Bioopo
bioopo@neuf.fr

DEBUTER EN LANGAGE C++

Calculatrice BVOS 2008

Version BVOS-0.01-0
Le 29 octobre 2007

Introduction

Bienvenue dans mon tutoriel concernant l'apprentissage de base du c++. Son apprentissage est un long chemin où découragement, blocages et agacement font partie du quotidien, mais la joie ressentie lors d'un challenge réussi (je ne parle pas d'un programme , mais de l'avancement de l'apprentissage dans la douleur) est bien plus grande que si l'on fait la même chose dans un langage plus facile et compréhensible. Voilà comment je vois l'apprentissage du c++, il faut être motivé, prendre son temps et ne pas avoir peur des échecs. Une fois que vous avez compris cela, vous pouvez vous lancer dans l'apprentissage du langage.

Ce tutoriel n'est pas une bible ni un cours magistral du c++, il faut le prendre pour ce qu'il est, ma vision de l'apprentissage de ce beau langage. Il n'expliquera pas tout, ce n'est pas son rôle, il faudra faire des recherches par vous même, internet regorge de données sur le code du c++. Il vous montrera une méthodologie personnelle de conception. Ce tutoriel est conçu à travers d'exemples, pour le début, une calculatrice basique, qui permet de se concentrer sur l'essentiel sans se préoccuper du superflu (puisqu'il n'y en a pas) .

Je vais vous expliquer comment sont construits ces tutoriaux, Il y a six grandes parties :

SECTION	*DESCRIPTION*
Introduction	Introduction des grandes lignes du projet en cours
Pré-conception	Analyse des besoins
Analyses	Analyses fonctionnelles et structurelles
Développement	Phase de codage
Post-développement	Phase de correction, détection des bugs
Conclusion	Conclusion du projet en cours

Ce document est une mise à jour de mon tutoriel de base, ce changement de dix sections en six, correspond à une amélioration de la compréhension et de la lisibilité. Toutes les parties de fond du document ont été conservées et améliorées.

Le tutoriel contiendra plusieurs types d'informations, avec un format spécifique à chaque type :

Les éléments normaux.

- Les remarques importantes.

Pour le code du programme, il sera noté comme suit :

Nom du fichier

.....

Nom de la fonction

code déjà écrit

nouveau code

.....

Un petit mot sur le projet, la calculatrice que nous allons créer à travers les tutoriaux sera une calculatrice en mode console, avec les opérations standards. Elle sera capable d'effectuer les additions, soustractions , multiplications, divisions ....ainsi que plusieurs opérations avancées que nous détaillerons dans un autre tutoriel. Car d'abord, il faut apprendre les bases et le superflu peut vite devenir gênant.

Pré-conception

N'étant pas un grand spécialiste en ce qui concerne la rédaction d'un cahier des charges, je me contente du minimum, son but étant de définir avec précision le but du programme et les moyens misent en oeuvre.

Objectifs

Le but du programme est d'afficher le résultat d'une opération saisie par l'utilisateur.

Moyens misent en oeuvre

Nous allons utiliser comme logiciels un éditeur de texte et un compilateur c++. Pour ma part, j'utilise kwrite et g++ sous Linux.

Nous allons programmer une calculatrice en mode console. (pourquoi le mode console, simplement car pour la calculatrice que nous allons programmer, une interface graphique est inutile, surtout elle nous rendrait l'apprentissage plus difficile, ce qui n'est pas le but).

L'exécutable de la calculatrice ne devra pas dépasser 500 Ko.

Connaissances

Il n'y a pas de besoin d'avoir des connaissances en c++ pour ce tutoriel, car c'est un tutoriel pour les personnes souhaitant débuter avec ce langage.

Analyses

L'analyse fonctionnelle sert à déterminer les différentes fonctions du programme, je parle des différentes portions qui, mis en inter-connexion parviendront à exécuter le programme.

Il n'y a pas de code, pas de langage c++, mais simplement du français, le but de l'analyse fonctionnelle étant de découper les différentes actions à entreprendre et de les connecter.

La notation des fonctions dans l'analyse fonctionnelle, il y a les fonctions principales, les fonctions secondaires et les fonctions annexes. Les fonctions principales, notées Fpx (x étant le numéro de la fonction, correspond au grandes lignes du programme. Les fonctions secondaires découpent les fonctions principales en plus actions localisées, elle sont notées Fsx.y (x étant le numéro de la fonction principale à laquelle elles se rapportent, y étant le numéro de la fonction secondaire). Les fonctions annexes sont des fonctions non-nécessaire au fonctionnement du programme, mais qui peuvent apporter des petits plus. Elles sont notées Faz (z étant le numéro de la fonction annexe).

Une fonction principale spécifique est la fonction FP0. C'est en faites la racine de l'analyse fonctionnelle, elle regroupe toutes les autres fonctions principales et les fonctions annexes.

Une analyse de fonction, qu'elle soit principale, secondaire ou annexe, contient toujours six spécificités:

- Un code d'identification (voir paragraphe ci-dessus).
- Son nom.
- Le ou les objectifs de la fonction.
- Les données que nous lui fournissons.
- Les actions utilisateurs.
- Les données en sortie.

