6 Les fonctions

Peter Schlagheck

Université de Liège

Ces notes ont pour seule vocation d'être utilisées par les étudiants dans le cadre de leur cursus au sein de l'Université de Liège. Aucun autre usage ni diffusion n'est autorisé, sous peine de constituer une violation de la Loi du 30 juin 1994 relative au droit d'auteur.

6 Les fonctions

6.1 Un exemple

On a besoin d'effectuer une certaine action plusieurs fois dans des contextes différents:

```
include <iostream>
using namespace std;

int main()
{
    ...
    ⟨action⟩;
    ...
    if ( ... )
    {
        ⟨action⟩;
        ...
    }
    ⟨action⟩;
    ...
}
```

où ⟨action⟩ effectue p. ex.

```
⟨instruction1⟩;
⟨instruction2⟩;
while ( ⟨expression⟩ )
{
    ⟨instruction3⟩;
    ⟨instruction4⟩;
}
⟨instruction5⟩;
```

...chaque fois avec d'autres variables

6.1 Un exemple d'une fonction

Programme pour calculer des nombres entiers k, l, m qui satisfont à $k^n + l^n = m^n$ pour un exposant $n > 0$ donné

$\longrightarrow$ problème de Fermat

Calcul de la puissance n d'un nombre entier:

```
int k, n;
cout << "entrez les deux entiers k et n:";
cin >> k >> n;
int kn = 1;
for ( int i = 1; i <= n; i++ )
  kn *= k;
cout << k << "^" << n << " = " << kn << endl;
```

6.1 Un exemple d'une fonction

Programme pour calculer des nombres entiers k, l, m qui satisfont à $k^n + l^n = m^n$ pour un exposant $n > 0$ donné

$\longrightarrow$ problème de Fermat

Calcul de la puissance n d'un nombre entier:

```
int k, n;
cout << "entrez les deux entiers k et n:";
cin >> k >> n;
int kn = 1;
for ( int i = 1; i <= n; i++ )
  kn *= k;
cout << k << "^" << n << " = " << kn << endl;
```

Exécution:

```
entrez deux entiers k et n: 3
4
3^4 = 81
```

6.1 Un exemple d'une fonction

Programme pour calculer des nombres entiers k, l, m qui satisfont à $k^n + l^n = m^n$ pour un exposant $n > 0$ donné

```
int n, Nmax;
cout << "exposant:";
cin >> n;
cout << "nombre maximal:";
cin >> Nmax;
for ( int k = 1; k <= Nmax; k++ )
for ( int l = 1; l <= Nmax; l++ )
for ( int m = 1; m <= Nmax; m++ )
{
  // calcul de k^n
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  // calcul de l^n
  int ln = 1;
  for ( int i = 1; i <= n; i++ )
    ln *= l;
  // calcul de m^n
  int mn = 1;
  for ( int i = 1; i <= n; i++ )
    mn *= m;
  if ( kn + ln == mn )
    cout << k << "^" << n << " + "
         << l << "^" << n << " = "
         << m << "^" << n << endl;
}
```

6.1 Un exemple d'une fonction

Programme pour calculer des nombres entiers k, l, m qui satisfont à $k^n + l^n = m^n$ pour un exposant $n > 0$ donné

```
int n, Nmax;
cout << "exposant:";
cin >> n;
cout << "nombre maximal:";
cin >> Nmax;
for ( int k = 1; k <= Nmax; k++ )
for ( int l = 1; l <= Nmax; l++ )
for ( int m = 1; m <= Nmax; m++ )
{
  // calcul de k^n
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  // calcul de l^n
  int ln = 1;
  for ( int i = 1; i <= n; i++ )
    ln *= l;
  // calcul de m^n
  int mn = 1;
  for ( int i = 1; i <= n; i++ )
    mn *= m;
  if ( kn + ln == mn )
    cout << k << "^" << n << " + "
         << l << "^" << n << " = "
         << m << "^" << n << endl;
}
```

Exécution:

```
exposant: 2
nombre maximal: 10
3^2 + 4^2 = 5^2
4^2 + 3^2 = 5^2
6^2 + 8^2 = 10^2
8^2 + 6^2 = 10^2
```

6.1 Un exemple d'une fonction

Programme pour calculer des nombres entiers k, l, m qui satisfont à $k^n + l^n = m^n$ pour un exposant $n > 0$ donné

```
int n, Nmax;
cout << "exposant:";
cin >> n;
cout << "nombre maximal:";
cin >> Nmax;
for ( int k = 1; k <= Nmax; k++ )
for ( int l = 1; l <= Nmax; l++ )
for ( int m = 1; m <= Nmax; m++ )
{
  // calcul de k^n
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  // calcul de l^n
  int ln = 1;
  for ( int i = 1; i <= n; i++ )
    ln *= l;
  // calcul de m^n
  int mn = 1;
  for ( int i = 1; i <= n; i++ )
    mn *= m;
  if ( kn + ln == mn )
    cout << k << "^" << n << " + "
         << l << "^" << n << " = "
         << m << "^" << n << endl;
}
```

Exécution:

```
exposant: 3
nombre maximal: 10
```

6.1 Un exemple d'une fonction

Programme pour calculer des nombres entiers k, l, m qui satisfont à $k^n + l^n = m^n$ pour un exposant $n > 0$ donné

```
int n, Nmax;
cout << "exposant:";
cin >> n;
cout << "nombre maximal:";
cin >> Nmax;
for ( int k = 1; k <= Nmax; k++ )
for ( int l = 1; l <= Nmax; l++ )
for ( int m = 1; m <= Nmax; m++ )
{
  // calcul de k^n
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  // calcul de l^n
  int ln = 1;
  for ( int i = 1; i <= n; i++ )
    ln *= l;
  // calcul de m^n
  int mn = 1;
  for ( int i = 1; i <= n; i++ )
    mn *= m;
  if ( kn + ln == mn )
    cout << k << "^" << n << " + "
         << l << "^" << n << " = "
         << m << "^" << n << endl;
}
```

Exécution:

```
exposant: 3
nombre maximal: 100
```

6.1 Un exemple d'une fonction

Programme pour calculer des nombres entiers k, l, m qui satisfont à $k^n + l^n = m^n$ pour un exposant $n > 0$ donné

```
int n, Nmax;
cout << "exposant:";
cin >> n;
cout << "nombre maximal:";
cin >> Nmax;
for ( int k = 1; k <= Nmax; k++ )
for ( int l = 1; l <= Nmax; l++ )
for ( int m = 1; m <= Nmax; m++ )
{
  // calcul de k^n
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  // calcul de l^n
  int ln = 1;
  for ( int i = 1; i <= n; i++ )
    ln *= l;
  // calcul de m^n
  int mn = 1;
  for ( int i = 1; i <= n; i++ )
    mn *= m;
  if ( kn + ln == mn )
    cout << k << "^" << n << " + "
         << l << "^" << n << " = "
         << m << "^" << n << endl;
}
```

Problème:

on doit écrire trois fois la même itération

Solution:

copier & coller dans l'éditeur...
ou la définition d'une fonction

6.1 Un exemple d'une fonction

Programme pour calculer des nombres entiers k, l, m qui satisfont à $k^n + l^n = m^n$ pour un exposant $n > 0$ donné

```
int n, Nmax;
cout << "exposant:";
cin >> n;
cout << "nombre maximal:";
cin >> Nmax;
for ( int k = 1; k <= Nmax; k++ )
for ( int l = 1; l <= Nmax; l++ )
for ( int m = 1; m <= Nmax; m++ )
  if ( puissance( k, n ) +
       puissance( l, n ) ==
       puissance( m, n ) )
    cout << k << "^" << n << " + "
         << l << "^" << n << " = "
         << m << "^" << n << endl;
```

Problème:

on doit écrire trois fois la même itération

Solution:

copier & coller dans l'éditeur...
ou la définition d'une fonction
`puissance( k, n )`

6.1 Un exemple d'une fonction

```
int puissance( int k, int n )
{
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  return kn;
}
```

6.1 Un exemple d'une fonction

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

int puissance( int k, int n )
{
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  return kn;
}

int main()
{
  int n, Nmax;
  cout << "exposant:";
  cin >> n;
  cout << "nombre maximal:";
  cin >> Nmax;
  for ( int k = 1; k <= Nmax; k++ )
  for ( int l = 1; l <= Nmax; l++ )
  for ( int m = 1; m <= Nmax; m++ )
    if ( puissance( k, n ) +
         puissance( l, n ) ==
         puissance( m, n ) )
      cout << k << "^" << n << " + "
           << l << "^" << n << " = "
           << m << "^" << n << endl;
}
```

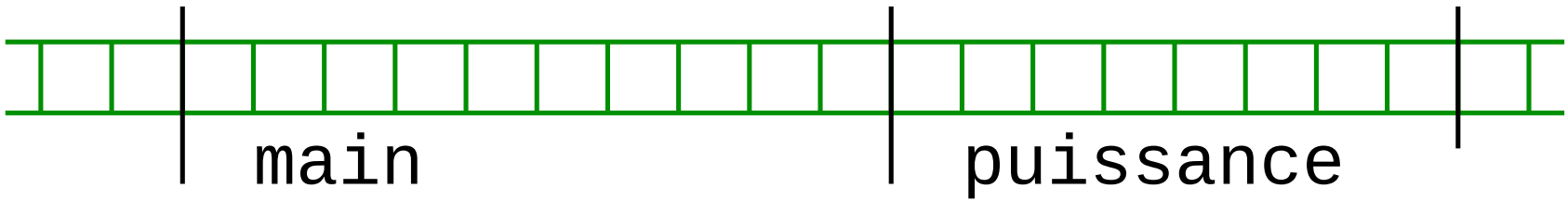
6.1 Un exemple d'une fonction

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

int puissance( int k, int n )
{
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  return kn;
}

int main()
{
  int k = 2;
  int n = 3;
  int r = puissance( k, n );
  int s = puissance( n, k );
  cout << r + s << endl;
}
```

Le flot du programme dans la mémoire:



