Types de Bases

1

Généré par Doxygen 1.7.6.1

Jeudi Novembre 22 2012 09 :13 :19

Table des matières

Chapitre 1

Projet : Les Types de base.

Description de la bibliothèque :

Les types disponibles :

Il y a actuellement 4 types qui ont été implémentés :
- Entier (équivalent int)
- Flottant (équivalent float)
- Car (équivalent char)
- String (équivalent char$*$)

Que représente un type ?

Chaque type représente un pointeur sur une structure commune à tous les types (-Type). Les types se différencient dans une seconde structure interne. L'accès à la valeur stockée dans la variable ne peut se faire directement. Dans les fonctions génériques proposées, toutes les variables devront être de type Type.

Auteur

VERDIER Frédéric, LAMEIRA Yannick

Chapitre 2

Index des structures de données

2.1 Structures de données

Liste des structures de données avec une brève description :

Chapitre 3

Index des fichiers

3.1 Liste des fichiers

Liste de tous les fichiers avec une brève description :

Chapitre 4

Documentation des structures de données

4.1 Référence de la structure car

`#include <typeBase.h>`

Champs de données

– char val

4.1.1 Documentation des champs

4.1.1.1 char car : :val

La documentation de cette structure a été générée à partir du fichier suivant :
– Documents/fac/tccp2012/Sd/TypesBase/typeBase.h

4.2 Référence de la structure entier

`#include <typeBase.h>`

Champs de données

– int val

4.2.1 Documentation des champs

4.2.1.1 int entier : :val

La documentation de cette structure a été générée à partir du fichier suivant :
– Documents/fac/tccp2012/Sd/TypesBase/typeBase.h

4.3 Référence de la structure flottant

`#include <typeBase.h>`

Champs de données

– float val

4.3.1 Documentation des champs

4.3.1.1 float flottant : :val

La documentation de cette structure a été générée à partir du fichier suivant :
– Documents/fac/tccp2012/Sd/TypesBase/typeBase.h

4.4 Référence de la structure string

`#include <typeBase.h>`

Champs de données

– char * val
– int taille

4.4.1 Documentation des champs

4.4.1.1 int string : :taille

4.4.1.2 char* string : :val

La documentation de cette structure a été générée à partir du fichier suivant :
– Documents/fac/tccp2012/Sd/TypesBase/typeBase.h

4.5 Référence de la structure Type

`#include <typeBase.h>`

Champs de données

– int type
– void ∗ objet

4.5.1 Documentation des champs

4.5.1.1 void∗ Type : :objet

4.5.1.2 int Type : :type

La documentation de cette structure a été générée à partir du fichier suivant :
– Documents/fac/tccp2012/Sd/TypesBase/typeBase.h

Chapitre 5

Documentation des fichiers

5.1 Référence du fichier Documents/fac/tccp2012/Sd/Types-Base/test_typeBase.c

```
#include "typeBase.h"
```

Fonctions

– int main ()

5.1.1 Documentation des fonctions

5.1.1.1 int main ( )

5.2 Référence du fichier Documents/fac/tccp2012/Sd/Types-Base/typeBase.c

```
#include "typeBase.h"
```

Fonctions

– Entier Entier_Creer (int valeur)
  Le constructeur Entier : Ce constructeur va nous permettre de créer une variable de type Entier avec la syntaxe suivante : Entier <variable> = Entier_Creer(<valeur>).
– Flottant Flottant_Creer (float valeur)
  Le constructeur Flottant : Ce constructeur va nous permettre de créer une variable de type Flottant avec la syntaxe suivante : Flottant <variable> = Flottant_Creer(<valeur>).
– Car Car_Creer (char valeur)
  Le constructeur Car : Ce constructeur va nous permettre de créer une variable de type Car avec la syntaxe suivante : Car <variable> = Car_Creer(<valeur>).
– String String_Creer (char *valeur)

Le constructeur String : Ce constructeur va nous permettre de créer une variable de type String avec la syntaxe suivante : String <variable> = String_Creer(<valeur>).