La mise à jour du documents à supprimer la liste des fonctions secondaires et l'explication détaillée. Ceci n'apportait rien aux documents et au contraire, l'alourdissait.

Il existe plusieurs méthodes pour faire une analyse fonctionnelle, nous n'en verrons qu'une, les diagrammes.

POINT D'ENTREE SORTIE DU PROGRAMME

DONNEES ENTRANTES DU PROGRAMME

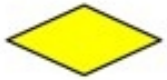
DONNEES SORTANTES DU PROGRAMME

UTILISATEUR DU PROGRAMME

ACTIONS UTILISATEUR DU PROGRAMME

FONCTIONS DU PROGRAMME

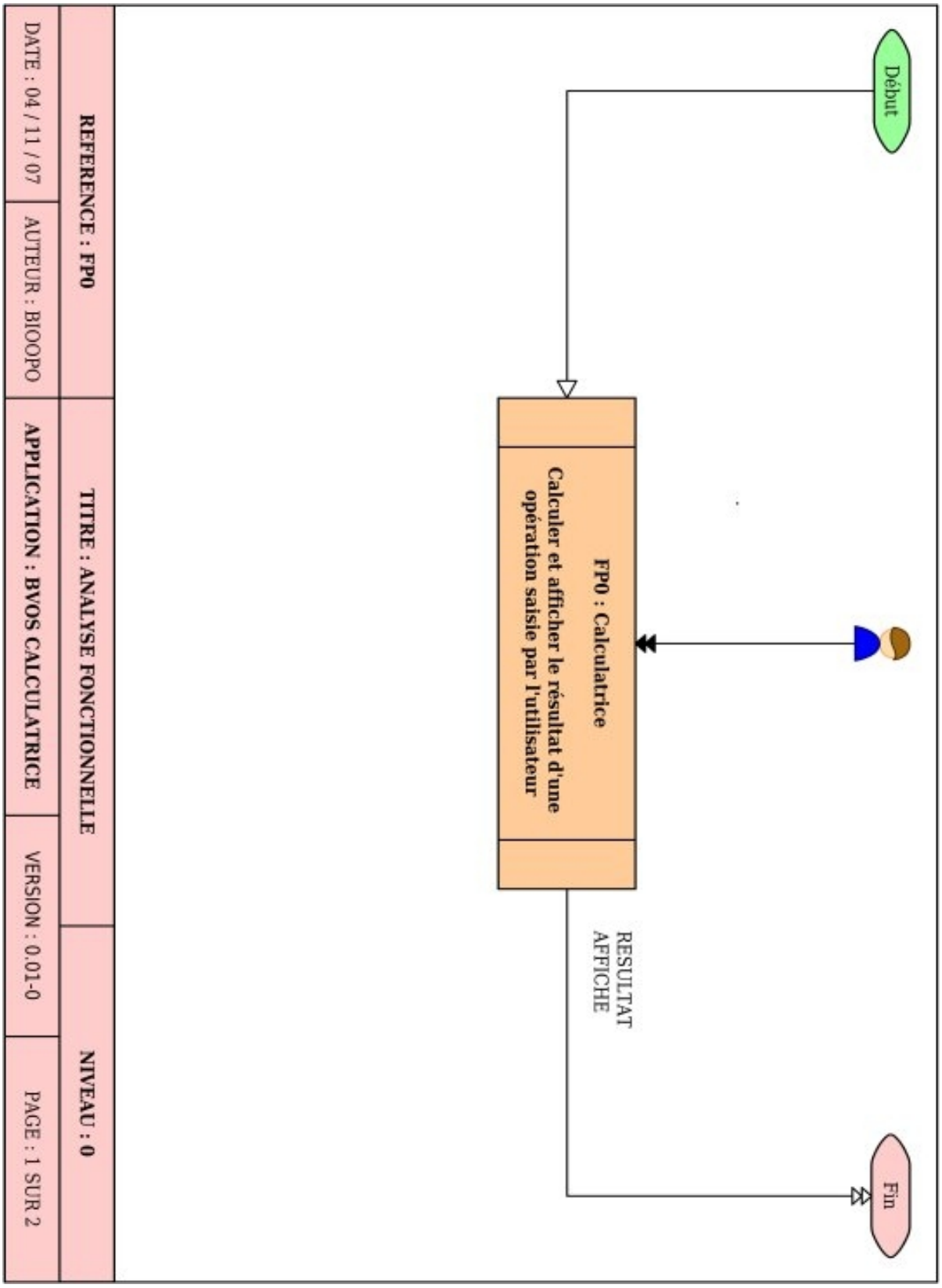
DONNEES INTERNES DU PROGRAMME



FONCTION DE TEST DU PROGRAMME

REFERENCE : LEGENDS		TITRE : LEGENDES DES OBJETS		NIVEAU : 0
DATE : 04 / 11 / 07	AUTEUR : BIOOPO	APPLICATION : BVOS CALCULATRICE	VERSION : 0.01-0	PAGE : 1 SUR 1

