Fonctions et portées

Exemples de code

Que fait ce bout de code?

```
int a=4, b=7; // entrée
int x=-1; // sortie
if (a >= b) {
  x = a;
} else {
  x = b;
}
```

Exemples de code

Il calcule le maximum!

```
int a = 4, b = 7; // entrée
int x = max(a, b); // sortie
```

Exemples de code

Que fait ce bout de code?

```
int a = -3; // ou 4
int x = 0; // sortie
if (a < 0) {
  x = -a;
} else {
  x = a;
}
```

Exemples de code

Il calcule la valeur absolue!

```
int a = -3; // ou 4
int x = abs(a);
```

Exemples de code

Que fait ce bout de code?

```
int a = 4, b = 7; // entrée et sortie
int tmp = a;
a = b;
b = tmp;
```

Exemples de code

Il échange les valeurs des deux variables!

```
int a = 4, b = 7; // entrée et sortie
swap(a, b);
```

Fonctions

Définition:

```
int carre(int x) {
  return x*x;
}
```

Usage:

```
int x = carre(3);
int y = carre(x);
if (carre(x) == carre(y)) { ... }
```

Fonctions

```
int carre(int x) {...}
 |    |   -----  -----
 |   nom    |       |
 |          |    corps
type de la  |
fonction    |
         paramètre
```

Un paramètre a un type et un nom. Dans le corps de la fonction, un paramètre s'utilise comme une variable.

Une fonction renvoie une valeur du type déclaré en utilisant une instruction "return":

```
return x*x;
```

Fonctions

Avec plusieurs paramètres et plusieurs instructions “return”:

```
int max_de_3(int a, int b, int c) {
  if (a > b) {
    if (a > c)
      return a;
    else
      return c;
  } else {
    if (b > c)
      return b;
    else
      return c;
  }
}
```

Fonctions

Tu peux aussi utiliser des fonctions à l'intérieur de ta fonction!

```
int max_de_3(int a, int b, int c) {
  return max(max(a, b), c);
}
```

Fonctions

Parfois, il peut être utile de créer une fonction qui ne retourne pas de valeur. On les déclare en utilisant le type spécial void:

```
void function(...) {
    ...
    return;
}
```

Portées

Une portée est l'intervalle entre la définition d'une variable et le "}" y correspondant.

```
signed main() {
  int a; // la portée de a commence
  int b; // la portée de b commence
  if (...) {
    int c = a+b; // la portée de c commence
                 // et on peut utiliser a et b ici!
  } // <-- la portée de c se termine
  int d = c; // erreur: c n'est plus à notre portée
  int e = a; // ok

  a = e+b; // ok
} // <- les portées de a, b et e se terminent
```

Portées

Tu ne peux pas définir la même variable à double.

```
signed main() {
  int a = 2;
  int a = 3; // erreur
}
```

Portées

Tu peux les définir dans des blocs de code séparés.

```
signed main() {
  if (...) {
    int a = 3;
  } else {
    int a = 1; // ok
  }
}
```

Portées

Attention: tu peux “cacher” des variables !

Très souvent, c’est une source d’erreur.

```
signed main() {
  int a = 1;
  if (...) {
    print(a); // imprime 1
    int a = 2;
    print(a); // imprime 2
  }
  print(a); // imprime 1
}
```

Portées

Les fonctions aussi ont leur propre portée séparée.

```
int f(int x) {
  return a + x; // erreur: a n'est pas défini
}

signed main() {
  int a;
  print(f(3));
}
```

Portées

Les fonctions aussi ont des noms.

```
signed main() {
  int a;
  print(f(3));  // erreur, f n'est pas déclarée
}

int f(int x) {
  return 2*x;
}
```

Portées

Les boucles for sont bizarres:

```
signed main() {
  for (int i = 0; i < 4; ++i) { // déf. de i
    // possible d'utiliser i
  }
  // plus possible d'utiliser i
}
```

Portées

Ligne directrice: garde la portée de tes variables aussi courte que possible.

Mauvais:

```
int n;
int reponse = -1;
n = read_int();
reponse = n*n;
print(reponse);
```

Portées

Ligne directrice: garde la portée de tes variables aussi courte que possible.

Bon:

```
int n = read_int();
int reponse = n*n;
print(reponse);
```

Portées

Ligne directrice: garde la portée de tes variables aussi courte que possible.

Mieux:

```
int n = read_int();
print(n*n);
```

Portées

Ligne directrice: garde la portée de tes variables aussi courte que possible.

Mauvais:

```
int somme = 0;
int x = 0;
for (int i = 0; i < n; ++i) {
  x = read_int();
  somme += x;
}
print(somme);
```

Portées

Ligne directrice: garde la portée de tes variables aussi courte que possible.

Bon:

```
int somme = 0;
for (int i = 0; i < n; ++i) {
  int x = read_int();
  somme += x;
}
print(somme);
```

Fonctions utiles

La bibliothèque standard fournit de nombreuses fonctions utiles.
Utilise-les!

```
max(a, b); // maximum de deux nombres
min(a, b); // minimum de deux nombres
max({a, b, c, d, e}); // max de n'importe quelle quantité
min({a, b, c, d, e}); // min de n'importe quelle quantité
gcd(a, b); // plus grand diviseur commun de a et b
abs(x); // valeur absolue de x

swap(a, b); // échange les valeurs de a et b
            // le fait en utilisant des références
            // -- attends le cours sur le C++ avancé
```