- Entier void Detruire (Type object)
  Les destructeurs : Ces destructeur vont nous permettre de libérer la zone mémoire allouée par une variable a. On utilisera la fonction void Detruire(Type a).
- void * Clone (Type object)
  Les copies de variables : Pour copier une variable, il faut utiliser void Clone(Type a). Pour cela, on utilisera un switch pour nous permettre de choisir suivant le type de object. Exemple : Entier e = Entier_Creer(5) ; Entier e2 = Clone(e) ;.*
- int Egal (Type a, Type b)
  Les tests d'égalité : La fonction int Egal(Type a, Type b) renvoie 0 si a=b, 1 sinon. A noter que si a et b sont de type différent, la fonction renverra 1. Exemple : Entier e = Entier_Creer(5) ; Entier e2 = Clone(e) ; Entier e3 = Entier_Creer(4) ; Egal(e, e2) ; (= 0) Egal(e, e3) ; (= 1)
- int Compare (Type a, Type b)
  Les comparaisons : La fonction Compare(Type a, Type b) compare un Type a à un Type B. Exemple : Entier e = Entier_Creer(5) ; Entier e1 = Entier_Creer(4) ; Flottant e2 = Flottant_Creer(5.) ; Compare(e,e1) ; (=-1) Compare(e,e2) ; (=-2 Un message d'erreur est ecrit sur la sortie standard)
- void Affiche (Type a)
  Affichage : La fonction void Affiche(Type a) affichera dans la sortie standard la valeur stockée dans a. Exemple : Entier e = Entier_Creer(5) ; Affiche(e) ; > 5.
- void * Eval (Type a)
  Récupérer la valeur : On récupère la valeur : La fonction void Eval(Type a) renvoie un pointeur vers la valeur stockée dans a. Exemple : Entier e = Entier_Creer(5) ; int* v = Eval(e) ; Attention : Modifier v modifiera e.*
- void Modifie (Type a, void *b)
  Modifie la valeur : La fonction void Modifie(Type a, void b) modifiera la valeur stockée dans a par la valeur pointée par b. Dans le cas ou b est d'un type différent de a, un cast est effectué. Exemple : Entier e = Entier_Creer(5) ; char c = 'a' ; Modifie(e, &c) ; (e vaudra 97)*
- String toString (Type a)
  Convertir en String : La fonction String toString(Type a) renvoie un String contenant la valeur stockée dans a. Exemple : Entier e = Entier_Creer(5) ; String s = toString(e) ; (s vaudra "5")
- void String_Concat (String a, String b)
  Concaténation : La fonction void String_Concat(String a, String b) concatène la chaine de caractère contenue dans a et celle contenue dans b et la stock dans a. Exemple : String s1 = String_Creer("Salut") ; String s2 = String_Creer(" lecteur !") ; String_Concat(s1,s2) ; Afficher(s1) ; > Salut lecteur !
- int String_Taille (String a)
  Connaître la taille du String : Int String_Taille(String a) renvoie la taille de la chaine de caractères contenue dans a. Exemple : String s = String_Creer("Coucou") ; int t = String_Taille(s) ; (t vaudra 6)
- char String_At (String a, int indice)
  Accéder à un caractère d'un String : Char String_At(String a, int i) renvoie le caractère à l'indice i de la chaine de caractères contenue dans a. Exemple : String s = String_Creer("Coucou") ; char c = String_At(s, 2) ; (c vaudra 'u')

5.2.1 Documentation des fonctions

5.2.1.1 void Affiche (Type *a*)

Affichage : La fonction void Affiche(Type a) affichera dans la sortie standard la valeur stockée dans a. Exemple : Entier e = Entier_Creer(5) ; Affiche(e) ; > 5.

Paramètres

a	La variable dont on veut afficher la valeur.

5.2.1.2 Car Car_Creer (char *valeur*)

Le constructeur Car : Ce constructeur va nous permettre de créer une variable de type Car avec la syntaxe suivante : Car <variable> = Car_Creer(<valeur>).