Fonction FP0 : Calculatrice

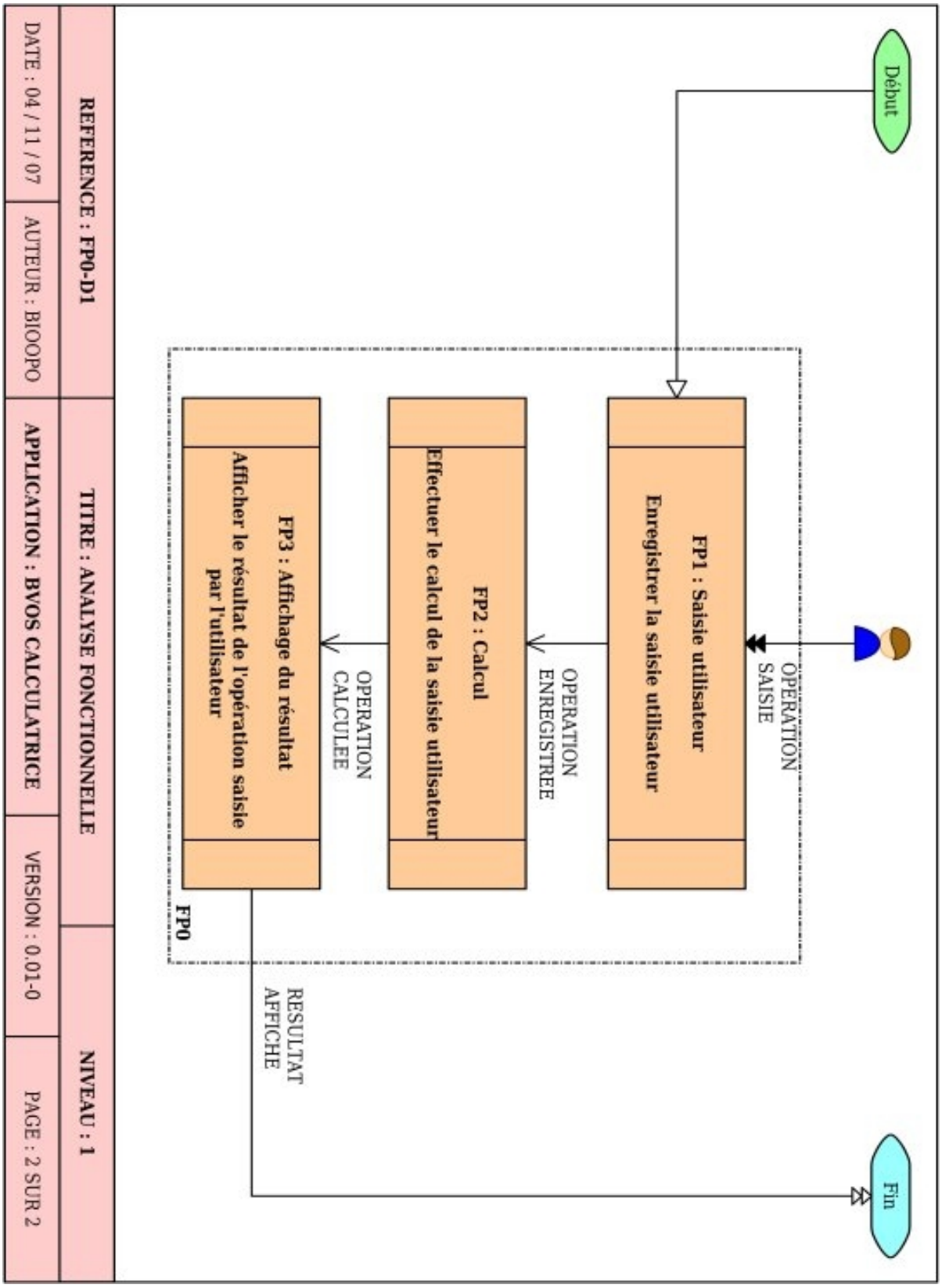


voilà, nous avons tous les éléments constituants une analyse fonctionnelle, la prochaine étape consiste à chercher comment va fonctionner notre fonction FP0 (la calculatrice). Heureusement, tout est pratiquement déjà écrit. Voici donc une liste sommaire du fonctionnement :

- Demander une saisie utilisateur et l'enregistrer.
- Faire le calcul adéquate.
- Afficher le résultat.

Nous avons dorénavant nos trois fonctions principales FP1, FP2 et FP3.

Fonction FP0-D1 : Calculatrice détaillée niveau 1



Voila, notre analyse fonctionnelle est terminée, bien sur, vous pouvez trouver ça barbant de refaire plusieurs fois les mêmes choses et faire tout ça pour un petit programme, mais habituez vous à le faire, car dans les plus gros projet, comme notre calculatrice en version 1, il sera bon d'avoir finit les fondations avant de programmer.

