Ana Carolina da Silva Carvalho de Sousa
Carlos Adir Ely Murussi Leite

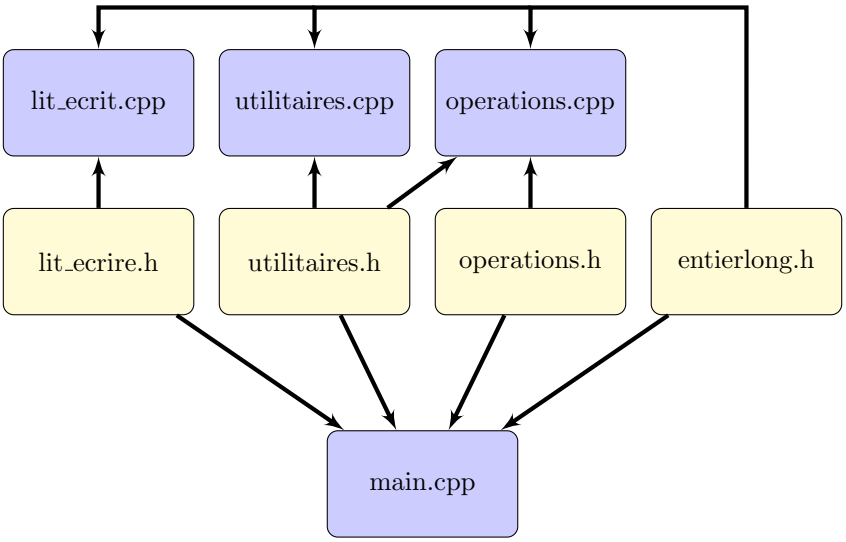
1 La architecture of files

There is a schematic below that show how the files are organized.

For exemple, inside the *operations.cpp*, there are the includes of *utilitaires.h*, *operations.h* and *entierlong.h*, while for the *utilitaires.cpp* it's included only *utilitaires.h* and *entierlong.h*.

With that diagram, there is no relation with classes and inheritance.

For the *main.cpp*, all the library are included.



2 Des fonctions

Nous avons au tout 9 fonctions, qui sont separées en 3 archives:

- lit_ecrit.h
- operations.h
- utilitaires.h

2.1 File lit_ecrit.h

Declaration of the 2 functions are:

```
void AfficheEntierLong(EntierLong);
EntierLong LitEntierLong();
```

2.1.1 Fonction AfficheEntierLong

Description .

This function shows the number given as argument. As the library **iostream** doesn't recognize the type EntierLong once we have created, we have to create a function to show the number instead only use the command **std::cout** as usual for default variables types(as **int**, **float**, and so on).

Use .

```
Entier a; // Declaration of variable
/* Here we change the value of a */
AfficheEntierLong(a); // We show the number a on the screen.
```

Parameters

- **EntierLong a:** EntierLong that we want to show on the screen.

Return

- None

Code .

```
void AfficheEntierLong(EntierLong a)
{
    //variables
    int i,j;
    // debut

    // signe
    if (a.Negatif)
    {
    cout <<"-";
    }
    // Chiffres
    /* on recherche le 1er chiffre non nul (ou le chiffre des unites dans le
        cas
    d'un entier nul*/
    i=MAXCHIFFRES-1;
    while ((a.Chiffres[i]==0)&&(i>0))
    {
     i=i-1;
    }
    /*on ecrit les Chiffres "utiles"*/
    for(j=i;j>=0;j=j-1)
    {
     cout << a.Chiffres[j];
    }
    cout << endl;
    // fin
}
```

2.1.2 Fonction LitEntierLong

Description .

This function read from the input(default keyboard) the number as the EntierLong. As the library **iostream** doesn't recognize the type EntierLong once we have created, we have to create a function to read the number instead only use the command **std::cin** as usual for default variables types(as **int**, **float**, and so on).

Use .

```
Entier a; // Declaration of variable
a = LitEntierLong(); // We put in a the number that we read.
```

Parameters

- None

Return

- **EntierLong a:** EntierLong that we read.