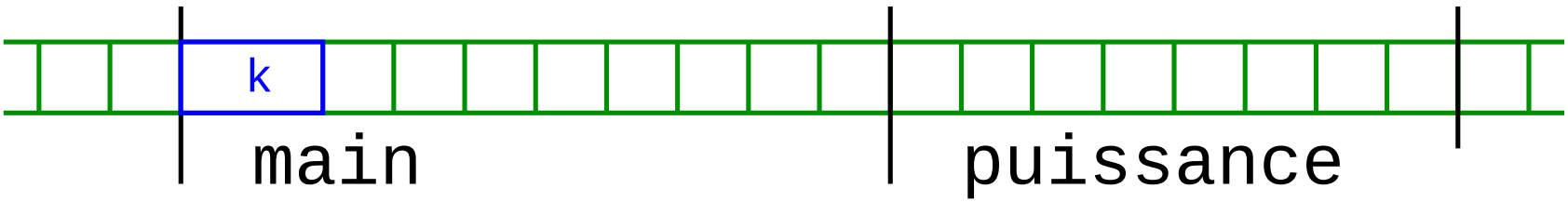
6.1 Un exemple d'une fonction

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

int puissance( int k, int n )
{
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  return kn;
}

int main()
{
  int k = 2;
  int n = 3;
  int r = puissance( k, n );
  int s = puissance( n, k );
  cout << r + s << endl;
}
```

Le flot du programme dans la mémoire:



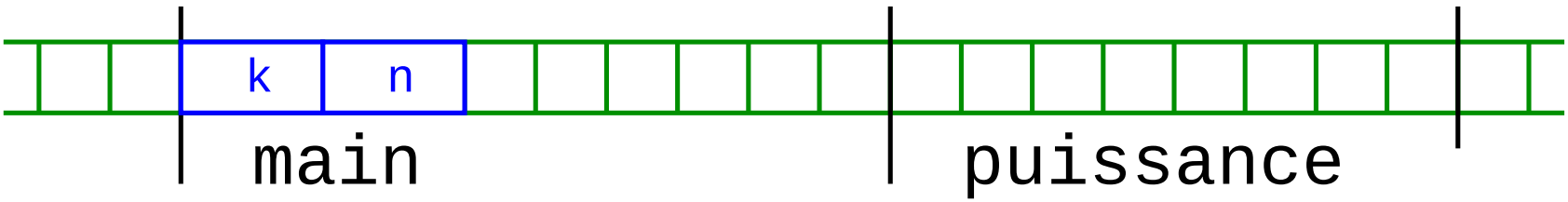
6.1 Un exemple d'une fonction

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

int puissance( int k, int n )
{
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  return kn;
}

int main()
{
  int k = 2;
  int n = 3;
  int r = puissance( k, n );
  int s = puissance( n, k );
  cout << r + s << endl;
}
```

Le flot du programme dans la mémoire:



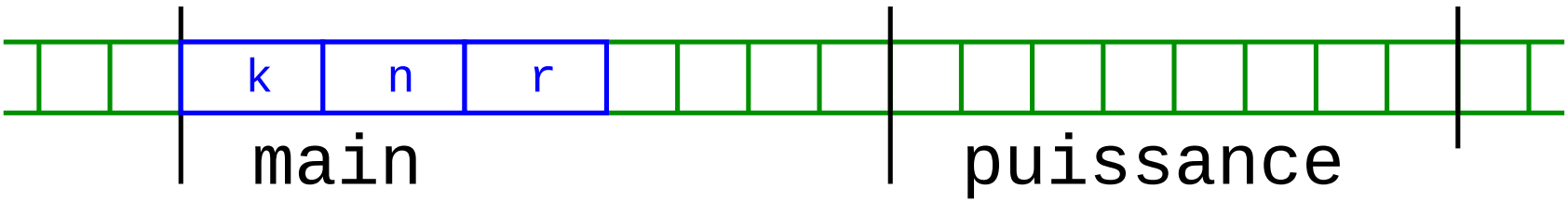
6.1 Un exemple d'une fonction

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

int puissance( int k, int n )
{
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  return kn;
}

int main()
{
  int k = 2;
  int n = 3;
  int r = puissance( k, n );
  int s = puissance( n, k );
  cout << r + s << endl;
}
```

Le flot du programme dans la mémoire:



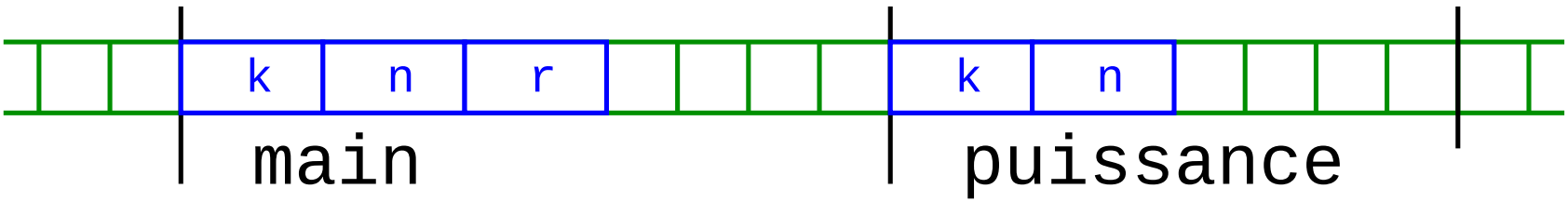
6.1 Un exemple d'une fonction

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

int puissance( int k, int n )
{
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  return kn;
}

int main()
{
  int k = 2;
  int n = 3;
  int r = puissance( k, n );
  int s = puissance( n, k );
  cout << r + s << endl;
}
```

Le flot du programme dans la mémoire:



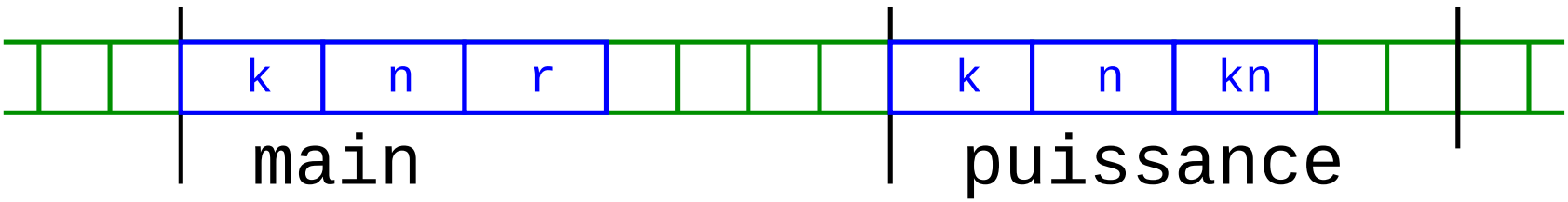
6.1 Un exemple d'une fonction

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

int puissance( int k, int n )
{
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  return kn;
}

int main()
{
  int k = 2;
  int n = 3;
  int r = puissance( k, n );
  int s = puissance( n, k );
  cout << r + s << endl;
}
```

Le flot du programme dans la mémoire:



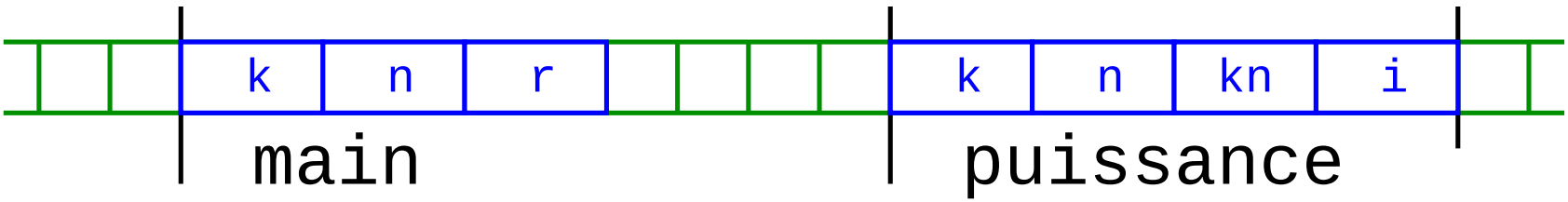
6.1 Un exemple d'une fonction

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

int puissance( int k, int n )
{
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  return kn;
}

int main()
{
  int k = 2;
  int n = 3;
  int r = puissance( k, n );
  int s = puissance( n, k );
  cout << r + s << endl;
}
```

Le flot du programme dans la mémoire:



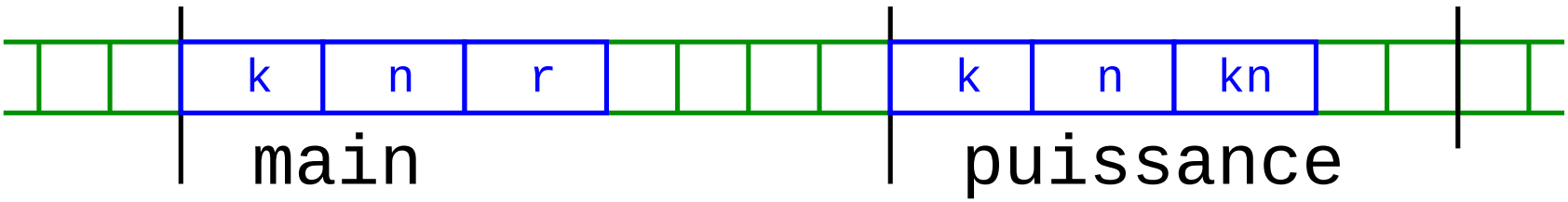
6.1 Un exemple d'une fonction

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

int puissance( int k, int n )
{
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  return kn;
}

int main()
{
  int k = 2;
  int n = 3;
  int r = puissance( k, n );
  int s = puissance( n, k );
  cout << r + s << endl;
}
```

Le flot du programme dans la mémoire:



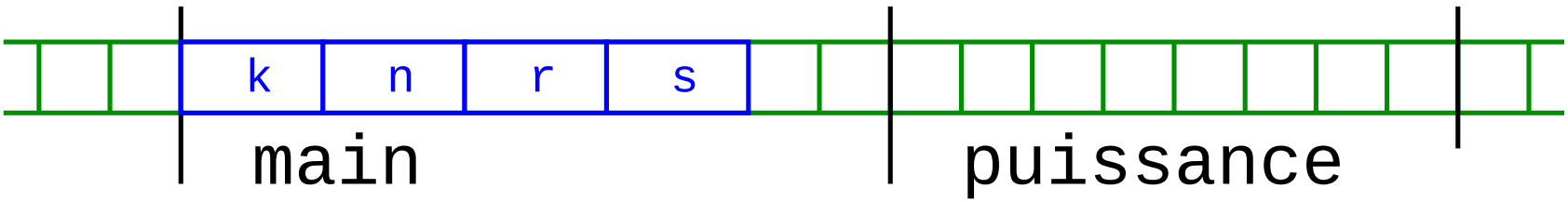
6.1 Un exemple d'une fonction

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

int puissance( int k, int n )
{
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  return kn;
}

int main()
{
  int k = 2;
  int n = 3;
  int r = puissance( k, n );
  int s = puissance( n, k );
  cout << r + s << endl;
}
```