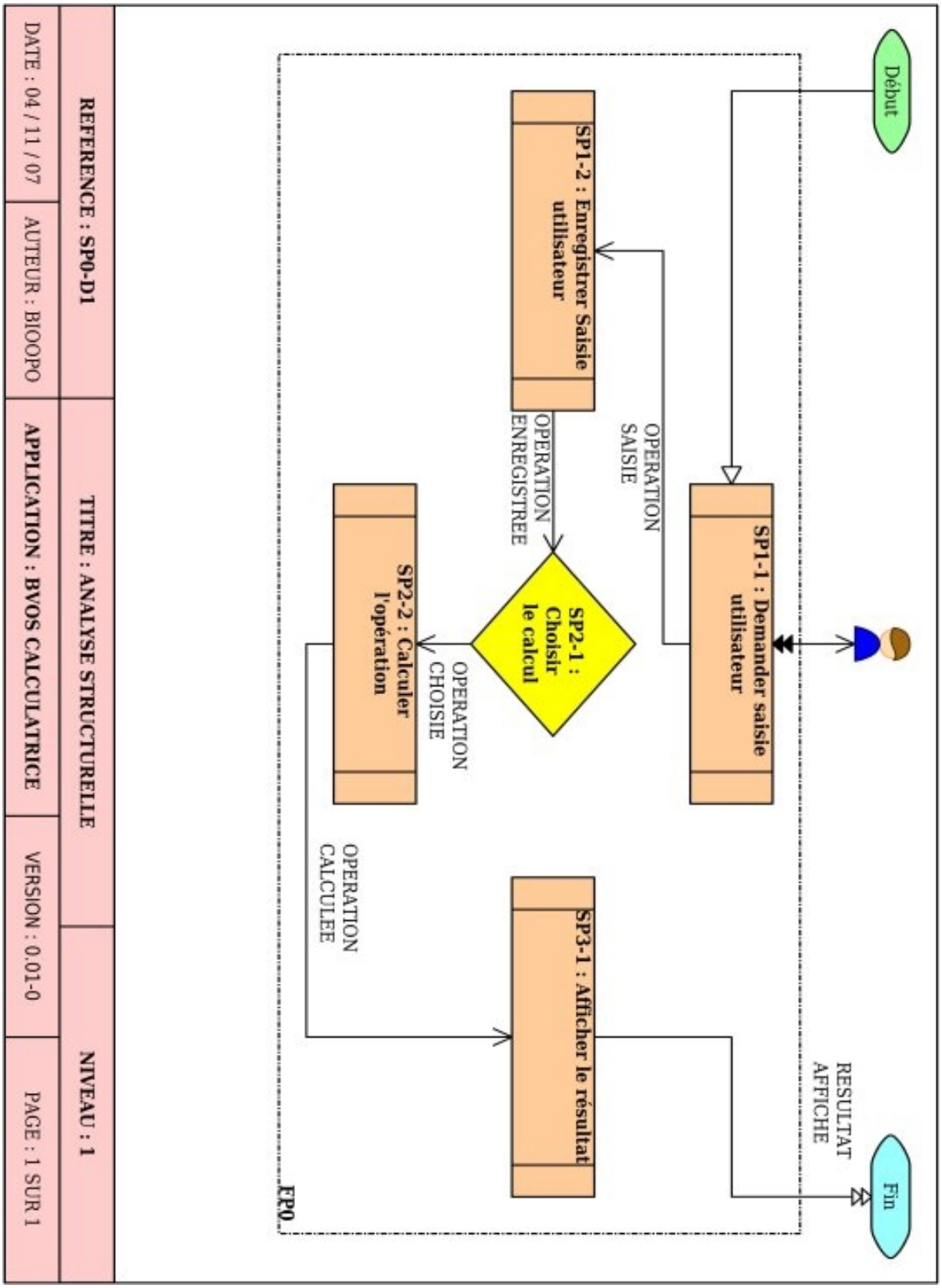
Attention, je vous déconseille, en plein milieu du programme de vouloir ajouter d'autres fonctions, il vaut mieux les noter sur un bout de papier, finir correctement votre projet, et ensuite, effectuer une nouvelle version de votre programme. Sinon, vous risquez de vous embrouiller et de faire des erreurs que vous mettrez un temps fou à réparer.

voilà, nous pouvons voir que tout s'imbrique parfaitement, nous avons donc terminé l'analyse fonctionnelle, passons donc maintenant à l'analyse structurelle.

L'analyse structurelle, comme son nom l'indique, définit les structures pour arriver à l'objectif de la fonction désirée, ceci n'est toujours pas du code, mais une logique générale de fonctionnement dans la conception d'une fonction. Plus simplement, il s'agit de définir les différentes actions à réaliser pour arriver à l'objectif de la fonction.

A la différence de l'analyse fonctionnelle, il n'y a pas de structure générique. La fonction SP0 est, en fait, La mise en relation de toutes les autres structures.

Structure SP0-D : Calculatrice (Détaillée)



Analyse des besoins de développement

Dans cette section, nous définirons les besoins du développement. Nous commencerons par expliquer les différents éléments puis, nous définirons nos besoins.

Les variables

Les variables sont des zones mémoires ou nous pouvons stocker des données, des nombres, des lettres ou d'autres types de données. Pour permettre d'allouer une zone mémoire, il faut que le programme connaisse la taille de la variable (il y a des exceptions que nous verront plus tard). Pour cela, il faut définir le type de la variable, il en existe des standards et des personnalisées, pour le moment, nous verrons les standards.