Paramètres

valeur	La valeur à stocker dans la variable.

Renvoie

valeur déclarée dans un type Car.

5.2.1.3 void* Clone (Type *object*)

Les copies de variables : Pour copier une variable, il faut utiliser void* Clone(Type a). Pour cela, on utilisera un switch pour nous permettre de choisir suivant le type de object. Exemple : Entier e = Entier_Creer(5) ; Entier e2 = Clone(e) ;.

Paramètres

object	La variable à copier.

Renvoie

une copie de object, NULL en cas d'erreur.

5.2.1.4 int Compare (Type *a*, Type *b*)

Les comparaisons : La fonction Compare(Type a, Type b) compare un Type a à un Type B. Exemple : Entier e = Entier_Creer(5) ; Entier e1 = Entier_Creer(4) ; Flottant e2 = Flottant_Creer(5.) ; Compare(e,e1) ; (=-1) Compare(e,e2) ; (=-2 Un message d'erreur est ecrit sur la sortie standard)

Paramètres

a	La variable à gauche de l'inégalité.
b	La variable à droite de l'inégalité.

Renvoie

-1 si a < b, 0 si a=b, 1 si a > b, et -2 en cas d'érreur.

5.2.1.5 Entier void Detruire (Type *object*)

Les destructeurs : Ces destructeur vont nous permettre de libérer la zone mémoire allouée par une variable a. On utilisera la fonction void Detruire(Type a).

Paramètres

object	La variable à détruire.

5.2.1.6 int Egal (Type *a,* Type *b*)

Les tests d'égalité : La fonction int Egal(Type a, Type b) renvoie 0 si a=b, 1 sinon. A noter que si a et b sont de type différent, la fonction renverra 1. Exemple : Entier e = Entier_Creer(5) ; Entier e2 = Clone(e) ; Entier e3 = Entier_Creer(4) ; Egal(e, e2) ; (= 0) Egal(e, e3) ; (= 1)

Paramètres

a	La variable à gauche de l'égalité à tester.
b	La variable à droite de l'égalité à tester.

Renvoie

renvoie 0 si égal sinon renvoie 1

5.2.1.7 Entier Entier_Creer (int *valeur*)

Le constructeur Entier : Ce constructeur va nous permettre de créer une variable de type Entier avec la syntaxe suivante : Entier <variable> = Entier_Creer(<valeur>).

Paramètres

valeur	La valeur à stocker dans la variable.

Renvoie

valeur déclarée dans un type Entier.

5.2.1.8 void* Eval (Type *a*)

Récupérer la valeur : On récupère la valeur : La fonction void* Eval(Type a) renvoie un pointeur vers la valeur stockée dans a. Exemple : Entier e = Entier_Creer(5) ; int* v = Eval(e) ; Attention : Modifier v modifiera e.

Paramètres

a	La variable dont on veut récupérer la valeur.

Renvoie

pointeur sur la valeur stockée dans la variable.

5.2.1.9 Flottant Flottant_Creer (float *valeur*)

Le constructeur Flottant : Ce constructeur va nous permettre de créer une variable de type Flottant avec la syntaxe suivante : Flottant <variable> = Flottant_Creer(<valeur>).

Paramètres

valeur	La valeur à stocker dans la variable.

Renvoie

valeur déclarée dans un type Flotant.

5.2.1.10 void Modifie (Type *a,* void * *b*)

Modifie la valeur : La fonction void Modifie(Type a, void* b) modifiera la valeur stockée dans a par la valeur pointée par b. Dans le cas ou b est d'un type différent de a, un cast est effectué. Exemple : Entier e = Entier_Creer(5) ; char c = 'a' ; Modifie(e, &c) ; (e vaudra 97)

Paramètres

a	La variable dont on veut modifier la valeur.
b	La nouvelle valeur stockée dans a.