Le flot du programme dans la mémoire:



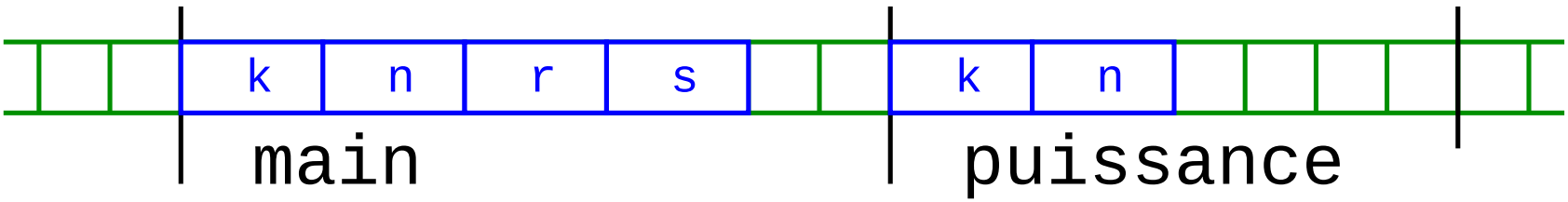
6.1 Un exemple d'une fonction

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

int puissance( int k, int n )
{
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  return kn;
}

int main()
{
  int k = 2;
  int n = 3;
  int r = puissance( k, n );
  int s = puissance( n, k );
  cout << r + s << endl;
}
```

Le flot du programme dans la mémoire:



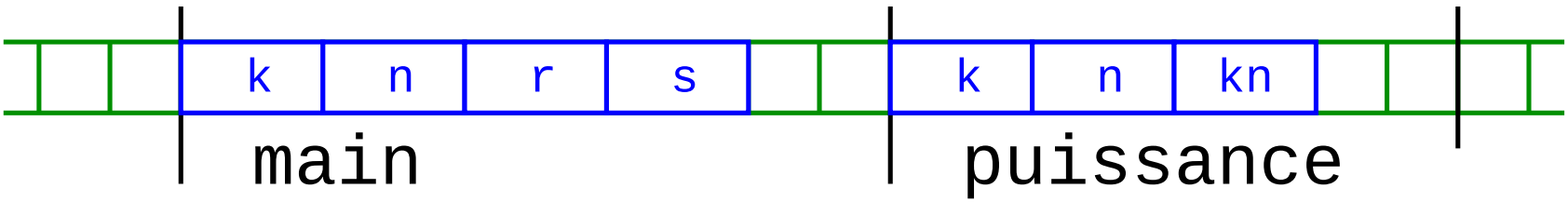
6.1 Un exemple d'une fonction

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

int puissance( int k, int n )
{
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  return kn;
}

int main()
{
  int k = 2;
  int n = 3;
  int r = puissance( k, n );
  int s = puissance( n, k );
  cout << r + s << endl;
}
```

Le flot du programme dans la mémoire:



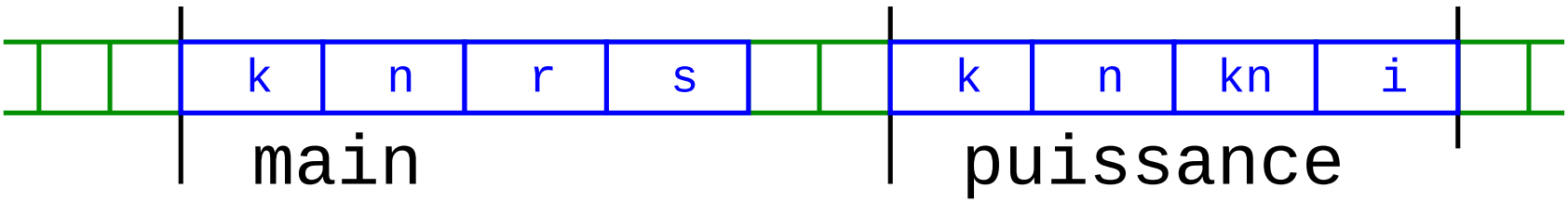
6.1 Un exemple d'une fonction

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

int puissance( int k, int n )
{
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  return kn;
}

int main()
{
  int k = 2;
  int n = 3;
  int r = puissance( k, n );
  int s = puissance( n, k );
  cout << r + s << endl;
}
```

Le flot du programme dans la mémoire:



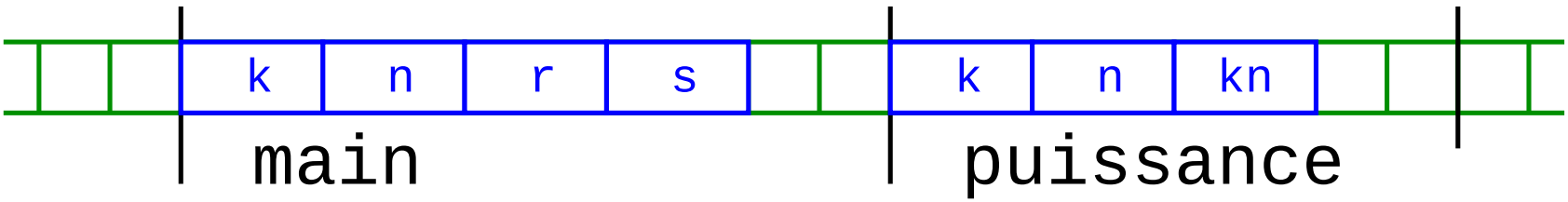
6.1 Un exemple d'une fonction

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

int puissance( int k, int n )
{
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  return kn;
}

int main()
{
  int k = 2;
  int n = 3;
  int r = puissance( k, n );
  int s = puissance( n, k );
  cout << r + s << endl;
}
```

Le flot du programme dans la mémoire:



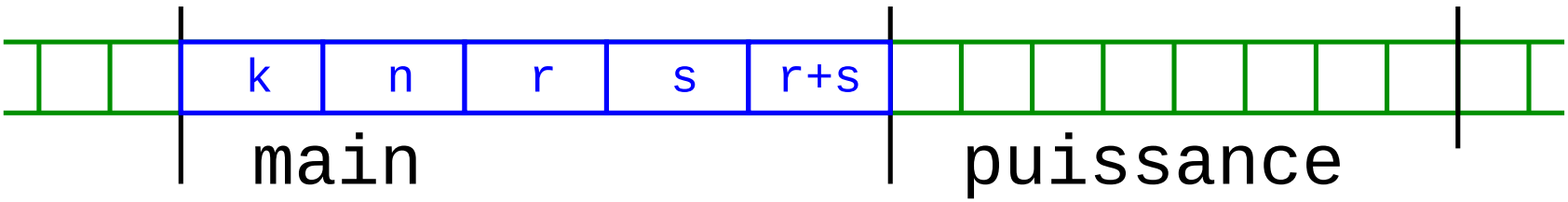
6.1 Un exemple d'une fonction

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

int puissance( int k, int n )
{
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  return kn;
}

int main()
{
  int k = 2;
  int n = 3;
  int r = puissance( k, n );
  int s = puissance( n, k );
  cout << r + s << endl;
}
```

Le flot du programme dans la mémoire:



Exécution de la fonction `puissance`

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

int puissance( int k, int n )
{
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  return kn;
}

int main()
{
  int k = 2;
  int n = 3;
  int r = puissance( k, n );
  int s = puissance( n, k );
  cout << r + s << endl;
}
```

Exécution:

```
17
```

Exécution de la fonction `puissance`

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

int puissance( int k, int n )
{
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  return kn;
}

int main()
{
  int k = 2;
  int n = 3;
  int r = puissance( 2, 3 );
  int s = puissance( 3, 2 );
  cout << r + s << endl;
}
```

Exécution:

```
17
```

Exécution de la fonction `puissance`

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

int puissance( int k, int n )
{
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  return kn;
}

int main()
{
  int k = 2;
  int n = 3;
  int r = puissance( k, n );
  int s = puissance( n + k, k );
  cout << r + s << endl;
}
```

Exécution:

```
33
```

Exécution de la fonction `puissance`

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

int puissance( int k, int n )
{
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  n = 6;
  k = 4;
  return kn;
}

int main()
{
  int k = 2;
  int n = 3;
  int r = puissance( k, n );
  int s = puissance( n, k );
  cout << r + s << endl;
  cout << n << " " << k << endl;
}
```

Exécution:

```
17
3 2
```

Exécution de la fonction `puissance`

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

int puissance( int k, int n )
{
  int kn = m;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  return kn;
}

int main()
{
  int k = 2;
  int n = 3;
  int m = 1;
  int r = puissance( k, n );
  int s = puissance( n, k );
  cout << r + s << endl;
}
```

⟶ `erreur: 'm' was not declared in this scope`

Exécution de la fonction `puissance`

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

int puissance( int k, int n )
{
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  return kn;
  kn = 0;
  return kn;
}

int main()
{
  int k = 2;
  int n = 3;
  int r = puissance( k, n );
  int s = puissance( n, k );
  cout << r + s << endl;
}
```

Exécution:

```
17
```

Exécution de la fonction `puissance`

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

int puissance( int k, int n )
{
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  return 0;
  return kn;
}

int main()
{
  int k = 2;
  int n = 3;
  int r = puissance( k, n );
  int s = puissance( n, k );
  cout << r + s << endl;
}
```

Exécution:

0

6.2 La structure générale d'une fonction

La définition d'une fonction:

⟨type⟩ ⟨nom⟩ (⟨type1⟩ ⟨nom1⟩, ⟨type2⟩ ⟨nom2⟩, ...)
{
    ⟨instruction1⟩;
    ⟨instruction2⟩;
    ...
    return ⟨expression_retour⟩;
}

⟶ ⟨expression_retour⟩ doit être du type ⟨type⟩
(ou convertible dans ce type-là)

6.2 La structure générale d'une fonction

La définition d'une fonction:

⟨type⟩ ⟨nom⟩ (⟨type1⟩ ⟨nom1⟩, ⟨type2⟩ ⟨nom2⟩, ...)
{
    ⟨instruction1⟩;
    ⟨instruction2⟩;
    ...
    return ⟨expression_retour⟩;
}

Appel de la fonction:

⟨nom⟩ (⟨expression1⟩, ⟨expression2⟩, ...)
    rend ⟨expression_retour⟩ comme valeur

⟶ ⟨expression1⟩ doit être du type ⟨type1⟩,
    ⟨expression2⟩ doit être du type ⟨type2⟩, ...
    (conversions implicites possibles)

6.2 La structure générale d'une fonction

L'appel de cette fonction corresponderait au code

⟨type⟩ ⟨nom⟩;
{
    ⟨type1⟩ ⟨nom1⟩ = ⟨expression1⟩;
    ⟨type2⟩ ⟨nom2⟩ = ⟨expression2⟩;
    ...
    ⟨instruction1⟩;
    ⟨instruction2⟩;
    ...
    ⟨nom⟩ = ⟨expression_retour⟩;
}

dans la partie du programme où cette fonction est appelée

(à l'exception qu'on ne pourrait pas utiliser d'autres variables du programme principal dans ce bloc-là)

6.2 La structure générale d'une fonction

La définition d'une fonction:

⟨type⟩ ⟨nom⟩ (⟨type1⟩ ⟨nom1⟩, ⟨type2⟩ ⟨nom2⟩, ...)
{
    ⟨instruction1⟩;
    ⟨instruction2⟩;
    ...
    return ⟨expression_retour⟩;
}

Les variables ⟨nom1⟩, ⟨nom2⟩, ... sont implicitement définies au début du bloc de la fonction (définition locale)

6.2 La structure générale d'une fonction

La définition d'une fonction:

⟨type⟩ ⟨nom⟩ (⟨type1⟩ ⟨nom1⟩, ⟨type2⟩ ⟨nom2⟩, ...)
{
    ⟨instruction1⟩;
    ⟨instruction2⟩;
    ...
    return ⟨expression_retour⟩;
}

Appel de la fonction:

⟨nom⟩ (⟨expression1⟩, ⟨expression2⟩, ...)
    rend ⟨expression_retour⟩ comme valeur

⟶ ⟨expression1⟩, ⟨expression2⟩, ... sont donc données à la fonction comme des copies

6.2 La structure générale d'une fonction

La définition d'une fonction:

```
⟨type⟩ ⟨nom⟩ ( ⟨type1⟩ ⟨nom1⟩, ⟨type2⟩ ⟨nom2⟩, ... )
{
    ⟨instruction1⟩;
    ⟨instruction2⟩;
    ...
    return ⟨expression_retour⟩;
}
```

l'instruction

```
return ⟨expression_retour⟩;
```

termine l'exécution de la séquence d'instructions de la fonction et rend la valeur de l'expression ⟨expression_retour⟩

$\longrightarrow$ il peut y avoir plusieurs instructions `return` dans une fonction

6.2 La structure générale d'une fonction

La définition d'une fonction:

⟨type⟩ ⟨nom⟩ (⟨type1⟩ ⟨nom1⟩, ⟨type2⟩ ⟨nom2⟩, ...)
{
    ⟨instruction1⟩;
    ⟨instruction2⟩;
    ...
    return ⟨expression_retour⟩;
    ...
    return ⟨expression_retour2⟩;
    ...
}

Exécution de la fonction `puissance`

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

int puissance( int k, int n )
{
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  return kn;
}

int main()
{
  int k = -2;
  int n = -3;
  int r = puissance( k, n );
  int s = puissance( n, k );
  cout << r + s << endl;
}
```

Exécution:

2

Exécution de la fonction `puissance`

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

int puissance( int k, int n )
{
  if ( n < 0 )
  {
    cout << "exposant negatif!" << endl;
    return 0;
  }
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  return kn;
}

int main()
{
  int k = -2;
  int n = -3;
  int r = puissance( k, n );
  int s = puissance( n, k );
  cout << r + s << endl;
}
```

Exécution:

```
exposant negatif!
exposant negatif!
0
```

Exécution de la fonction `puissance`

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

int puissance( int k, int n )
{
  if ( n < 0 )
  {
    cerr << "exposant negatif!" << endl;
    return 0;
  }
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  return kn;
}

int main()
{
  int k = -2;
  int n = -3;
  int r = puissance( k, n );
  int s = puissance( n, k );
  cout << r + s << endl;
}
```

Exécution:

```
exposant negatif!
exposant negatif!
0
```

Vos libertés en tant que programmeurs en C/C++

→ dans une fonction, vous pouvez faire tout ce qui est aussi possible dans `main()`

⟶ définir des variables,
placer des branchement et des boucles,
...

Vos libertés en tant que programmeurs en C/C++

- → dans une fonction, vous pouvez faire tout ce qui est aussi possible dans `main()`
- → vos fonctions peuvent rendre n'importe quel type

  ($\longrightarrow$ des entiers, des flottants, des caractères, ...)

Vos libertés en tant que programmeurs en C/C++

- → dans une fonction, vous pouvez faire tout ce qui est aussi possible dans `main()`
- → vos fonctions peuvent rendre n'importe quel type
- → elles peuvent contenir autant d'arguments que vous voulez

  ⟶ même pas d'arguments de tout !

Vos libertés en tant que programmeurs en C/C++

- → dans une fonction, vous pouvez faire tout ce qui est aussi possible dans `main()`
- → vos fonctions peuvent rendre n'importe quel type
- → elles peuvent contenir autant d'arguments que vous voulez

```
int maxint()
{
  int i = 1;
  while ( i > 0 )
    i++;
  i--;
  return i;
}
```

Appel:

```
cout << maxint() << endl;
```

```
2147483647
```

Vos libertés en tant que programmeurs en C/C++

- → dans une fonction, vous pouvez faire tout ce qui est aussi possible dans `main()`
- → vos fonctions peuvent rendre n'importe quel type
- → elles peuvent contenir autant d'arguments que vous voulez
- → vous pouvez choisir n'importe quels noms pour vos fonctions, tant qu'il n'y ait pas d'ambiguité avec d'autres fonctions en ce qui concerne leurs noms et leurs listes d'arguments !

  Par ce fait, les deux fonctions

```
int puissance( int k, int n )
{ ... }
```

```
int puissance( double k, int n )
{ ... }
```

  peuvent coexister dans le même programme.

6.3 Les routines

Des routines sont des fonctions qui ne rendent pas de valeur

et qui sont appelées comme des instructions
(et non pas comme des expressions)

Structure générale:

```
void ⟨nom⟩ ( ⟨type1⟩ ⟨nom1⟩, ⟨type2⟩ ⟨nom2⟩, ... )
{
    ⟨instruction1⟩;
    ⟨instruction2⟩;
    ...
    return;
}
```

Appel de la routine:

```
⟨nom⟩ ( ⟨expression1⟩, ⟨expression2⟩, ... );
```

6.3 Les routines

Des routines sont des fonctions qui ne rendent pas de valeur

et qui sont appelées comme des instructions
(et non pas comme des expressions)

Structure générale:

```
void ⟨nom⟩ ( ⟨type1⟩ ⟨nom1⟩, ⟨type2⟩ ⟨nom2⟩, ... )
{
    ⟨instruction1⟩;
    ⟨instruction2⟩;
    ...
    return;
}
```

$\longrightarrow$ `void` signifie un type “vide”

$\longrightarrow$ pas de valeur de retour
l’instruction `return;` n’est même pas nécessaire

6.3 Les routines

Des routines sont des fonctions qui ne rendent pas de valeur

et qui sont appelées comme des instructions
(et non pas comme des expressions)

Structure générale:

```
void ⟨nom⟩ ( ⟨type1⟩ ⟨nom1⟩, ⟨type2⟩ ⟨nom2⟩, ... )
{
    ⟨instruction1⟩;
    ⟨instruction2⟩;
    ...
}
```

6.3 Les routines

Exemple: input des chiffres entre 0 et 9

```
int main()
{
  int k, l, m;
  cout << "tapez trois chiffres:";
  cin >> k >> l >> m;
  check_number( k );
  check_number( l );
  check_number( m );
  ...
}
```

$\longrightarrow$ message d'alerte dans `check_number`
si `k`, `l` ou `m` ne sont pas contenus entre 0 et 9

6.3 Les routines

```
void check_number( int n )
{
  if ( n < 0 )
    cout << "warning: " << n
         << " ne doit pas etre negatif" << endl;
  if ( n > 9 )
    cout << "warning: " << n
         << " doit etre plus petit que 10" << endl;
}

int main()
{
  int k, l, m;
  cout << "tapez trois chiffres:";
  cin >> k >> l >> m;
  check_number( k );
  check_number( l );
  check_number( m );
  ...
}
```

6.3 Les routines

Manipulation des variables dans des routines
(ou dans des fonctions):

$\longrightarrow$ utilisation de l'opérateur `&`
dans la définition de la routine (ou de la fonction)

⟨type⟩ ⟨nom⟩ (⟨type1⟩ ⟨nom1⟩,
⟨type2⟩ &⟨nom2⟩,
⟨type3⟩ ⟨nom3⟩, ...)
{
...
}

$\longrightarrow$ la fonction ⟨nom⟩ travaille avec l'original ,
et non pas avec la copie, du deuxième argument

6.3 Les routines

Manipulation des variables dans des routines
(ou dans des fonctions):

$\longrightarrow$ utilisation de l'opérateur `&`
dans la définition de la routine (ou de la fonction)

⟨type⟩ ⟨nom⟩ (⟨type1⟩ ⟨nom1⟩,
⟨type2⟩ &⟨nom2⟩,
⟨type3⟩ ⟨nom3⟩, ...)
{
...
}

Appel:

⟨nom⟩ (⟨expression1⟩, ⟨variable2⟩, ⟨expression3⟩, ...)

$\longrightarrow$ ⟨variable2⟩ peut être manipulée par la fonction ⟨nom⟩

Une routine d'échange de deux variables

```
int main()
{
  cout << "2 nombres:";
  int a, b;
  cin >> a >> b;
  if ( a < b )
    echange( a, b );
  // 'echange' devrait echanger a et b
  cout << a << " > " << b << endl;
}
```

Une routine d'échange de deux variables