Pour stocker des nombres, il y a plusieurs types :

- int , qui signifie integer pour les nombres entiers (1 2 3 .....)
- float, pour les nombres à virgule (1,23 2,5 ....)
- double, pour les nombres à virgule
- long, pour les nombres entiers

Pour stocker les caractères comme les lettres ou les caractères spéciaux (+ - ; $), il y a un seul type de base :

- char (contient 1 caractère)

pour pouvoir utiliser ces variables, il faut les définir et les initialiser car la zone allouée peut contenir des données anciennes et créer des erreurs dans votre programme.

- Pour Déclarer une variable :

  type_de_la_variable Nom_de_la_variable;

- Pour initialiser une variable (attribuer une valeur) :

  Nom_de_la_variable = valeur;

- Pour Définir et initialiser la variable en même temps :

  type_de_la_variable Nom_de_la_variable= valeur;

exemple :

int intNumeroRue = 0;

Cela signifie que nous avons alloués une zone mémoire que le programme connaît sous le nom intNombre et que nous lui avons donnés comme valeur 0.

- Il existe un principe dans le nommage des variable, il s'agit de mettre un préfixe a la variable indiquant le type de cette variable, et de mettre un nom définissant bien le rôle de la variable. Cela rend le code plus lisible sans avoir a chercher le type et le rôle de la variable quand il s'agit d'un long programme. Prenez l'habitude de cette écriture.

Maintenant il ne nous reste plus qu'à définir les variables dont nous avons besoins. (il y aura dans le programme des variables que nous ne mettrons pas ici, car ce seront des variables pour effectuer correctement nos fonctions. Ici, nous ne définissons que les variables importantes.

Le programme doit effectuer une opération sur deux nombres et afficher le résultat. Il nous faut donc trois variables pour les nombres et un pour le signe de l'opération.

Variable	*Type*	*Init*	*Description*
intNombre1	int	0	premier nombre de l'opération
intNombre2	int	0	second nombre de l'opération
intResultat	int	0	Résultat de l'opération
chrOperand	char	'0'	signe de l'opération

Les fonctions de base

Les fonctions sont des bout de programme, qui permettent d'effectuer des taches spécifiques. On s'en sert pour éviter de ré-écrire des portions de code qui reviennent plusieurs fois, et aussi pour appeler des fonctions tel que l'écriture à l'écran, la lecture d'un fichier ....

Il existe plusieurs types de fonctions, les fonctions standard, les fonctions interne au programme et des fonctions regroupées en fichier qui peuvent être utilisée par plusieurs programmes.

Une fonction interne se déclare comme ceci :

- type_de_la_fonction Nom_de_la_fonction (arguments de la fonction);

Elle s'implante (codage de la fonction) comme ceci :

- type_de_la_fonction Nom_de_la_fonction (arguments de la fonction)

```
{
    Code de la fonction;
}
```

Nous nous contenterons pour ce tutoriel de voir la notion de fonction standard et d'en utiliser.

Nous allons utiliser dans ce tutoriel les fonctions standard de lecture au clavier pour la saisie de l'opération et de l'autre, l'affichage à l'écran pour afficher le résultat. Ces fonctions se trouvent dans des fichiers appelés bibliothèques, c'est grâce à ces fichiers que nous pouvons utiliser les fonctions standard, sinon nous devrions les écrire à chaque fois.

- La fonction cin est une fonction de base du c++, elle permet à l'utilisateur de saisir des données au clavier, puis de l'enregistrer dans une variable. Elle s'utilise comme ceci :

  cin >> Nom_De_Variable;

- La fonction cout est une fonction de base du c++, elle permet d'afficher des données à l'écran. Elle s'utilise comme ceci :

  cout << Nom_De_Variable;

Les instructions

Les instructions sont des fonctions spéciales, faites pour exécuter un code sous certaines conditions, il en existe plusieurs, mais nous allons en voir une seule, l'instruction de test conditionnel if.

L'instruction if, teste une condition, le test ne pouvant prendre que deux états, vrai ou faux, et exécute un code en fonction de ce résultat.

En français, cela pourrais ce traduire par : si la condition est vrai, exécuter le code 1 sinon exécuter le code 2.

L'instruction if dans sa forme la plus simple ne permet d'exécuter un code que si la condition est vrai. Nous détaillerons ses différentes formes dans un autre tutoriel. Elle s'utilise comme cela :

```
if(condition)
{
code a exécuter;
}
```

Les opérateurs

Les opérateurs sont des symboles servant à changer ou tester les valeurs. Ils en existent plusieurs sortes, nous ne verrons ici que les opérateurs arithmétiques et de comparaisons.

Les cinq opérateurs arithmétiques que nous verrons sont l'addition (+), la soustraction (-), la multiplication (*), la division(/) et le signe égal (=).

Le seul opérateur de comparaison que nous verrons dans ce tutoriel est l'opérateur de comparaison d'égalité (==).