Code .

```
EntierLong LitEntierLong()
{
    //variables
    char Nb[MAXCHIFFRES+1];
    EntierLong a;
    int i,l;

    // debut
    cin >> Nb;
    a.Negatif = (Nb[0]=='-');
    l=strlen(Nb);
    if ((a.Negatif)||(Nb[0]=='+'))
    {
     //on decale le caractere de fin de chaine d'un indice a gauche
     for(i=0;i<l;i=i+1)
     {
      Nb[i]=Nb[i+1];
     }
     l=strlen(Nb);
    }
    for (i=0;i<l;i=i+1)
    {
     a.Chiffres[i] = toascii(Nb[l-1-i])-toascii('0');
    }
    // on complete par des 0
    for (i=l;i<MAXCHIFFRES;i=i+1)
    {
     a.Chiffres[i]=0;
    }
    return a;
    // fin
}
```

2.2 File utilitaires.h

```
EntierLong convertion_sl(int entierstand);
bool compare (EntierLong entier1, EntierLong entier2);
bool menor_abs (EntierLong entier1, EntierLong entier2);
```

2.2.1 Fonction convertion_sl

Description .

This function do the convertion of a **int** into a **EntierLong**, like we have the convertion

$$\text{float}(3)$$

It would be the same as

EntierLong(2993)

but with the argument given instead 2993.

Use .

```
EntierLong a; // Declaration of variable EntierLong
int b; // Declaration of a variable int
/* Here we change the value of b */
a = convertion_sl(b); // We convert the int b to EntierLong a.
```

Parameters

- **int n:** The **int** that we want to convert to EntierLong

Return

- **EntierLong n:** EntierLong converted from the **int** given

Code .

```

EntierLong convertion_sl(int entierstand)
{
 EntierLong n;
 int i;
 i= 0;
 int resto;

 if (entierstand <0)
 {
  n.Negatif = true;
  entierstand = -entierstand;
 }

 else
 {
  n.Negatif = false;
 }

 while (entierstand != 0)
 {
  resto = entierstand % 10;
  n.Chiffres [i]= resto;
  i ++;
  entierstand = entierstand/10;
  //std::cout << "(" << entierstand << "," << resto << std::endl;
 }
 for(; i<MAXCHIFFRES; i++)
 {
  n.Chiffres [i]= 0;
 }
 return n;
```

2.2.2 Function compare

Description .

This function do a comparation between its 2 arguments: $entier1$ and $entier2$. Verifying if they are equal.

$$result = \begin{cases} true, & \text{if } entier1 = entier2 \\ false, & \text{if } entier1 \neq entier2 \end{cases}$$

Use .

```
EntierLong a, b; // Declaration of 2 variables EntierLong
bool c; // boolean for get the result of comparation between a and b
/* Here we change the value of a and b */
c = compare(a, b); // The comparation between a and b.
```

Parameters

- **EntierLong entier1**
- **EntierLong entier2**

Return

- **bool result:** The result of comparing. **true** only if $entier1 = entier2$.

Code .

```

bool compare (EntierLong entier1, EntierLong entier2)
{
 int i;
 if (entier1.Negatif != entier2.Negatif)
 {
  return false;
 }
 for (i=0; i< MAXCHIFFRES; i++)
 {
  if(entier1.Chiffres[i] != entier2.Chiffres[i])
  {
   return false;
  }
 }
 return true;
```

2.2.3 Fonction menor_abs

Description .

This function do a comparation between its 2 arguments: $entier1$ and $entier2$. Verifying if the frist one is smaller(in module) than the second.

$$result = \begin{cases} true, & \text{if } |entier1| < |entier2| \\ false, & \text{if } |entier1| \geq |entier2| \end{cases}$$

Use .

```
EntierLong a, b; // Declaration of the variables we want to compare
bool c; // for get the result of comparation between a and b
/* Here we change the value of a and b */
c = menor_abs(a, b); // The comparation between a and b.
```

Parameters

- **EntierLong entier1**
- **EntierLong entier2**

Return

- **bool result:** The result of comparing. **true** only if $|entier1| < |entier2|$.