```
void echange( int &a, int &b )
{
  int c = a;
  a = b;
  b = c;
}

int main()
{
  cout << "2 nombres:";
  int a, b;
  cin >> a >> b;
  if ( a < b )
    echange( a, b );
  cout << a << " > " << b << endl;
}
```

Une routine d'échange de deux variables

```
void echange( int &a, int &b )
{
  int c = a;
  a = b;
  b = c;
}

int main()
{
  cout << "2 nombres:";
  int a, b;
  cin >> a >> b;
  if ( a < b )
    echange( a, b );
  cout << a << " > " << b << endl;
}
```

```
2 nombres: 5
3
5 > 3
```

Une routine d'échange de deux variables

```
void echange( int &a, int &b )
{
  int c = a;
  a = b;
  b = c;
}

int main()
{
  cout << "2 nombres:";
  int a, b;
  cin >> a >> b;
  if ( a < b )
    echange( a, b );
  cout << a << " > " << b << endl;
}
```

```
2 nombres: 2
6
6 > 2
```

Une routine d'échange de deux variables

```
void echange( int a, int b )
{
  int c = a;
  a = b;
  b = c;
}

int main()
{
  cout << "2 nombres:";
  int a, b;
  cin >> a >> b;
  if ( a < b )
    echange( a, b );
  cout << a << " > " << b << endl;
}
```

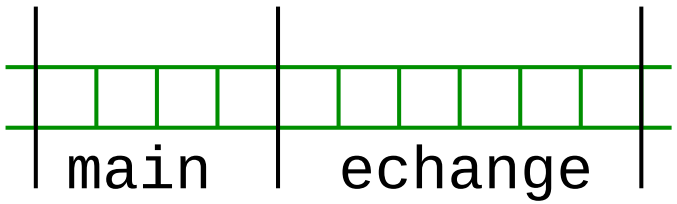
```
2 nombres: 2
6
2 > 6
```

Une routine d'échange de deux variables

```
void echange( int a, int b )
{
  int c = a;
  a = b;
  b = c;
}

int main()
{
  cout << "2 nombres:";
  int a, b;
  cin >> a >> b;
  if ( a < b )
    echange( a, b );
  cout << a << " > " << b << endl;
}
```

```
2 nombres:
```

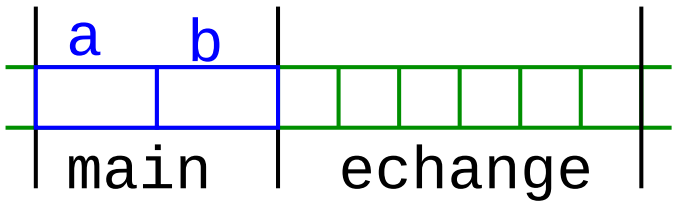


Une routine d'échange de deux variables

```
void echange( int a, int b )
{
  int c = a;
  a = b;
  b = c;
}

int main()
{
  cout << "2 nombres:";
  int a, b;
  cin >> a >> b;
  if ( a < b )
    echange( a, b );
  cout << a << " > " << b << endl;
}
```

```
2 nombres:
```

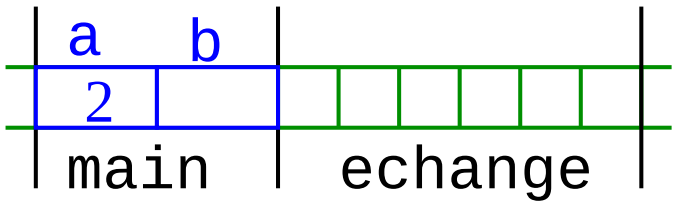


Une routine d'échange de deux variables

```
void echange( int a, int b )
{
  int c = a;
  a = b;
  b = c;
}

int main()
{
  cout << "2 nombres:";
  int a, b;
  cin >> a >> b;
  if ( a < b )
    echange( a, b );
  cout << a << " > " << b << endl;
}
```

```
2 nombres: 2
```

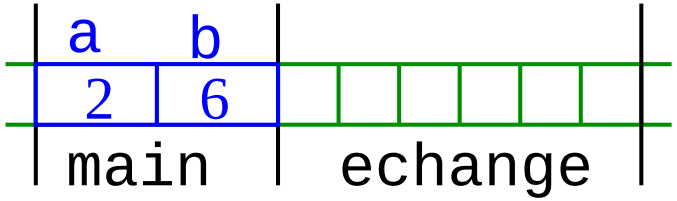


Une routine d'échange de deux variables

```
void echange( int a, int b )
{
  int c = a;
  a = b;
  b = c;
}

int main()
{
  cout << "2 nombres:";
  int a, b;
  cin >> a >> b;
  if ( a < b )
    echange( a, b );
  cout << a << " > " << b << endl;
}
```

```
2 nombres: 2
6
```

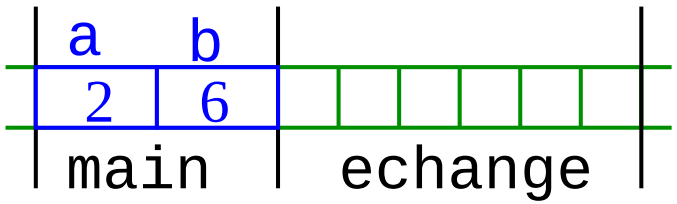


Une routine d'échange de deux variables

```
void echange( int a, int b )
{
  int c = a;
  a = b;
  b = c;
}

int main()
{
  cout << "2 nombres:";
  int a, b;
  cin >> a >> b;
  if ( a < b )
    echange( a, b );
  cout << a << " > " << b << endl;
}
```

```
2 nombres: 2
6
```

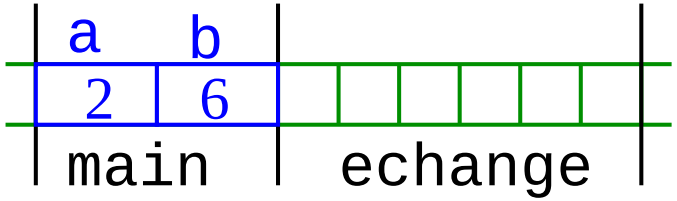


Une routine d'échange de deux variables

```
void echange( int a, int b )
{
  int c = a;
  a = b;
  b = c;
}

int main()
{
  cout << "2 nombres:";
  int a, b;
  cin >> a >> b;
  if ( a < b )
    echange( a, b );
  cout << a << " > " << b << endl;
}
```

```
2 nombres: 2
6
```

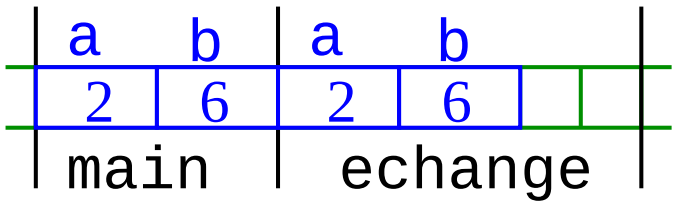


Une routine d'échange de deux variables

```
void echange( int a, int b )
{
  int c = a;
  a = b;
  b = c;
}

int main()
{
  cout << "2 nombres:";
  int a, b;
  cin >> a >> b;
  if ( a < b )
    echange( a, b );
  cout << a << " > " << b << endl;
}
```

```
2 nombres: 2
6
```

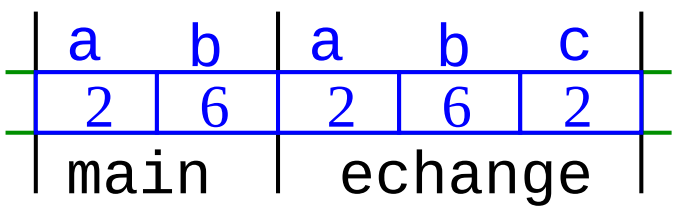


Une routine d'échange de deux variables

```
void echange( int a, int b )
{
  int c = a;
  a = b;
  b = c;
}

int main()
{
  cout << "2 nombres:";
  int a, b;
  cin >> a >> b;
  if ( a < b )
    echange( a, b );
  cout << a << " > " << b << endl;
}
```

```
2 nombres: 2
6
```

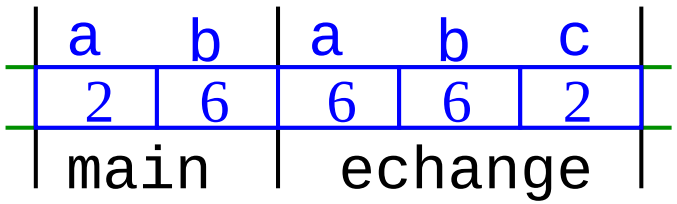


Une routine d'échange de deux variables

```
void echange( int a, int b )
{
  int c = a;
  a = b;
  b = c;
}

int main()
{
  cout << "2 nombres:";
  int a, b;
  cin >> a >> b;
  if ( a < b )
    echange( a, b );
  cout << a << " > " << b << endl;
}
```

```
2 nombres: 2
6
```

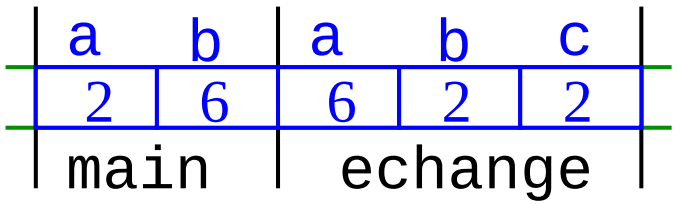


Une routine d'échange de deux variables

```
void echange( int a, int b )
{
  int c = a;
  a = b;
  b = c;
}

int main()
{
  cout << "2 nombres:";
  int a, b;
  cin >> a >> b;
  if ( a < b )
    echange( a, b );
  cout << a << " > " << b << endl;
}
```

```
2 nombres: 2
6
```

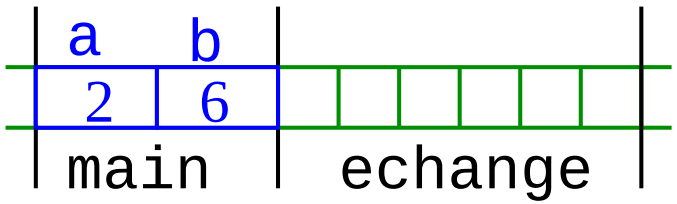


Une routine d'échange de deux variables

