Memento C/C++

Ce feuillet donne une idée des commandes et déclarations les plus importantes (et non toutes les commandes et déclarations) en C et en C++.

Les éléments entre < > appartiennent à la grammaire définissant le langage, et sont (généralement) définis par ::= suivi d'une liste de possibilités séparées par |. Les éléments **en gras** appartiennent au langage lui-même. Un élément entre [ ] est optionnel, $(e)^+$ désigne l'élément e répété une ou plusieurs fois, alors que $(e)^*$ est équivalent à $\left[(e)^+\right]$ (e est répété zéro, une ou plusieurs fois).

< programme > ::= (< Inclusion de fichier >)*

(< Définition de macros >)*
(< Définition de type > ;)*
(< Déclaration de variable > ;)*
(< Déclaration de fonction >)*

< Déclaration de la fonction principale >

Des commentaires peuvent être ajoutés où que ce soit.

< Commentaire > ::= < Commentaire C > | < Commentaire C++ >

< Commentaire C > ::= `/*` < Texte > `*/`

< Commentaire C++ > ::= `//` < Texte > < Fin de ligne >

Commandes de pré-compilation

< Inclusion de fichier > ::= `#include` < Référence de fichier >

< Référence de fichier > ::= `"` < Nom de fichier > `"` | `<` < Nom de fichier > `>`

< Définition de macros > ::= `#define` < Identificateur > < Expression >

Identificateurs et expressions

Identificateurs

< Identificateur > ::= < Caractère alphabétique > (< Caractère alpha-numérique >)*

< Caractère alphabétique > ::= < Caractère alphabétique minuscule > |
< Caractère alphabétique majuscule > | _

< Caractère alphabétique minuscule > ::= `a` | ... | `z`

< Caractère alphabétique majuscule > ::= `A` | ... | `Z`

< Caractère alpha-numérique > ::= < Caractère alphabétique > |
< Caractère numérique >

< Caractère numérique > ::= `0` | ... | `9`

Expressions

< Expression > ::= < Expression primaire > | < Opérateur unaire > < Expression > |
< Expression > < Opérateur binaire > < Expression > |
< Lvalue > < Opérateur d'affectation > < Expression > |
< Opérateur lvalue > < Lvalue > |
< Lvalue > < Opérateur lvalue > |
< Expression > `?` < Expression > `:` < Expression > |
`(` < Typage > `)` < Expression >

< Expression primaire > ::= < Lvalue > | < Constante > | `(` < Expression > `)` |
< Expression primaire > `(` < Expression > `)`

< Lvalue > ::= < Identificateur > | < Expression primaire > `[` < Expression > `]` |
< Lvalue > `.` < Identificateur > |
< Expression primaire > `->` < Identificateur > |
`*` < Expression > | `(` < Lvalue > `)`

< Opérateur unaire > ::= `*` | `&` | `-` | `!` | `~`

< Opérateur binaire > ::= `+` | `-` | `*` | `/` | `%` | `<<` | `>>` | `==` | `!=` | `<` | `>` | `<=` | `>=` |
`&` | `^` | | | `&&` | || | `,`

< Opérateur d'affectation > ::= `=` | `+=` | `-=` | `*=` | `/=` | `%=` | `<<=` | `>>=` | `&=` | `^=` | `|=`

< Opérateur lvalue > ::= `++` | `--`

< Typage > ::= < Type > [< Abstraction déclarable >]

< Abstraction déclarable > ::=
`(` < Abstraction déclarable > `)` | `*` < Abstraction déclarable > |
< Abstraction déclarable > `()` | < Abstraction déclarable > `[` [< Constante >] `]`

Déclarations

Types

< Définition de type > ::=
`typedef` < Type > < Élément déclarable > $\left(\texttt{,} < \text{Élément déclarable} >\right)^{*}$ `;` |
< Constructeur de type > `;`