- Attention, l'opérateur égal (=) et l'opérateur de comparaison d'égalité (==) ne font pas du tout la même chose, l'opérateur égal (=) attribut une valeur tandis que l'opérateur de comparaison d'égalité test si les valeurs sont identiques.

voilà, nous pouvons enfin passer au développement.

Développement

Il est grand temps maintenant de commencer le code de notre projet. Mais avant ça, je vais vous expliquer ce qu'est un projet en programmation. Un projet est l'ensemble des fichiers qui composent un programme. Pour le langage C++, il est constitué de plusieurs types de fichiers, les principaux sont les fichiers d'en-têtes (avec l'extension .h) et les fichiers sources (avec l'extension .cpp).

Que vous utilisiez un environnement de développement intégré (IDE) comme Visual C++, Dev-C++ et autres, ou que vous utilisiez un simple éditeur de texte et un compilateur séparé, c'est la même chose.

J'utilise dorénavant, pour ma part, un éditeur de texte (kwrite) et un compilateur séparé (G++) sous Linux. Mon projet va donc seulement avoir le fichier source principal du projet. Le fichier main.cpp.

Avant de nous lancer dans le développement de notre projet, regardons de plus prêt ce qu'il doit contenir.

main.cpp

```
int main()
{
    return 0;
}
```

Seul le fichier source étant obligatoire, nous commencerons donc par lui. Il n'y a qu'une seule fonction obligatoire dans un projet C++, la fonction main, qui est la fonction d'entrée du programme. La fonction main, bien que obligatoire, est une fonction interne au projet.

Dans sa version la plus simple (le cas ici), l'implémentation de la fonction main est simple. Elle est définit comme suit :

le type de retour de la fonction : int

le nom de la fonction : main

Les arguments a passer a la fonction : (), ici, il n'y en a pas

Ces arguments peuvent être des variables, des chaînes de caractères ou des nombres.

Quand il s'agit de variables, nous définissons la variable avec son type et son nom.

Les accolades : { }

c'est ici que l'on écrit le code de la fonction. La fonction débute a l'accolade d'ouverture et finit a l'accolade de fin. Selon le type de retour de la fonction, elle devra retourner une valeur. C'est la fonction de l'instruction return. Nous commencerons dans cette version a coder a l'intérieur même de la fonction pour ensuite, dans le prochain tutoriel examiner la notion de fonction.

Une des choses importantes à faire dans cette phase de programmation est de commenter notre code afin qu'il soit plus lisible par d'autres personnes et pour nous, dans les phases de recherches d'erreurs(Une des parties les plus importantes de la programmation).

Il existe en c++, deux façons de mettre des commentaires, la première est un commentaire sur une seul ligne, nous utilisons pour cela deux barres obliques pour définir que la ligne est une ligne de commentaire.

```
//ceci est un commentaire
ceci n'est pas un commentaire
```

Il y a aussi une deuxième façon, c'est de créer un bloc de commentaire qui peut être sur plusieurs ligne, nous utilisons une barre oblique suivi d'un astérisque, que nous terminons par un astérisque suivi d'une barre oblique.

```
/* ceci est un bloc de commentaire */
/*ceci est un
bloc de commentaire*/
```

/* ceci n'est pas un bon car nous n'avons pas fermer le bloc, le programme fermera le bloc a la prochaine fin de bloc.

Pour ma part, je met un entête au début de chaque fichier de mon programme. Pour notre calculatrice par exemple :

```
/**************************************
programme : BVOS calculatrice 2008
Fichier : main.cpp
Créateur : Bioopo
Mail : bioopo@neuf.fr
Création : 02/11/2007
**********************************/
```

ainsi qu'une brève description du projet :

```
/**********************************
Ce programme est une calculatrice
en console pour l'apprentissage des
bases en c++
**********************************/
```

ensuite je définit des zones pour chaque parties différentes du programmes :

Fichiers a inclure

Nom de fonction avec sa description

Initialisation des variables

Code principal de la fonctions .....

Apprendre a bien utiliser les commentaires peut vous éviter de longues nuits d'acharnement pour retrouver une erreurs.

N'oubliez pas : Beaucoup d'informations valent mieux que pas assez, mais trop d'informations (qui ne servent a rien) tuent l'information. De plus, ne mettez pas de commentaires général, par exemple

```
//variable de type double
double dblNum;
```

Ce commentaire ne sert a rien, il faut indiquez plus d'information :

```
//variable contenant le premier nombre saisie
double dblNum;
```

si vous notez correctement le nom des variables, il est inutile dans le commentaire d'inscrire le type de cette variable. De plus, il vaut mieux mettre un nom plus significatif a la variable :

```
//variable contenant le premier nombre saisie
double dblSaisie1;
```

voila un commentaire correct qui explique le but de cette variable ainsi qu'un nom de variable correct qui définit en lui même son type et son but.