```
void echange( int a, int b )
{
  int c = a;
  a = b;
  b = c;
}

int main()
{
  cout << "2 nombres:";
  int a, b;
  cin >> a >> b;
  if ( a < b )
    echange( a, b );
  cout << a << " > " << b << endl;
}
```

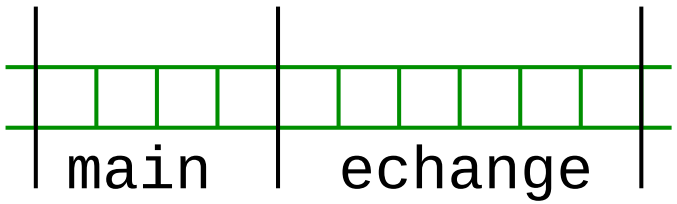
```
2 nombres: 2
6
2 > 6
```



Une routine d'échange de deux variables

```
void echange( int &a, int &b )
{
  int c = a;
  a = b;
  b = c;
}

int main()
{
  cout << "2 nombres:";
  int a, b;
  cin >> a >> b;
  if ( a < b )
    echange( a, b );
  cout << a << " > " << b << endl;
}
```

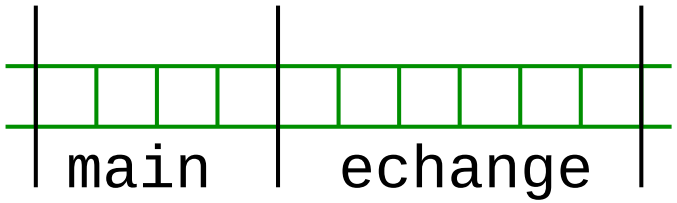


Une routine d'échange de deux variables

```
void echange( int &a, int &b )
{
  int c = a;
  a = b;
  b = c;
}

int main()
{
  cout << "2 nombres:";
  int a, b;
  cin >> a >> b;
  if ( a < b )
    echange( a, b );
  cout << a << " > " << b << endl;
}
```

```
2 nombres:
```

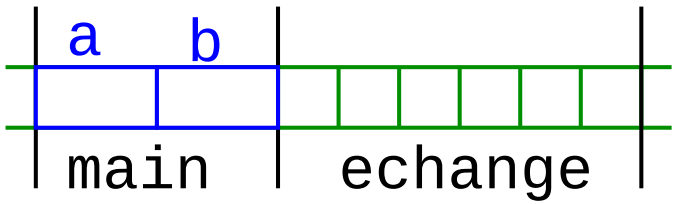


Une routine d'échange de deux variables

```
void echange( int &a, int &b )
{
  int c = a;
  a = b;
  b = c;
}

int main()
{
  cout << "2 nombres:";
  int a, b;
  cin >> a >> b;
  if ( a < b )
    echange( a, b );
  cout << a << " > " << b << endl;
}
```

```
2 nombres:
```

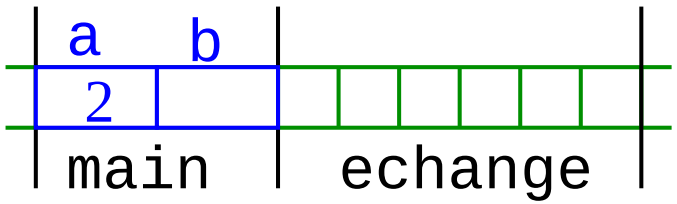


Une routine d'échange de deux variables

```
void echange( int &a, int &b )
{
  int c = a;
  a = b;
  b = c;
}

int main()
{
  cout << "2 nombres:";
  int a, b;
  cin >> a >> b;
  if ( a < b )
    echange( a, b );
  cout << a << " > " << b << endl;
}
```

```
2 nombres: 2
```

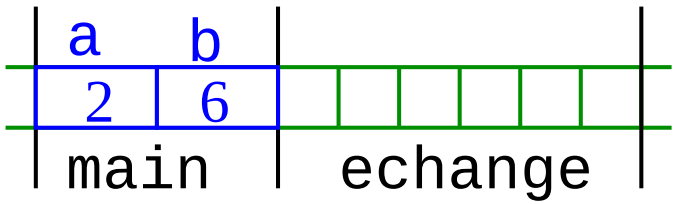


Une routine d'échange de deux variables

```
void echange( int &a, int &b )
{
  int c = a;
  a = b;
  b = c;
}

int main()
{
  cout << "2 nombres:";
  int a, b;
  cin >> a >> b;
  if ( a < b )
    echange( a, b );
  cout << a << " > " << b << endl;
}
```

```
2 nombres: 2
6
```

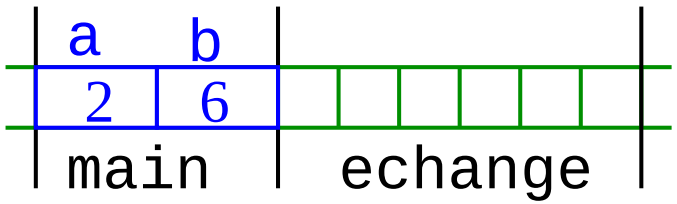


Une routine d'échange de deux variables

```
void echange( int &a, int &b )
{
  int c = a;
  a = b;
  b = c;
}

int main()
{
  cout << "2 nombres:";
  int a, b;
  cin >> a >> b;
  if ( a < b )
    echange( a, b );
  cout << a << " > " << b << endl;
}
```

```
2 nombres: 2
6
```

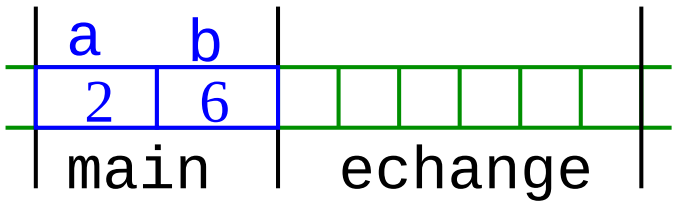


Une routine d'échange de deux variables

```
void echange( int &a, int &b )
{
  int c = a;
  a = b;
  b = c;
}

int main()
{
  cout << "2 nombres:";
  int a, b;
  cin >> a >> b;
  if ( a < b )
    echange( a, b );
  cout << a << " > " << b << endl;
}
```

```
2 nombres: 2
6
```

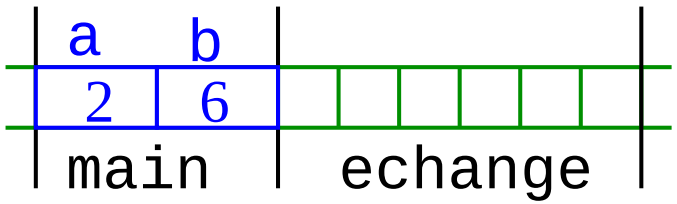


Une routine d'échange de deux variables

```
void echange( int &a, int &b )
{
  int c = a;
  a = b;
  b = c;
}

int main()
{
  cout << "2 nombres:";
  int a, b;
  cin >> a >> b;
  if ( a < b )
    echange( a, b );
  cout << a << " > " << b << endl;
}
```

```
2 nombres: 2
6
```

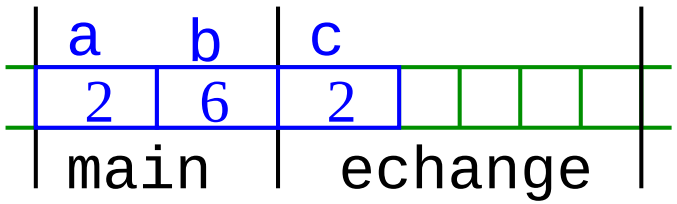


Une routine d'échange de deux variables

```
void echange( int &a, int &b )
{
  int c = a;
  a = b;
  b = c;
}

int main()
{
  cout << "2 nombres:";
  int a, b;
  cin >> a >> b;
  if ( a < b )
    echange( a, b );
  cout << a << " > " << b << endl;
}
```

```
2 nombres: 2
6
```

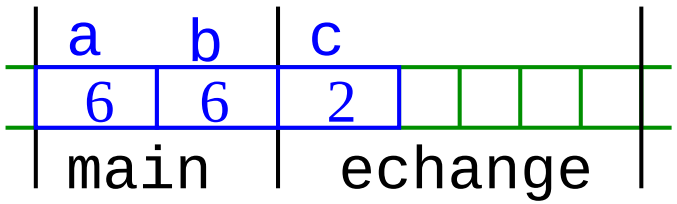


Une routine d'échange de deux variables

```
void echange( int &a, int &b )
{
  int c = a;
  a = b;
  b = c;
}

int main()
{
  cout << "2 nombres:";
  int a, b;
  cin >> a >> b;
  if ( a < b )
    echange( a, b );
  cout << a << " > " << b << endl;
}
```

```
2 nombres: 2
6
```

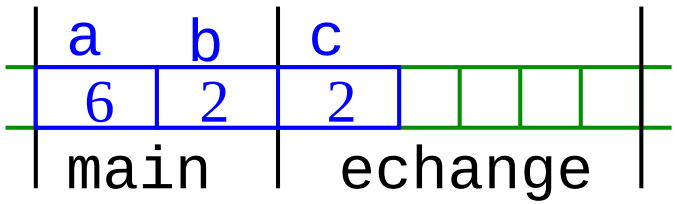


Une routine d'échange de deux variables

```
void echange( int &a, int &b )
{
  int c = a;
  a = b;
  b = c;
}

int main()
{
  cout << "2 nombres:";
  int a, b;
  cin >> a >> b;
  if ( a < b )
    echange( a, b );
  cout << a << " > " << b << endl;
}
```

```
2 nombres: 2
6
```

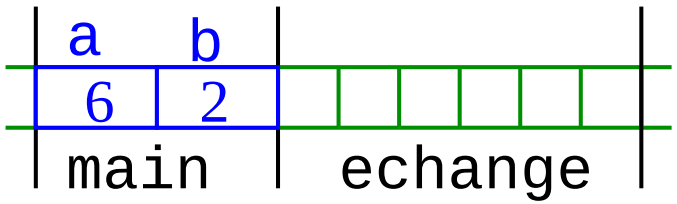


Une routine d'échange de deux variables

```
void echange( int &a, int &b )
{
  int c = a;
  a = b;
  b = c;
}

int main()
{
  cout << "2 nombres:";
  int a, b;
  cin >> a >> b;
  if ( a < b )
    echange( a, b );
  cout << a << " > " << b << endl;
}
```