Code .

```
bool menor_abs (EntierLong entier1, EntierLong entier2)
{
 int i;
 for (i= MAXCHIFFRES-1; i>= 0; i--)
 {
  if (entier2.Chiffres[i] > entier1.Chiffres[i])
  {
   return true;
  }
  else if (entier2.Chiffres[i] < entier1.Chiffres[i])
  {
   return false;
  }
 }
 return false;
}
```

2.3 File operations.h

```
EntierLong add (EntierLong entier1, EntierLong entier2);
EntierLong sub (EntierLong entier1, EntierLong entier2);
EntierLong add_qq (EntierLong entier1, EntierLong entier2);
EntierLong sub_qq (EntierLong entier1, EntierLong entier2);
EntierLong mult (EntierLong entier1, EntierLong entier2);
EntierLong div (EntierLong entier1, EntierLong entier2);
```

2.3.1 Fonction add

Description .

This function do the sum of its 2 parameters only if they have the same signal. If they don't, so, the number 0 is given.

$$result = \begin{cases} entier1 + entier2 & \text{if } signal(entier1) = signal(entier2) \\ 0 & \text{if } signal(entier1) \neq signal(entier2) \end{cases}$$

Use .

```
EntierLong a, b, sum; //
/* Here we change the value of a and b */
sum = add(a, b); // The value of the sum of a + b if they have the same
    signal.
```

Parameters

- **EntierLong entier1**
- **EntierLong entier2**

Return

- **EntierLong entier3:** The result of the sum $(entier1 + entier2)$, or zero.

Code .

```
EntierLong add (EntierLong entier1, EntierLong entier2)
{
 EntierLong entier3;
 int i;

 for(i=0; i<MAXCHIFFRES; i++)
 {
  entier3.Chiffres [i]= 0;
 }

 if (entier1.Negatif == entier2.Negatif)
 {
  for (i= 0; i < MAXCHIFFRES; i++)
  {
   entier3.Chiffres [i] = entier3.Chiffres [i]+ entier1.Chiffres[i] + entier2
        .Chiffres[i];
   if (entier3.Chiffres [i] > 9)
   {
    if(i+1 < MAXCHIFFRES)
    {
     entier3.Chiffres [i+1] += entier3.Chiffres [i]/10;
    }
    entier3.Chiffres [i] %= 10;

   }
  }
 }
 else
 {
  //std:: cout << "Signals sont differents";
 }
 entier3.Negatif = entier1.Negatif;
 return entier3;
}
```

2.3.2 Fonction sub

Description .

This function do the subtraction of its 2 parameters only if they have the same signal and the frist one is bigger(in module) than the second. If they don't, so, the number 0 is given.

$$result = \begin{cases} 0 & \text{if } signal(entier1) \neq signal(entier2) \\ 0 & \text{if } |entier1| < |entier2| \\ entier1 - entier2 & \text{else} \end{cases}$$

Use .

```
EntierLong a, b, s;
/* Here we change the value of a and b */
s = sub(a, b); // The value of the sum of a - b if they have the same signal
    and abs(a) >= abs(b).
```

Parameters

- **EntierLong entier1**
- **EntierLong entier2**

Return

- **EntierLong entier3:** The result of the subtraction ($entier1 - entier2$), or zero.

Code .

```
EntierLong sub (EntierLong entier1, EntierLong entier2)
{
 bool teste;
 EntierLong entier3;
 int i;
 for(i=0; i<MAXCHIFFRES; i++)
 {
  entier3.Chiffres [i]= 0;

 }


 if (entier1.Negatif == entier2.Negatif)
 {
  teste = menor_abs (entier2, entier1);
  if (teste == true)
  {
   for (i= 0; i < MAXCHIFFRES; i++)
   {
    entier3.Chiffres [i] = entier3.Chiffres[i] + entier1.Chiffres[i] -
         entier2.Chiffres[i];
    while (entier3.Chiffres [i] < 0)
    {
     entier3.Chiffres [i] += 10;
     if(i + 1 < MAXCHIFFRES)
      entier3.Chiffres [i+1] --;
    }
   }
  }
 }
 entier3.Negatif = entier1.Negatif;


 return entier3;

}
```

2.3.3 Fonction add_qq

Description .