Voila je pense que nous pouvons passez au code maintenant. Ajoutons donc nos commentaires

main.cpp

```
/**********************************
programme : BVOS calculatrice 2008
Fichier : main.cpp
Créateur : Bioopo
Mail : bioopo@neuf.fr
Création : 02/11/2007
**********************************/
/**********************************
Ce programme est une calculatrice
en console pour l'apprentissage des
bases en c++
**********************************/

/*****************************************************
SECTION DE FICHERS A INCLURE
*****************************************************/

/*****************************************************
SECTION DE CONFIGURATION
*****************************************************/

/*****************************************************
SECTION DE CODE
*****************************************************/

/**********************************
Fonction main
Point d'entrée du programme
**********************************/
int main()
{
    // Quitter le programme
    return 0;
}
```

La première chose à faire est d'inclure les fichiers d'en-tête dont nous aurons besoin pour appeler les fonctions standard. Ceci ce définit grâce à l'instruction #include.

La seconde chose est de définir un espace de noms standard, qui va servir à pouvoir appeler les différentes fonctions standards, sinon, le compilateur ne reconnaîtrait pas ces fonctions.

main.cpp

```
.......
/*****************************************************
SECTION DE FICHERS A INCLURE
*****************************************************/
#include <iostream>

/*****************************************************
SECTION DE CONFIGURATION
*****************************************************/
using namespace std;
.....
```

L'instruction #include sert à inclure au projet les fichiers d'en-tête, ces fichiers contiennent les déclarations des fonctions, non le code de celle-ci. Mais nous le verrons dans le prochain tutoriel. Nous incluons ici le fichier iostream. Celui-ci contient les fonctions standard d'entrées / sorties du C++.

L'instruction using namespace std sert à définir l'espace de nom à standard.

Nous allons maintenant initialiser les variables dont nous avons besoin, ceci est important, car sinon, les variables pourraient avoir des valeurs incorrect et cela serait source d'erreurs.

main.cpp

```
.....

/**********************************
Fonction main
Point d'entrée du programme
**********************************/
int main()
{
    //variable contenant les nombres saisies
    int intNombre1 = 0, intNombre2 = 0;

    //variable contenant le résultat de l'opération
    int intResultat =0;

    //variable contenant le signe de l'opération
    char chrOperand = '0';

    // Quitter le programme
    return 0;
}
```

Ensuite nous pouvons passer à la fonction FP1, la structure du code étant définit par la structure SP1.

Nous devons donc demander la saisie à l'utilisateur puis l'enregistrer dans une variable, ces deux opérations sont faites en une seule fois grâce à la fonction cin.

main.cpp

```
.....

/**********************************
Fonction main
Point d'entrée du programme
**********************************/
int main()
{
	//variable contenant les nombres saisies
	int intNombre1 = 0, intNombre2 = 0;

	//variable contenant le résultat de l'opération
	int intResultat =0;

	//variable contenant le signe de l'opération
	char chrOperand = '0';

	//Fonction permettant de saisir le premier nombre
	cin >> intNombre1;

	//Fonction permettant de saisir le signe de l'opération
	cin >> chrOperand;

	//Fonction permettant de saisir le second nombre
	cin >> intNombre2;

	// Quitter le programme
	return 0;
}
```

Maintenant passons à la fonction de calcul FP2, définit par SP2. Nous devons déterminer le signe de l'opération puis effectuer le calcul adéquat. Pour cela, nous allons utiliser l'instruction de test conditionnel if.

main.cpp

```
.....
int main()
{
	.....

	//Fonction permettant de saisir le second nombre
	cin >> intNombre2;

	//Test pour déterminer le signe de l'opération et calcul l'opération
	if(chrOperand == '+')
	{
		intResultat = intNombre1 + intNombre2;
	}
	if(chrOperand == '-')
	{
		intResultat = intNombre1 - intNombre2;
	}
	if(chrOperand == '*')
	{
		intResultat = intNombre1 * intNombre2;
	}
	if(chrOperand == '/')
	{
		intResultat = intNombre1 / intNombre2;
	}
	// Quitter le programme
	return 0;
}
```

Pour finir, codons la fonction d'affichage FP3 définit par SP3. Nous devons donc afficher le résultat de l'opération, pour cela, nous utilisons la fonction standard cout.

main.cpp

```
.....
int main()
{
	.....
	if(chrOperand == '/')
	{
		intResultat = intNombre1 / intNombre2;
	}

	//Fonction affichant le résultat de l'opération à l'écran
	cout << intResultat;

	// Quitter le programme
	return 0;
}
```

Voilà, le programme est terminé. Il ne nous reste plus qu'à tester le programme et de faire les corrections post-développement.

Post-développement

La phase de post-développement va nous permettre de tester notre programme et de voir ce qui ne va pas, puis de corriger tout cela.

La première chose à faire est de compiler le projet pour avoir un programme exécutable. Si vous utiliser un IDE ou un autre compilateur que g++, reportez vous au manuel de votre logiciel.

Pour construire l'exécutable avec g++, faîtes en mode console (Console sous Linux ou fenêtre dos sous windows) en étant dans le répertoire du projet:

g++ -o calc main.cpp

Il va vous créer l'exécutable calc. sous Linux, toujours en mode console, tapez :

./calc

entrée le chiffre 12 et appuyez sur entrée.

entrée le signe + et appuyez sur entrée.

entrée le chiffre 2 et appuyez sur entrée.