5.2.1.11 char String_At (String *a,* int *indice*)

Accéder à un caractère d'un String : Char String_At(String a, int i) renvoie le caractère à l'indice i de la chaine de caractères contenue dans a. Exemple : String s = String_Creer("Coucou") ; char c = String_At(s, 2) ; (c vaudra 'u')

Paramètres

a	La variable sur laquelle s'applique la fonction.
indice	L'indice du caractère à récupérer dans la chaine de caractères.

Renvoie

le caractère à la position i stockée dans la chaine de caractères contenue dans a, '\0' en cas d'erreur.

5.2.1.12 void String_Concat (String *a*, String *b*)

Concaténation : La fonction void String_Concat(String a, String b) concatène la chaine de caractère contenue dans a et celle contenue dans b et la stock dans a. Exemple : String s1 = String_Creer("Salut") ; String s2 = String_Creer(" lecteur !") ; String_Concat(s1,s2) ; Afficher(s1) ; > Salut lecteur !

Paramètres

a	La variable à gauche de la concaténation dans laquelle on stocke le résultat de la concaténation.
b	La variable à droite de la concaténation.

5.2.1.13 String String_Creer (char * *valeur*)

Le constructeur String : Ce constructeur va nous permettre de créer une variable de type String avec la syntaxe suivante : String <variable> = String_Creer(<valeur>).

Paramètres

valeur	La valeur à stocker dans la variable.

Renvoie

valeur déclarée dans un type String.

5.2.1.14 int String_Taille (String *a*)

Connaître la taille du String : Int String_Taille(String a) renvoie la taille de la chaine de caractères contenue dans a. Exemple : String s = String_Creer("Coucou") ; int t = String_Taille(s) ; (t vaudra 6)

Paramètres

a	La variable à analyser.

Renvoie

taille de la chaine de caractère.

5.2.1.15 String toString (Type *a*)

Convertir en String : La fonction String toString(Type a) renvoie un String contenant la valeur stockée dans a. Exemple : Entier e = Entier_Creer(5) ; String s = toString(e) ; (s vaudra "5")

Paramètres

a	La variable à convertir en String.

Renvoie

un String result si cela marche sinon renvoie NULL .

5.3 Référence du fichier Documents/fac/tccp2012/Sd/TypesBase/typeBase.h

`#include <stdio.h> #include <stdlib.h> #include <string.h>`

Structures de données

- struct Type
- struct entier
- struct flottant
- struct car
- struct string

Définitions de type

- typedef struct Type * Type
  Struture globale à tous les types (équivalent à un void) : Strutures internes qui représentent tous les types.*
- typedef struct Type * Entier
  Strutures de Type Entier : Cette strutures va nous permettre de créer les types Entier . Ce type sera appelé par un int égal à 1.
- typedef struct Type * Flottant
  Strutures de Type Flottant : Cette strutures va nous permettre de créer les types Flottant . Ce type sera appelé par un int égal à 2.
- typedef struct Type * Car
  Strutures de Type Car : Cette strutures va nous permettre de créer les types Car . Ce type sera appelé par un int égal à 3.
- typedef struct Type * String
  Strutures de Type String : Cette strutures va nous permettre de créer les types String . Ce type sera appelé par un int égal à 4.
- typedef struct entier * entier
  Structure entier : Structure interne gérant les Entiers. Cette structure n'est pas manipulable par l'utilisateur.
- typedef struct flottant * flottant
  Strutures flottant : Structure interne gérant les Flottants. Cette structure n'est pas manipulable par l'utilisateur.
- typedef struct car * car
  Strutures car : Structure interne gérant les Car. Cette structure n'est pas manipulable par l'utilisateur.
- typedef struct string * string
  Strutures String : Structure interne gérant les String. Cette structure n'est pas manipulable par l'utilisateur.

Fonctions

- Entier Entier_Creer (int valeur)

Le constructeur Entier : Ce constructeur va nous permettre de créer une variable de type Entier avec la syntaxe suivante : Entier <variable> = Entier_Creer(<valeur>).

