reverse order to the screen

Example

Input	Output
„Please input value 1: “ 2	6
„Please input value 2: “ 4	4
„Please input value 3: “ 6	2

Universität Bremen