This function do the sum between its arguments whatever of value of its arguments.

$$result = entier1 + entier2$$

Use .

```
EntierLong a, b, sum;
/* Here we change the value of a and b */
sum = add_qq(a, b); // The value of the sum of a + b
```

Parameters

- **EntierLong entier1**
- **EntierLong entier2**

Return

- **EntierLong entier3:** The result of the sum of the parameters.

Code .

```
EntierLong add_qq (EntierLong entier1, EntierLong entier2)
{
 EntierLong a;
 if (entier1.Negatif == entier2.Negatif)
 {
  return add (entier1, entier2);
 }
 else
 {
  if (menor_abs (entier1, entier2))
  {
   if(entier1.Negatif == true)
    entier1.Negatif = false;
   else
    entier1.Negatif = true;
   a = sub(entier2, entier1);
   return a;
  }
  else
  {
   if(entier2.Negatif == true)
    entier2.Negatif = false;
   else
    entier2.Negatif = true;
   a= sub (entier1, entier2);
   return a;
  }

 }

}
```

2.3.4 Fonction sub_qq

Description .

This function do the subtraction between its arguments whatever of value of its arguments.

$$result = entier1 - entier2$$

Use .

```
EntierLong a, b, s;
/* Here we change the value of a and b */
s = sub_qq(a, b); // The value of the subtraction a - b
```

Parameters

- **EntierLong entier1**
- **EntierLong entier2**

Return

- **EntierLong a:** The result of the sum of the parameters.

Code .

```
EntierLong sub_qq (EntierLong entier1, EntierLong entier2)
{
 entier2.Negatif = !entier2.Negatif;
 return add_qq(entier1, entier2);
}
```

2.3.5 Fonction Mult

Description .

Cette fonction multiplie deux nombres donnés, quels que soient les signes. Pour cela, une norme générale a été utilisée en fonction des indices de nombre au format EntierLong.

Soit E le resultat de la multiplication, alors, nous pouvons ecrire

$$E = \sum_{i=0}^{MAXC.-1} 10^i e_i$$

où nous avons

$$e_i = \sum_{j=0}^{i} a_j \cdot b_{i-j}$$

Use .

```
EntierLong n1, n2; //
/* Here we change the value of n1 and n2 */
multi = mult(a, b); // The value of the sum of n1 + n2 if they have the same
    signal.
```

Parameters

- **EntierLong entier1**
- **EntierLong entier2**

Return

- **EntierLong entier3:** The result of the multiplication $(entier1 \cdot entier2)$.

Code .

```
EntierLong mult(EntierLong n1, EntierLong n2)
{
 EntierLong e;
 int i, j;
 if (n1.Negatif == n2.Negatif)
  e.Negatif = false;
 else
  e.Negatif = true;
 for(i = 0; i < MAXCHIFFRES; i++)
  e.Chiffres[i] = 0;
 for(i = 0; i < MAXCHIFFRES; i++)
 {
  for(j = 0; j <= i; j++)
  {
   e.Chiffres[i] += n1.Chiffres[j]*n2.Chiffres[i-j];
```

```
    if(e.Chiffres[i] > 9 && i+1 < MAXCHIFFRES)
    {
     e.Chiffres[i+1] += e.Chiffres[i]/10;
     e.Chiffres[i] %= 10;
    }
   }
  }
  return e;

}
```

2.3.6 Fonction Fibonacci

Description .

