# IN204
# Programmation Orientée Objet – La surcharge des opérateurs

Séance de Travaux Dirigés du 9 et 16 octobre 2019

B. Monsuez

## Partie I – Surcharge d'opérateurs

**Question n°1**

Créer un projet dans lequel vous définissez une classe de nombre complexe qui aura typiquement la structure suivante :

```
#ifndef complexHPP
#define complexHPP

class Complex
{
private:
        double mRealPart;
        double mImaginaryPart;
public:
        Complex();
        ~Complex();
...
};

#endif
```

**Question n°1.1**

Ajouter à cette classe un ou plusieurs opérateurs de conversion convertissant un nombre flottant en un nombre complexe.

**Question n°1.2**

Ajouter à cette classe une fonction qui affiche le nombre complexe sur la console.

**Question n°1.3**

Créer deux fonctions statiques créant des nombre complexes, l'une créant un nombre complexe à partir d'une paire correspondant à la partie réelle et imaginaire, l'autre créant un nombre complexe à partir de la paire $(\rho, \theta)$ correspondant à la notation polaire de ce nombre.

**Question n°1.4**

Créer une variable globale ayant comme type la classe complexe et ayant pour valeur la valeur imaginaire 1.

**Question n°1.5**

Tester le bon comportement de votre classe sur des exemples simples. Notamment, est-il possible d'écrire désormais :

```
Complex complexValue = 3.3 + 5 * I;
```

**Question n°2**

Nous considérons les opérations de base simple que sont l'addition et la soustraction.

**Question n°2.1**

Proposer une surcharge des opérations + et –. Implanter ces dernières et tester.

```
Complex operator + (const Complex& aRightValue) const;
Complex operator - (const Complex& aRightValue) const;
```

**Question n°2.2**

Proposer une surcharge des opérations + et –. Implanter ces dernières et tester.

```
Complex operator + (double aLeftValue, const Complex& aRightValue);
Complex operator – (double a aLeftValue, const Complex& aRightValue);
```

Expliquer la différence avec les opérations précédentes ?

**Question n°2.3**

Tester le bon comportement de votre classe sur des exemples simples. Notamment, est-il possible d'écrire désormais :

```
Complex complexValue = 3.3 + 5 * I;
```

**Question n°2.4**

Proposer une surcharge des opérations += et –=. Implanter ces dernières et tester.

```
Complex& operator += (const Complex& aRightValue);
Complex& operator -= (const Complex& aRightValue);
```

Expliquer pourquoi les signatures des operations += et -= sont différentes de celles des operations + et - ?

**Question n°3**

Nous considérons les opérations de base que sont la multiplication et la division.

**Question n°3.1**

Proposer deux fonctions de conversion entre la représentation polaire et la représentation canonique des nombres complexes et implanter les.

**Question n°3.2**

Proposer une surcharge des opérations + et –. Implanter ces dernières et tester.

```
        Complex operator * (const Complex& aRightValue) const;
        Complex operator / (const Complex& aRightValue) const;
```

**Question n°3.3 (optionnel)**

Proposer une surcharge des opérations + et –. Implanter ces dernières et tester.

```
        Complex& operator *= (const Complex& aRightValue);
        Complex& operator /= (const Complex& aRightValue);
```

# Partie II – Manipulation des flux.

**Question n°1**

Nous considérons toujours notre classe de nombres complexes, nous souhaitons ne plus utiliser la méthode d'affichage sur la console mais définir une première méthode d'affichage sur les flux.

**Question n°1.1**

Nous considérons que l'affichage par défaut s'effectue selon le format

```
x + i y
```

Définissez une opération d'affichage d'un objet ayant pour type la classe Complex vers un flux.

Testez cette opération.

**Question n°1.2**

Nous souhaitons faire l'opération inverse, c'est-à-dire convertir la représentation d'un nombre complexe ayant l'une des représentations suivantes vers un nombre complexe.

```
x + i y
x - i y
i y
- i y
x
```

Définissez une opération de lecture d'un flux effectuant cette lecture d'un objet de type Complex.

Pour ce faire, il va falloir déterminer le format au fur et à mesure de la lecture des éléments à partir du flux. En Annexe, vous avez la document d'istream qui vous indiquera les fonctions qu'offrent la classe et qui vous seront utiles pour analyser le flux en entrée.

**Question n°1.3 (En dehors du cours !)**

Proposer la création d'un modificateur qui va permettre non pas d'écrire un petit « i » mais un grand « I » pour indiquer la partie imaginaire en lieu et place du petit « i ».

Plus d'information est disponible en complément. Ceci dépasse de loin l'objet du TD.

# Annexe – Document d'Input Stream

Input stream objects can read and interpret input from sequences of characters. Specific members are provided to perform these input operations (see functions below).

The standard object `cin` is an object of this type.