- Flottant Flottant_Creer (float valeur)
  Le constructeur Flottant : Ce constructeur va nous permettre de créer une variable de type Flottant avec la syntaxe suivante : Flottant <variable> = Flottant_Creer(<valeur>).
- Car Car_Creer (char valeur)
  Le constructeur Car : Ce constructeur va nous permettre de créer une variable de type Car avec la syntaxe suivante : Car <variable> = Car_Creer(<valeur>).
- String String_Creer (char *valeur)
  Le constructeur String : Ce constructeur va nous permettre de créer une variable de type String avec la syntaxe suivante : String <variable> = String_Creer(<valeur>).
- void Detruire (Type objet)
  Les destructeurs : Ces destructeur vont nous permettre de libérer la zone mémoire allouée par une variable a. On utilisera la fonction void Detruire(Type a).
- void * Clone (Type objet)
  Les copies de variables : Pour copier une variable, il faut utiliser void Clone(Type a). Pour cela, on utilisera un switch pour nous permettre de choisir suivant le type de object. Exemple : Entier e = Entier_Creer(5) ; Entier e2 = Clone(e) ;.*
- int Egal (Type a, Type b)
  Les tests d'égalité : La fonction int Egal(Type a, Type b) renvoie 0 si a=b, 1 sinon. A noter que si a et b sont de type différent, la fonction renverra 1. Exemple : Entier e = Entier_Creer(5) ; Entier e2 = Clone(e) ; Entier e3 = Entier_Creer(4) ; Egal(e, e2) ; (= 0) Egal(e, e3) ; (= 1)
- int Compare (Type a, Type b)
  Les comparaisons : La fonction Compare(Type a, Type b) compare un Type a à un Type B. Exemple : Entier e = Entier_Creer(5) ; Entier e1 = Entier_Creer(4) ; Flottant e2 = Flottant_Creer(5.) ; Compare(e,e1) ; (=-1) Compare(e,e2) ; (=-2 Un message d'erreur est ecrit sur la sortie standard)
- void Affiche (Type objet)
  Affichage : La fonction void Affiche(Type a) affichera dans la sortie standard la valeur stockée dans a. Exemple : Entier e = Entier_Creer(5) ; Affiche(e) ; > 5.
- void * Eval (Type objet)
  Récupérer la valeur : On récupère la valeur : La fonction void Eval(Type a) renvoie un pointeur vers la valeur stockée dans a. Exemple : Entier e = Entier_Creer(5) ; int* v = Eval(e) ; Attention : Modifier v modifiera e.*
- void Modifie (Type objet, void *val)
  Modifie la valeur : La fonction void Modifie(Type a, void b) modifiera la valeur stockée dans a par la valeur pointée par b. Dans le cas ou b est d'un type différent de a, un cast est effectué. Exemple : Entier e = Entier_Creer(5) ; char c = 'a' ; Modifie(e, &c) ; (e vaudra 97)*
- String toString (Type a)
  Convertir en String : La fonction String toString(Type a) renvoie un String contenant la valeur stockée dans a. Exemple : Entier e = Entier_Creer(5) ; String s = toString(e) ; (s vaudra "5")
- void String_Concat (String a, String b)
  Concaténation : La fonction void String_Concat(String a, String b) concatène la chaine de caractère contenue dans a et celle contenue dans b et la stock dans a. Exemple : String s1 = String_Creer("Salut") ; String s2 = String_Creer(" lecteur !") ; String_Concat(s1,s2) ; Afficher(s1) ; > Salut lecteur !
- int String_Taille (String a)
  Connaître la taille du String : Int String_Taille(String a) renvoie la taille de la chaine de caractères contenue dans a. Exemple : String s = String_Creer("Coucou") ; int t = String_Taille(s) ; (t vaudra 6)
- char String_At (String a, int indice)
  Accéder à un caractère d'un String : Char String_At(String a, int i) renvoie le caractère à l'indice i de la chaine de caractères contenue dans a. Exemple : String s = String_Creer("Coucou") ; char c = String_At(s, 2) ; (c vaudra 'u')

5.3.1 Documentation des définitions de type

5.3.1.1 typedef struct Type* Car

Strutures de Type Car : Cette strutures va nous permettre de créer les types Car . Ce type sera appelé par un int égal à 3.