```
2 nombres: 2
6
6 > 2
```



Une routine d'échange de deux variables

```
void echange( int &a, int &b )
{
  int c = a;
  a = b;
  b = c;
}

int main()
{
  cout << "2 nombres:";
  int a, b;
  cin >> a >> b;
  if ( a < b )
    echange( a + b, b );
  cout << a << " > " << b << endl;
}
```

⟶ erreur:

```
no matching function for call to 'echange(int, int&)'
```

6.4 Quelques fonctions standards

`double fabs( double x )` $\longrightarrow$ valeur absolu d'un flottant
`double sqrt( double x )` $\longrightarrow$ racine $\sqrt{x}$
`double exp( double x )` $\longrightarrow$ fonction exponentielle e^x
`double log( double x )` $\longrightarrow$ logarithme naturel $\ln(x)$
`double sin( double x )` $\longrightarrow$ sinus (en radians)
`double cos( double x )` $\longrightarrow$ cosinus (en radians)
`double tan( double x )` $\longrightarrow$ tangente (en radians)
`double asin( double x )` $\longrightarrow$ arcsin (en radians)
`double acos( double x )` $\longrightarrow$ arccos (en radians)
`double atan( double x )` $\longrightarrow$ arctan (en radians)

`double atan2( double y, double x )` $\longrightarrow \varphi \in (-\pi, \pi]$
pour lequel $\sin(\varphi) = y/\sqrt{x^2+y^2}$ et $\cos(\varphi) = x/\sqrt{x^2+y^2}$

$\longrightarrow$ il faut inclure `#include <cmath>`
(ou `#include <math.h>` dans C)
pour utiliser ces fonctions mathématiques

6.4 Quelques fonctions standards

`double fabs( double x )` $\longrightarrow$ valeur absolu d'un flottant
`double sqrt( double x )` $\longrightarrow$ racine $\sqrt{x}$
`double exp( double x )` $\longrightarrow$ fonction exponentielle e^x
`double log( double x )` $\longrightarrow$ logarithme naturel $\ln(x)$
`double sin( double x )` $\longrightarrow$ sinus (en radians)
`double cos( double x )` $\longrightarrow$ cosinus (en radians)
`double tan( double x )` $\longrightarrow$ tangente (en radians)
`double asin( double x )` $\longrightarrow$ arcsin (en radians)
`double acos( double x )` $\longrightarrow$ arccos (en radians)
`double atan( double x )` $\longrightarrow$ arctan (en radians)

`double atan2( double y, double x )` $\longrightarrow \varphi \in (-\pi, \pi]$
pour lequel $\sin(\varphi) = y/\sqrt{x^2+y^2}$ et $\cos(\varphi) = x/\sqrt{x^2+y^2}$

`double pow( double x, double y )` $\longrightarrow x^y$

Des nombres aléatoires

$\longrightarrow$ inclure `#include <cstdlib>` (ou `<stdlib.h>` pour C)

`int rand()` $\longrightarrow$ nombre aléatoire entier entre 0 et $2^{31} - 1$

Des nombres aléatoires

$\longrightarrow$ inclure `#include <cstdlib>` (ou `<stdlib.h>` pour C)

`int rand()` $\longrightarrow$ nombre aléatoire entier entre 0 et $2^{31} - 1$

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

int main()
{
  cout << rand() << endl;
  cout << rand() << endl;
}
```

Exécution:

```
1804289383
846930886
```

(toujours)

Des nombres aléatoires

$\longrightarrow$ inclure `#include <cstdlib>` (ou `<stdlib.h>` pour C)

`int rand()` $\longrightarrow$ nombre aléatoire entier entre 0 et $2^{31} - 1$

```
void srand( unsigned int seed )
```
$\longrightarrow$ *seed* du générateur des nombres aléatoires

combiner avec la fonction `time` dans `<ctime>` (ou `<time.h>`)
(ça rend le temps en secondes découlé depuis le 1/1/1970)

Des nombres aléatoires

$\longrightarrow$ inclure `#include <cstdlib>` (ou `<stdlib.h>` pour C)

`int rand()` $\longrightarrow$ nombre aléatoire entier entre 0 et $2^{31} - 1$

`void srand( unsigned int seed )`
$\longrightarrow$ *seed* du générateur des nombres aléatoires

combiner avec la fonction `time` dans `<ctime>` (ou `<time.h>`)
(ça rend le temps en secondes découlé depuis le 1/1/1970)

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

int main()
{
  srand( time( 0 ) );
  cout << rand() << endl;
  cout << rand() << endl;
}
```

Exécution:

```
422776193
244047141
```

Des nombres aléatoires

$\longrightarrow$ inclure `#include <cstdlib>` (ou `<stdlib.h>` pour C)

`int rand()` $\longrightarrow$ nombre aléatoire entier entre 0 et $2^{31} - 1$

`void srand( unsigned int seed )`
$\longrightarrow$ *seed* du générateur des nombres aléatoires

combiner avec la fonction `time` dans `<ctime>` (ou `<time.h>`)
(ça rend le temps en secondes découlé depuis le 1/1/1970)

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

int main()
{
  srand( time( 0 ) );
  cout << rand() << endl;
  cout << rand() << endl;
}
```

Exécution:

```
1282589832
1036426062
```

Des nombres aléatoires

$\longrightarrow$ inclure `#include <cstdlib>` (ou `<stdlib.h>` pour C)

`int rand()` $\longrightarrow$ nombre aléatoire entier entre 0 et $2^{31} - 1$

`void srand( unsigned int seed )`
$\longrightarrow$ *seed* du générateur des nombres aléatoires

combiner avec la fonction `time` dans `<ctime>` (ou `<time.h>`)
(ça rend le temps en secondes découlé depuis le 1/1/1970)

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

int main()
{
  srand( time( 0 ) );
  cout << rand() << endl;
  cout << rand() << endl;
}
```

Exécution:

```
1318789015
568117324
```

Des nombres aléatoires

$\longrightarrow$ inclure `#include <cstdlib>` (ou `<stdlib.h>` pour C)

`int rand()` $\longrightarrow$ nombre aléatoire entier entre 0 et $2^{31} - 1$

```
void srand( unsigned int seed )
```
$\longrightarrow$ *seed* du générateur des nombres aléatoires

combiner avec la fonction `time` dans `<ctime>` (ou `<time.h>`)
(ça rend le temps en secondes découlé depuis le 1/1/1970)

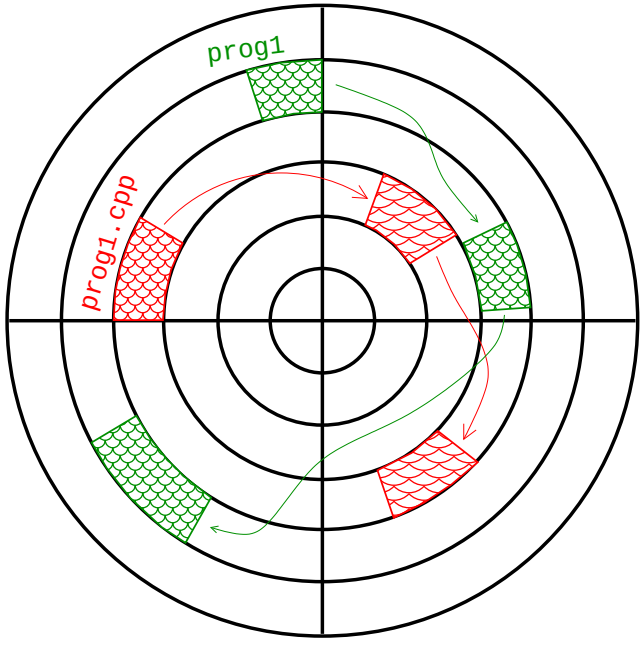
$\longrightarrow$ très utile pour des calculs stochastiques
(et pour la programmation des jeux . . .)

6.5 Lecture et écriture des fichiers

Un fichier est une collection de données informatiques, représentées en octets, qui se trouve sur le disque dur (ou sur une clé USB / un CD . . .):

`/home/peter/cours/prog/prog1.cpp`

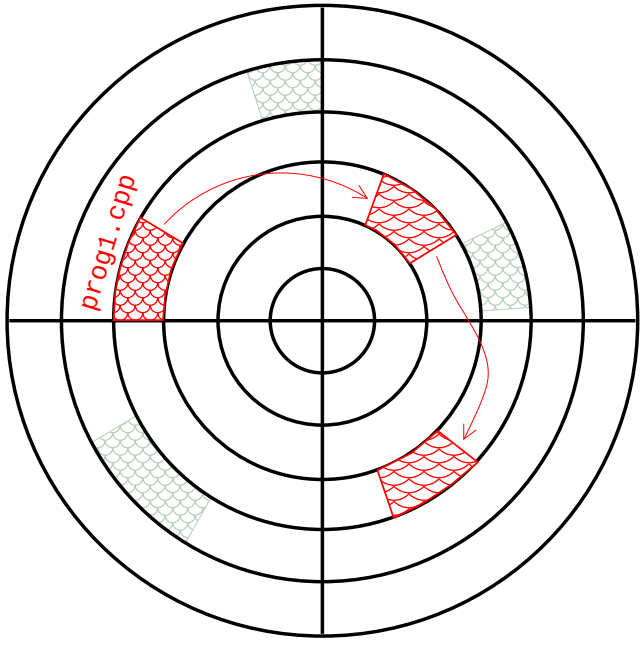
`/home/peter/cours/prog/prog1`



6.5 Lecture et écriture des fichiers

Un fichier est une collection de données informatiques, représentées en octets, qui se trouve sur le disque dur (ou sur une clé USB / un CD . . .):

```
/home/peter/cours/prog/prog1.cpp
/home/peter/cours/prog/prog1
rm prog1
```



$\longrightarrow$ enlever des fichiers fait toujours laisser des traces sur le disque (que les professionnels pourront accéder . . .)

Les fichiers ASCII

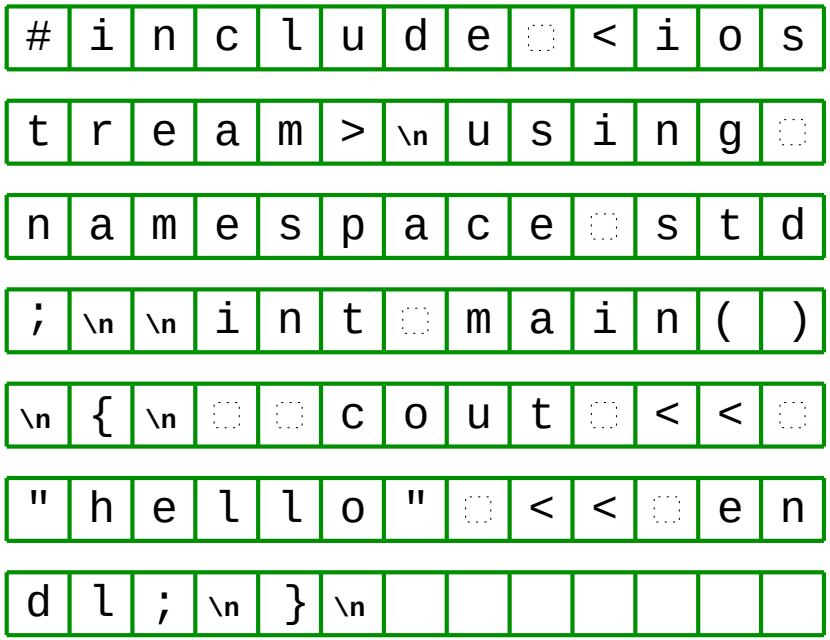
Un fichier ASCII est un fichier qui contient de l'information lexicale, c.-à-d., des caractères encodés selon le code ASCII.

Le fichier `prog1.cpp` ...

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

int main()
{
  cout << "hello" << endl;
}
```

... enregistré sur le disque dur :



Les fichiers ASCII

Un fichier ASCII est un fichier qui contient de l'information lexicale, c.-à-d., des caractères encodés selon le code ASCII.