This is an instantiation of `basic_istream` with the following template parameters:

| template parameter | definition | comments |
|---|---|---|
| `charT` | `char` | Aliased as member `char_type` |
| `traits` | `char_traits<char>` | Aliased as member `traits_type` |

Objects of these classes keep a set of internal fields inherited from `ios_base` and `ios`:

| | field | member functions | description |
|---|---|---|---|
| *Formatting* | format flags | `flags`<br>`setf`<br>`unsetf` | A set of internal flags that affect how certain input/output operations are interpreted or generated.<br>See member type `fmtflags`. |
| | field width | `width` | Width of the next formatted element to insert. |
| | display precision | `precision` | Decimal precision for the next floating-point value inserted. |
| | locale | `getloc`<br>`imbue` | The `locale` object used by the function for formatted input/output operations affected by localization properties. |
| | fill character | `fill` | Character to pad a formatted field up to the *field width* (`width`). |
| *State* | error state | `rdstate`<br>`setstate`<br>`clear` | The current error state of the stream.<br>Individual values may be obtained by calling `good`, `eof`, `fail` and `bad`.<br>See member type `iostate`. |
| | exception mask | `exceptions` | The state flags for which a failure exception is thrown.<br>See member type `iostate`. |
| *Other* | callback stack | `register_callback` | Stack of pointers to functions that are called when certain events occur. |
| | extensible arrays | `iword`<br>`pword`<br>`xalloc` | Internal arrays to store objects of type `long` and `void*`. |
| | tied stream | `tie` | Pointer to output stream that is flushed before each i/o operation on this stream. |
| | stream buffer | `rdbuf` | Pointer to the associated `streambuf` object, which is charge of all input/output operations. |

To these, `istream` adds the *character count* (accessible using member `gcount`).

## Member types

The class contains the following member class:

**sentry**

Prepare stream for input (public member class )

Along with the following member types:

| member type | definition |
|---|---|
| char_type | char |
| traits_type | char_traits<char> |
| int_type | int |
| pos_type | streampos |
| off_type | streamoff |

| And these member types inherited from ios_base through ios: |
|---|
| **event** |
| Type to indicate event type (public member type ) |
| **event_callback** |
| Event callback function type (public member type ) |
| **failure** |
| Base class for stream exceptions (public member class ) |
| **fmtflags** |
| Type for stream format flags (public member type ) |
| **Init** |
| Initialize standard stream objects (public member class ) |
| **iostate** |
| Type for stream state flags (public member type ) |
| **openmode** |
| Type for stream opening mode flags (public member type ) |
| **seekdir** |
| Type for stream seeking direction flag (public member type ) |

## Public member functions

| **(constructor)** |
|---|
| Construct object (public member function ) |
| **(destructor)** |
| Destroy object (public member function ) |
| **Formatted input**: |
| **operator>>** |
| Extract formatted input (public member function ) |
| **Unformatted input**: |
| **gcount** |
| Get character count (public member function ) |
| **get** |
| Get characters (public member function ) |
| **getline** |
| Get line (public member function ) |
| **ignore** |
| Extract and discard characters (public member function ) |
| **peek** |
| Peek next character (public member function ) |
| **read** |
| Read block of data (public member function ) |
| **readsome** |
| Read data available in buffer (public member function ) |

| | |
|---|---|
| **putback** | |
| | Put character back (public member function ) |
| **unget** | |
| | Unget character (public member function ) |
| **Positioning**: | |
| **tellg** | |
| | Get position in input sequence (public member function ) |
| **seekg** | |
| | Set position in input sequence (public member function ) |
| **Synchronization**: | |
| **sync** | |
| | Synchronize input buffer (public member function ) |

## Protected member functions

| | |
|---|---|
| **operator=** | |
| | Move assignment (protected member function ) |
| **swap** | |
| | Swap internals (protected member function ) |

## Public member functions inherited from `ios`

| | |
|---|---|
| **good** | |
| | Check whether state of stream is good (public member function ) |
| **eof** | |
| | Check whether eofbit is set (public member function ) |
| **fail** | |
| | Check whether either failbit or badbit is set (public member function ) |
| **bad** | |
| | Check whether badbit is set (public member function ) |
| **operator!** | |
| | Evaluate stream (not) (public member function ) |
| **operator bool** | |
| | Evaluate stream (public member function ) |
| **rdstate** | |
| | Get error state flags (public member function ) |
| **setstate** | |
| | Set error state flag (public member function ) |
| **clear** | |
| | Set error state flags (public member function ) |
| **copyfmt** | |
| | Copy formatting information (public member function ) |
| **fill** | |
| | Get/set fill character (public member function ) |
| **exceptions** | |
| | Get/set exceptions mask (public member function ) |
| **imbue** | |
| | Imbue locale (public member function ) |
| **tie** | |
| | Get/set tied stream (public member function ) |
| **rdbuf** | |
| | Get/set stream buffer (public member function ) |

| | |
|---|---|
| **narrow** | |
| | Narrow character (public member function ) |
| **widen** | |
| | Widen character (public member function ) |

## Public member functions inherited from `ios_base`

| | |
|---|---|
| **flags** | |
| | Get/set format flags (public member function ) |
| **setf** | |
| | Set specific format flags (public member function ) |
| **unsetf** | |
| | Clear specific format flags (public member function ) |
| **precision** | |
| | Get/Set floating-point decimal precision (public member function ) |
| **width** | |
| | Get/set field width (public member function ) |
| **imbue** | |
| | Imbue locale (public member function ) |
| **getloc** | |
| | Get current locale (public member function ) |
| **xalloc** | |
| | Get new index for extensible array [static] (public static member function ) |
| **iword** | |
| | Get integer element of extensible array (public member function ) |
| **pword** | |
| | Get pointer element of extensible array (public member function ) |
| **register_callback** | |
| | Register event callback function (public member function ) |
| **sync_with_stdio** | |
| | Toggle synchronization with cstdio streams [static] (public static member function ) |