5.3.1.2 typedef struct car* car

Strutures car : Structure interne gérant les Car. Cette structure n'est pas manipulable par l'utilisateur.

5.3.1.3 typedef struct Type* Entier

Strutures de Type Entier : Cette strutures va nous permettre de créer les types Entier . Ce type sera appelé par un int égal à 1.

5.3.1.4 typedef struct entier* entier

Structure entier : Structure interne gérant les Entiers. Cette structure n'est pas manipulable par l'utilisateur.

5.3.1.5 typedef struct Type* Flottant

Strutures de Type Flottant : Cette strutures va nous permettre de créer les types Flottant . Ce type sera appelé par un int égal à 2.

5.3.1.6 typedef struct flottant* flottant

Strutures flottant : Structure interne gérant les Flottants. Cette structure n'est pas manipulable par l'utilisateur.

5.3.1.7 typedef struct Type* String

Strutures de Type String : Cette strutures va nous permettre de créer les types String . Ce type sera appelé par un int égal à 4.

5.3.1.8 typedef struct string* string

Strutures String : Structure interne gérant les String. Cette structure n'est pas manipulable par l'utilisateur.

5.3.1.9 typedef struct Type* Type

Struture globale à tous les types (équivalent à un void*) : Strutures internes qui représentent tous les types.

5.3.2 Documentation des fonctions

5.3.2.1 void Affiche (Type *a*)

Affichage : La fonction void Affiche(Type a) affichera dans la sortie standard la valeur stockée dans a. Exemple : Entier e = Entier_Creer(5) ; Affiche(e) ; > 5.

Paramètres

a	La variable dont on veut afficher la valeur.

5.3.2.2 Car Car_Creer (char *valeur*)

Le constructeur Car : Ce constructeur va nous permettre de créer une variable de type Car avec la syntaxe suivante : Car <variable> = Car_Creer(<valeur>).

Paramètres

valeur	La valeur à stocker dans la variable.

Renvoie

valeur déclarée dans un type Car.

5.3.2.3 void* Clone (Type *object*)

Les copies de variables : Pour copier une variable, il faut utiliser void* Clone(Type a). Pour cela, on utilisera un switch pour nous permettre de choisir suivant le type de object. Exemple : Entier e = Entier_Creer(5) ; Entier e2 = Clone(e) ;.

Paramètres

object	La variable à copier.

Renvoie

une copie de object, NULL en cas d'erreur.

5.3.2.4 int Compare (Type *a*, Type *b*)

Les comparaisons : La fonction Compare(Type a, Type b) compare un Type a à un Type B. Exemple : Entier e = Entier_Creer(5) ; Entier e1 = Entier_Creer(4) ; Flottant e2

= Flottant_Creer(5.) ; Compare(e,e1) ; (=-1) Compare(e,e2) ; (=-2 Un message d'erreur est ecrit sur la sortie standard)

Paramètres

a	La variable à gauche de l'inégalité.
b	La variable à droite de l'inégalité.

Renvoie

-1 si a $<$ b, 0 si a=b, 1 si a $>$ b, et -2 en cas d'érreur.

5.3.2.5 void Detruire (Type *object*)

Les destructeurs : Ces destructeur vont nous permettre de libérer la zone mémoire allouée par une variable a. On utilisera la fonction void Detruire(Type a).

Paramètres

object	La variable à détruire.

5.3.2.6 int Egal (Type *a,* Type *b*)

Les tests d'égalité : La fonction int Egal(Type a, Type b) renvoie 0 si a=b, 1 sinon. A noter que si a et b sont de type différent, la fonction renverra 1. Exemple : Entier e = Entier_Creer(5) ; Entier e2 = Clone(e) ; Entier e3 = Entier_Creer(4) ; Egal(e, e2) ; (= 0) Egal(e, e3) ; (= 1)

Paramètres

a	La variable à gauche de l'égalité à tester.
b	La variable à droite de l'égalité à tester.