Cette fonction prend l'indice du terme de Fibonacci et calcule la valeur de la série pour ce terme. Le stockage se fait via le type EntierLong.

$$u_n = \begin{cases} 0 & \text{if} \quad n = 0 \\ 1 & \text{if} \quad n = 1 \\ u_{n-2} + u_{n-1} & \text{if } n \end{cases}$$

Use .

```
EntierLong n; //
/* Here we select the n term index of Fibonacci serie */
u_{n}= u_{n-2}+u_{n-1} // The value of the n term if n != 0 and i != 1.
```

Parameters

- **int n**

Return

- **EntierLong c:** The result of n term of Fibonacci serie (c), or 0 (if n= 0) or 1 (if n = 1) .

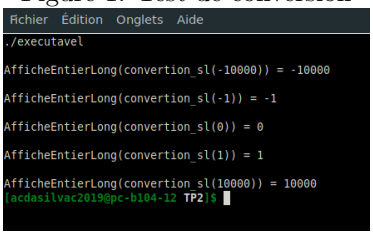
Code .

```
EntierLong Fibo (int n)
{
    EntierLong a,b,c;
    int i;

    a= convertion_sl (0);
    b= convertion_sl (1);
    if (n==0)
    {
        return a;
    }

    if (n==1)
    {
        return b;
    }
    for (i= 1; i<n; i++)
    {
        c= add_qq(a,b);
        a=b;
        b=c;
    }
    return c;
}
```

3 Tests

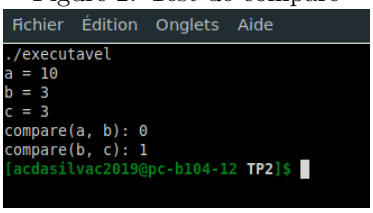
3.1 Test de conversion

Figure 1: Test de conversion



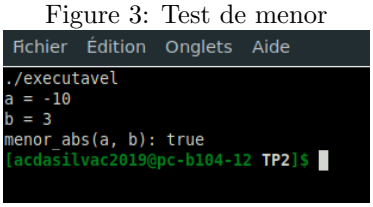
3.2 Test de compare

Figure 2: Test de compare



3.3 Test de menor_abs

Figure 3: Test de menor



3.4 Test de add_qq

Le test que faire la addition entre les argumenters, n'import qui sont eux.

Figure 4: Test de addition

```
Fichier  Édition  Onglets  Aide
./executavel
            a = 2345
            b = 1233
add_qq(a, b) = 3578
[acdasilvac2019@pc-b104-12 TP2]$
```

3.5 Test de sub_qq

Figure 5: Test de subtraction

```
Fichier  Édition  Onglets  Aide
./executavel
            a = 1233
            b = 2345
sub_qq(a, b) = -1112
[acdasilvac2019@pc-b104-12 TP2]$
```

3.6 Test de mult

Figure 6: Test de multiplication

```
Fichier  Édition  Onglets  Aide
./executavel
          a = 70
          b = 11
mult(a, b) = 770
[acdasilvac2019@pc-b104-12 TP2]$
```

3.7 Test de fibonacci

Figure 7: Test de fibonacci

```
Fichier  Édition  Onglets  Aide

./executavel
Fibo(0) = 0
Fibo(1) = 1
Fibo(2) = 1
Fibo(3) = 2
Fibo(4) = 3
Fibo(5) = 5
Fibo(10) = 55
Fibo(20) = 6765
Fibo(100) = 354224848179261915075
Fibo(1000) = 43466557686937456435688527675040625802564660517371780402481729089536555417949051890403879840079255169295922593080322634775209689623239873322471161642996440906533187938298969649928516003704476137795166849228875
Fibo(10000) = 157688908058783183404917434556270520223564846495196112460268313970975069382648706613264507665074611512677522748621598642530711298441182622661057163515069260029861704945425047491378115154139941550671256271197133252763631939606902895650288268608362241082050562430701794976171121233066073310059947366875
[acdasilvac2019@pc-b104-12 TP2]$
```

3.8 Test de fibonacci 2

Figure 8: Test de fibonacci 2