Normalement, le programme affiche 14 et retourne sur la console.

Le programme fonctionne.

La première des choses qui frappent quand on utilise le programme, c'est le manque d'informations pour l'utilisateur, nous savons quoi faire puisque c'est notre programme, mais il faut penser que dans 6 mois, nous ne nous en souviendrons plus, et si quelqu'un d'autre le lance, il aura une page vide et ne sera pas quoi faire.

- Il faut donc diriger l'utilisateur sur la fonction du programme et sur ce qu'il doit faire.

Ensuite, nous testons différentes valeurs pour les différents champs à remplir pour découvrir des bugs (erreur de programmation). Nous voyons que :

Si nous mettons un nombre a virgule, le programme ne fais pas ce qu'on lui demande.

- Il faut changer le types de variables de entier (int) à double, ainsi que le nom des variables, et faire l'initialisation de celles-ci avec 0.0 au lieu de 0.

Si nous mettons un autre caractère que les signes + - * / pour le signe de l'opération, ou si nous mettons des lettres ou autres que des nombres dans la saisie des nombres, le programme fait ne fais pas ce qu'on lui demande.

- Il faut créer un test de validité des données. Ceci se fera dans un prochain tutoriel.

Corrigeons donc le manque d'informations du programme. Un simple affichage d'informations suffit. Puis changeons les variables dans tout le programme pour accepter les nombres à virgules.

Un dernier conseil, prenez le temps de tester le programme en phase de post-développement, et commencer par des valeurs correctes, pour finir par du n'importe quoi. Puis si vous ne pouvez corriger ces erreurs, pensez à mettre des messages d'informations pour dire à l'utilisateur qu'il s'est trompé ou pourquoi le programme plante.

Voici le code mise à jour :

main.cpp

```
/**********************************
programme : BVOS calculatrice 2008
Fichier : main.cpp
Créateur : Bioopo
Mail : bioopo@neuf.fr
Création : 02/11/2007
**********************************/
/**********************************
Ce programme est une calculatrice
en console pour l'apprentissage des
bases en c++
**********************************/

/*****************************************************
SECTION DE FICHERS A INCLURE
*****************************************************/
#include <iostream>

/*****************************************************
SECTION DE CONFIGURATION
*****************************************************/
using namespace std;
```

main.cpp

```
/*****************************************************
SECTION DE CODE
*****************************************************/

/**********************************
Fonction main
Point d'entrée du programme
**********************************/
int main()
{
	//variable contenant les nombres saisies
	double dblNombre1 = 0.0, dblNombre2 = 0.0;

	//variable contenant le résultat de l'opération
	double dblResultat =0.0;

	//variable contenant le signe de l'opération
	char chrOperand = '0';

	//Fonction permettant de saisir le premier nombre
	cout << "Saisir le premier nombre" << endl;
	cin >>dblNombre1;

	//Fonction permettant de saisir le signe de l'opération
	cout << "Saisir le signe de l'opération" << endl;
	cin >> chrOperand;

	//Fonction permettant de saisir le second nombre
	cout << "Saisir le second nombre" << endl;
	cin >> dblNombre2;
```

main.cpp

```
        //Test pour déterminer le signe de l'opération et calcul l'opération
        if(chrOperand == '+')
        {
                dblResultat = dblNombre1 + dblNombre2;
        }
        if(chrOperand == '-')
        {
                dblResultat = dblNombre1 - dblNombre2;
        }
        if(chrOperand == '*')
        {
                dblResultat = dblNombre1 *  dblNombre2;
        }
        if(chrOperand == '/')
        {
                dblResultat =dblNombre1 / dblNombre2;
        }


        //Fonction affichant le résultat de l'opération à l'écran
        cout << "Le resultat est : ";
        cout << dblResultat  << endl;


        // Quitter le programme
        return 0;
}
```

Ne pas oublier d'inclure une ligne vierge en fin de fichier, pour éviter les messages d'erreurs.

Conclusion

Nous avons vu pas mal de base du c++ dans ce chapitre, apprenez a les maîtrisez avant de continuer, faites des petits programmes test, recherchez dans des livres ou sur internet des informations concernant les différents types de variables. Une fois tout cela assimilez passez au prochain tutoriel.

Base vu dans ce tutoriel:

- Définir une variable et son type. (char, int et double)
- Initialiser les variables
- faire un calcul de base en mathématique
- créer une fonction de test standard (if)
- Enregistrer une variable a partir d'une saisie au clavier (cin)
- Afficher a l'écran un message ou une variable. (cout)
- Une nomenclature pour les variables
- insérer des commentaires
- Une méthode pour bien concevoir son projet.

Je finirais ce tutoriel en vous disant de bien l'assimiler, d'essayer de faire des petits programmes utilisant les connaissances acquises avant de vous lancer dans le tutoriel suivant. Je sais que ça limite les possibilités de programmes, mais c'est important d'assimiler ces bases avant de continuer. Le c++ n'est pas un langage que l'on apprend en deux jours.

En espérant que ce tutoriel a pu vous aider et qu'il vous a plu. Bon courage pour la suite.

Bientôt le prochain tutoriel.

Merci a tous.