< Type > ::= < Type C > | < Type C++ >

< Type C > ::=
`char` | `short` | `int` | `long` | `unsigned` | `float` | `double` |
`enum` [{ < Identificateur > (, < Identificateur >)* }] |
< Structure ou union > < Identificateur > |
< Structure ou union > [< Identificateur >] { (< Déclaration de champ >)+ }

< Type C++ > ::= `const` < Type C > | < Identificateur >

< Structure ou union > ::= `struct` | `union`

< Déclaration de champ > ::= < Type > < Champ > (, < Champ >)* ;

< Champ > ::= < Élément déclarable > [: < Constante >] | : < Constante >

< Élément déclarable > ::= < Élément déclarable C > | < Élément déclarable C++ >

< Élément déclarable C > ::= < Identificateur > | (< Élément déclarable C >) |
`*` < Élément déclarable C > | < Élément déclarable C > `()` |
< Élément déclarable C > `[` [< Constante >] `]`

< Élément déclarable C++ > ::= `&` < Élément déclarable >

< Constructeur de type > ::=
< Constructeur de type C > | < Constructeur de type C++ >

< Constructeur de type C > ::=
< Structure ou union > < Identificateur > { (< Déclaration de champ >)+ }

< Constructeur de type C++ > ::=
`class` < Identificateur >
[: [< Déclaration d'héritage >] < Identificateur > (, < Identificateur >)*
(, < Déclaration d'héritage > < Identificateur > (, < Identificateur >)*)*]
{ [< Déclaration d'héritage >] :
(< Déclaration de champ >)*
(< Déclaration de fonction >)*
(< Déclaration d'héritage > :
(< Déclaration de champ >)*
(< Déclaration de fonction >)*)* }

< Déclaration d'héritage > ::= `public` | `protected` | `private`

Variables

< Déclaration de variable > ::=
[< Classe de mémorisation >] < Type > < Élément initialisable >
$\left(\text{, < Élément initialisable >} \right)^*$;

< Classe de mémorisation > ::= `auto` | `static` | `extern` | `register`

< Élément initialisable > ::= < Élément déclarable > [< Initialisation >]

< Initialisation > ::= `=` < Expression > | `=` { < Liste d'initialisations > [`,`] }

< Liste d'initialisations > ::= < Expression > |
< Liste d'initialisations > `,` < Liste d'initialisations > |
{ < Liste d'initialisations > }

Fonctions

< Déclaration de fonction > ::= < En-tête C++ >
< Type > < Élément déclarable > `(` [< Déclaration de variable >
(`,` < Déclaration de variable >)*] `)`
< Spécification C++ >
< Bloc d'instructions >

< En-tête C++ > ::= [`inline`] [`friend`]

< Spécification C++ > ::= [`const`]

< Déclaration de la fonction principale > ::= [`int`] `main (` [`int argc, char *arv[];`] `)`
< Bloc d'instructions >

Instructions

< Instruction > ::= `;` | < Expression > `;` | < Bloc d'instructions > |
`if (` < Expression > `)` < Instruction > [`else` < Instruction >] |
`while (` < Expression > `)` < Instruction > |
`do` < Instruction > `while (` < Expression > `);` |
`for (` [< Expression >] `;` [< Expression >] `;` [< Expression >] `)`
< Instruction > |
`switch` < Expression > {
(`case` < Constante > `:` < Instruction > [`break;`])*
[`default:` < Instruction >]
}

< Bloc d'instructions > ::= { (< Définition de type > `;`)*
(< Déclaration de variable > `;`)*
(< Déclaration de fonction > `;`)*
(< Instruction >)* } |

Références bibliographiques

[1] B. W. Kernighan and D. M. Ritchie. *Le langage C.* Manuels informatiques. Masson, Paris, 1985. Titre en anglais : The C Programming Language ; traduit par Th. Buffenoir.

[2] B. Stroustrup. *Le langage C++.* Références. CampusPress France, Paris, 1999. Titre en anglais : The C++ Programming Language ; traduit par Ch. Eberhardt.