Le fichier `prog1.cpp` ...

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

int main()
{
  cout << "hello" << endl;
}
```

... enregistré sur le disque dur :

35	105	110	99	108	117	100	101	32	60	105	111	115
116	114	101	97	109	62	10	117	115	105	110	103	32
110	97	109	101	115	112	97	99	101	32	115	116	100
59	10	10	105	110	116	32	109	97	105	110	40	41
10	123	10	32	32	99	111	117	116	32	60	60	32
34	104	101	108	108	111	34	32	60	60	32	101	110
100	108	59	10	125	10							

$\longrightarrow$ un fichier ASCII contient des octets entre 32 et 126 (et éventuellement quelques "octets de contrôle", comme 10 $=$ \n)

Input/output avec des fichiers ASCII

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

int puissance( int k, int n )
{
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  return kn;
}

int main()
{
  int n, Nmax;
  cout << "exposant:";
  cin >> n;
  cout << "nombre maximal:";
  cin >> Nmax;
  ofstream out( "fichier.dat" );
  for ( int k = 1; k <= Nmax; k++ )
  for ( int l = 1; l <= Nmax; l++ )
  for ( int m = 1; m <= Nmax; m++ )
    if ( puissance( k, n ) +
         puissance( l, n ) ==
         puissance( m, n ) )
      out << k << "^" << n << " + "
          << l << "^" << n << " = "
          << m << "^" << n << endl;
}
```

Exécution:

```
exposant: 2
nombre maximal: 20
```

$\longrightarrow$ création d'un fichier `fichier.dat` dans le répertoire actuel

Input/output avec des fichiers ASCII

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

int puissance( int k, int n )
{
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  return kn;
}

int main()
{
  int n, Nmax;
  cout << "exposant:";
  cin >> n;
  cout << "nombre maximal:";
  cin >> Nmax;
  ofstream out( "fichier.dat" );
  for ( int k = 1; k <= Nmax; k++ )
  for ( int l = 1; l <= Nmax; l++ )
  for ( int m = 1; m <= Nmax; m++ )
    if ( puissance( k, n ) +
         puissance( l, n ) ==
         puissance( m, n ) )
      out << k << "^" << n << " + "
          << l << "^" << n << " = "
          << m << "^" << n << endl;
}
```

Le contenu de `fichier.dat`:

```
3^2 + 4^2 = 5^2
4^2 + 3^2 = 5^2
5^2 + 12^2 = 13^2
6^2 + 8^2 = 10^2
8^2 + 6^2 = 10^2
8^2 + 15^2 = 17^2
9^2 + 12^2 = 15^2
12^2 + 5^2 = 13^2
12^2 + 9^2 = 15^2
12^2 + 16^2 = 20^2
15^2 + 8^2 = 17^2
16^2 + 12^2 = 20^2
```

Input/output avec des fichiers ASCII

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

int puissance( int k, int n )
{
  int kn = 1;
  for ( int i = 1; i <= n; i++ )
    kn *= k;
  return kn;
}

int main()
{
  int n, Nmax;
  ifstream inp( "param.inp");
  inp >> n;
  inp >> Nmax;
  ofstream out( "fichier.dat" );
  for ( int k = 1; k <= Nmax; k++ )
  for ( int l = 1; l <= Nmax; l++ )
  for ( int m = 1; m <= Nmax; m++ )
    if ( puissance( k, n ) +
         puissance( l, n ) ==
         puissance( m, n ) )
      out << k << "^" << n << " + "
          << l << "^" << n << " = "
          << m << "^" << n << endl;
}
```

$\longrightarrow$ lecture des paramètres `n` et `Nmax` depuis le fichier `param.inp`

ceci pourrait contenir

```
2
20
```

$\longrightarrow$ sortie dans `fichier.dat`

Input/output avec des fichiers ASCII

disponible avec `#include <fstream>`

`ifstream` ⟨nom⟩`(` ⟨string⟩ `);`

⟶ définition d'une "variable" ⟨nom⟩ du type `ifstream`
pour la lecture depuis le fichier ⟨string⟩ (entre guillemets)

`ofstream` ⟨nom⟩`(` ⟨string⟩ `);`

⟶ définition d'une "variable" ⟨nom⟩ du type `ofstream`
pour l'écriture dans le fichier ⟨string⟩

⟶ utilisation comme `cin` et `cout`

Input/output avec des fichiers binaires

Fichier “binaire” $=$ fichier qui contient de l’information binaire générique, sans la restriction aux caractères ASCII.

$\longrightarrow$ fichiers d’exécution de programme (`prog1`)

$\longrightarrow$ fichiers Word, Excel, PowerPoint, pdf, . . .

Input/output avec des fichiers binaires

Fichier “binaire” = fichier qui contient de l’information binaire générique, sans la restriction aux caractères ASCII.

Lecture d’un fichier binaire :

```
ifstream inp( "file.dat" );
char c;
inp.get( c );
```

$\longrightarrow$ lecture d’un caractère (contenu entre 0 et 255) depuis le fichier `file.dat`

Input/output avec des fichiers binaires

Fichier “binaire” = fichier qui contient de l’information binaire générique, sans la restriction aux caractères ASCII.

Ecriture dans un fichier binaire :

```
ofstream out( "file.dat" );
char c = 'A';
out.put( c );
out.put( c + 100 );
out.put( 27 );
out.put( 'e' );
```

$\longrightarrow$ écriture des caractères 65 (’A’), 165, 27, 99 (’e’) dans le fichier `file.dat`

Input/output avec des fichiers binaires

Affichage du contenu d'un fichier binaire :

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

int main()
{
  ifstream inp( "prog1.cpp" );
  char c;
  while ( inp.get( c ) )
    cout << int( c ) << " ";
}
```

Le contenu de `prog1.cpp` :

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

int main()
{
  cout << "hello" << endl;
}
```

Input/output avec des fichiers binaires

Affichage du contenu d'un fichier binaire :

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

int main()
{
  ifstream inp( "prog1.cpp" );
  char c;
  while ( inp.get( c ) )
    cout << int( c ) << " ";
}
```

Le contenu de `prog1.cpp` :

35	105	110	99	108	117	100	101	32	60	105	111	115
116	114	101	97	109	62	10	117	115	105	110	103	32
110	97	109	101	115	112	97	99	101	32	115	116	100
59	10	10	105	110	116	32	109	97	105	110	40	41
10	123	10	32	32	99	111	117	116	32	60	60	32
34	104	101	108	108	111	34	32	60	60	32	101	110
100	108	59	10	125	10							

Exécution:

```
35 105 110 99 108 117 100 101 32 60 105 111 115 116 114
101 97 109 62 10 117 115 105 110 103 32 110 97 109 101
115 112 97 99 101 32 115 116 100 59 10 10 105 110 116 32
109 97 105 110 40 41 10 123 10 32 32 99 111 117 116 32
60 60 32 34 104 101 108 108 111 34 32 60 60 32 101 110
100 108 59 10 125 10
```

Input/output avec des fichiers binaires

Affichage du contenu d'un fichier binaire :

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

int main()
{
  ifstream inp( "prog1" );
  char c;
  while ( inp.get( c ) )
    cout << int( c ) << " ";
}
```

Le contenu de `prog1.cpp` :

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

int main()
{
  cout << "hello" << endl;
}
```

```
c++ prog1.cpp -o prog1
```

Exécution:

```
127 69 76 70 2 1 1 0 0 0 0 0 0 0 0 0 3 0 62 0 1 0 0 0
-80 7 0 0 0 0 0 0 64 0 0 0 0 0 0 0 -104 27 0 0 0 0 0 0 0
0 0 0 64 0 56 0 9 0 64 0 29 0 28 0 6 0 0 0 4 0 0 0 64 0
0 0 0 0 0 0 64 0 0 0 0 0 0 0 64 0 0 0 0 0 0 0 -8 1 0 0 0
0 0 0 -8 1 0 0 0 0 0 0 8 0 0 0 0 0 0 0 3 0 0 0 4 0 0 0
56 2 0 0 0 0 0 0 56 2 0 0 0 0 0 0 56 2 0 0 0 0 0 0 28 0
0 0 0 0 0 0 28 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 5 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 112
11 0 0 0 0 0 0 112 11 0 0 0 0 0 0 0 32 0 0 0 0 0 1 0 0
0 6 0 0 0 120 13 0 0 0 0 0 0 120 13 32 0 0 0 0 0 120 13
32 0 0 0 0 0 -104 2 0 0 0 0 0 0 -64 3 0 0 0 0 0 0 0 0 32
0 0 0 0 0 2 0 0 0 6 0 0 0 -112 13 0 0 0 0 0 0 -112 13 32
0 0 0 0 0 -112 13 32 0 0 0 0 0 0 2 0 0 0 0 0 0 0 2 0 0 0
0 0 0 8 0 0 0 0 0 0 0 4 0 0 0 4 0 0 0 84 2 0 0 0 0 0 0
```