Renvoie

renvoie 0 si égal sinon renvoie 1

5.3.2.7 Entier Entier_Creer (int *valeur*)

Le constructeur Entier : Ce constructeur va nous permettre de créer une variable de type Entier avec la syntaxe suivante : Entier <variable> = Entier_Creer(<valeur>).

Paramètres

valeur	La valeur à stocker dans la variable.

Renvoie

valeur déclarée dans un type Entier.

5.3.2.8 void* Eval (Type *a*)

Récupérer la valeur : On récupère la valeur : La fonction void* Eval(Type a) renvoie un pointeur vers la valeur stockée dans a. Exemple : Entier e = Entier_Creer(5) ; int* v = Eval(e) ; Attention : Modifier v modifiera e.

Paramètres

a	La variable dont on veut récupérer la valeur.

Renvoie

pointeur sur la valeur stockée dans la variable.

5.3.2.9 Flottant Flottant_Creer (float *valeur*)

Le constructeur Flottant : Ce constructeur va nous permettre de créer une variable de type Flottant avec la syntaxe suivante : Flottant <variable> = Flottant_Creer(<valeur>).

Paramètres

valeur	La valeur à stocker dans la variable.

Renvoie

valeur déclarée dans un type Flotant.

5.3.2.10 void Modifie (Type *a,* void * *b*)

Modifie la valeur : La fonction void Modifie(Type a, void* b) modifiera la valeur stockée dans a par la valeur pointée par b. Dans le cas ou b est d'un type différent de a, un cast est effectué. Exemple : Entier e = Entier_Creer(5) ; char c = 'a' ; Modifie(e, &c) ; (e vaudra 97)

Paramètres

a	La variable dont on veut modifier la valeur.
b	La nouvelle valeur stockée dans a.

5.3.2.11 char String_At (String *a,* int *indice*)

Accéder à un caractère d'un String : Char String_At(String a, int i) renvoie le caractère à l'indice i de la chaine de caractères contenue dans a. Exemple : String s = String_Creer("Coucou") ; char c = String_At(s, 2) ; (c vaudra 'u')

Paramètres

a	La variable sur laquelle s'applique la fonction.
indice	L'indice du caractère à récupérer dans la chaine de caractères.

Renvoie

le caractère à la position i stockée dans la chaine de caractères contenue dans a, '\0' en cas d'erreur.

5.3.2.12 void String_Concat (String *a,* String *b*)

Concaténation : La fonction void String_Concat(String a, String b) concatène la chaine de caractère contenue dans a et celle contenue dans b et la stock dans a. Exemple : String s1 = String_Creer("Salut") ; String s2 = String_Creer(" lecteur !") ; String_Concat(s1,s2) ; Afficher(s1) ; > Salut lecteur !

Paramètres

a	La variable à gauche de la concaténation dans laquelle on stocke le résultat de la concaténation.
b	La variable à droite de la concaténation.

5.3.2.13 String String_Creer (char * *valeur*)

Le constructeur String : Ce constructeur va nous permettre de créer une variable de type String avec la syntaxe suivante : String <variable> = String_Creer(<valeur>).

Paramètres

valeur	La valeur à stocker dans la variable.

Renvoie

valeur déclarée dans un type String.

5.3.2.14 int String_Taille (String *a*)

Connaître la taille du String : Int String_Taille(String a) renvoie la taille de la chaine de caractères contenue dans a. Exemple : String s = String_Creer("Coucou") ; int t = String_Taille(s) ; (t vaudra 6)

Paramètres

a	La variable à analyser.

Renvoie

taille de la chaine de caractère.

5.3.2.15 String toString (Type *a*)

Convertir en String : La fonction String toString(Type a) renvoie un String contenant la valeur stockée dans a. Exemple : Entier e = Entier_Creer(5) ; String s = toString(e) ; (s

vaudra "5")

Paramètres

a	La variable à convertir en String.

Renvoie

un String result si cela marche sinon renvoie NULL .