Compte rendu V2 du TP2 (15/11/2024)

Problème 1 (V2)

```c
#include <stdio.h>

float calcul(float a, float b, char operateur)
{
    float resultat;
    switch (operateur) {
        case '+':
            resultat = a + b;
            break;
        case '-':
            resultat = a - b;
            break;
        case '*':
            resultat = a * b;
            break;
        case '/':
            if (b == 0) {
                printf("Erreur : Division par zéro impossible\n");
                return 0;
            } else {
                resultat = a / b;
            }
            break;
        default:
            printf("Erreur : Opérateur invalide\n");
            return 0;
    }
    return resultat;
}

int main()
{
    float a, b, resultat;
    char operateur;

    printf("Entrez le premier nombre : ");
    scanf("%f", &a);

    printf("Entrez l'opérateur (+, -, *, /) : ");
    scanf(" %c", &operateur);

    printf("Entrez le deuxième nombre : ");
    scanf("%f", &b);

    resultat = calcul(a, b, operateur);

    if (resultat != 0 || (resultat == 0 && operateur == '+')) {
        printf("Le résultat est : %f\n", resultat);
    }

    return 0;
}
```

J’ai crée une deuxième fonction dédiée au calcul des valeurs obtenues dans le main().

Résultat :

```
~/CLionProjects/barbet_TP2/PB1
./a.out
Entrez le premier nombre : 10
Entrez l'opérateur (+, -, *, /) : +
Entrez le deuxième nombre : 10
Le résultat est : 20.000000
```

Problème 2 (V2)

```c
#include <stdio.h>


int sommepremiersimpairs(int n)
{
    int somme = 0;

    for (int i = 1; somme + i <= n; i += 2) {
        somme += i;
    }

    return somme;
}

int main() {
    int n, somme = 0;

    printf("Entrez un nombre entier n : ");
    scanf("%d", &n);

    somme = sommepremiersimpairs(n);

    printf("La somme des premiers nombres impairs inférieurs ou égaux à %d est : %d\n", n,
somme);

    return 0;
}
```

Pareil, j’ai crée une deuxième fonction qui gère le calcul.

Résultat :

~/CLionProjects/barbet_TP2/PB2
./a.out
Entrez un nombre entier n : 27
La somme des premiers nombres impairs inférieurs ou égaux à 27 est : 25

Problème 3 (V2):

```c
#include <stdio.h>

void affiche_table()
{
    int ligne, colonne, produit;

    printf("    ");
    for (colonne = 1; colonne <= 10; colonne++) {
        printf("%3d ", colonne);
    }
    printf("\n");

    printf("----");
    for (colonne = 1; colonne <= 10; colonne++) {
        printf("----");
    }
    printf("\n");

    for (ligne = 1; ligne <= 10; ligne++) {
        printf("%2d |", ligne);
        for (colonne = 1; colonne <= 10; colonne++) {
            produit = ligne * colonne;
            if (produit < 10) {
                printf("  %d ", produit);
            } else if (produit < 100) {
                printf(" %d ", produit);
            } else {
                printf("%d ", produit);
            }
        }
        printf("\n");
    }
}

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

J’ai mis tous les calculs dans une nouvelle fonction.

Résultat :

```
~/CLionProjects/barbet_TP2/PB3
./a.out
     1   2   3   4   5   6   7   8   9  10
----------------------------------------
 1 |  1   2   3   4   5   6   7   8   9  10
 2 |  2   4   6   8  10  12  14  16  18  20
 3 |  3   6   9  12  15  18  21  24  27  30
 4 |  4   8  12  16  20  24  28  32  36  40
 5 |  5  10  15  20  25  30  35  40  45  50
 6 |  6  12  18  24  30  36  42  48  54  60
 7 |  7  14  21  28  35  42  49  56  63  70
 8 |  8  16  24  32  40  48  56  64  72  80
 9 |  9  18  27  36  45  54  63  72  81  90
10 | 10  20  30  40  50  60  70  80  90 100
```

Problème 4 :

principalTP2.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "../include/header.h"

void menu()
{
   int choix;

   do {
      printf("\nQue desirez-vous faire :\n");
      printf("(1) Effectuer un calcul.\n");
      printf("(2) Calculer la somme des premiers impairs.\n");
      printf("(3) Afficher la table de multiplication.\n");
      printf("(0) Quitter.\n");
      printf("Sélectionnez 0, 1, 2 ou 3 puis appuyez sur Entrée.\n");
      printf("--> ");
      scanf("%d", &choix);
      getchar();

      switch (choix) {
         case 1:
            effectuerCalcul();
            break;
         case 2: {
            int n;
            printf("Entrez un entier : ");
            scanf("%d", &n);
            getchar();
            sommeImpairs(n);
            break;
         }
         case 3:
            afficherTableMultiplication();
            break;
         case 0:
            printf("Au revoir !\n");
            break;
         default:
            printf("Choix invalide.\n");
      }
   } while (choix != 0);
}

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

effectuerCalcul.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "../include/header.h"

int effectuerCalcul()
{
   float a, b, resultat;
   char operateur;

   printf("Entrez le premier nombre : ");
   scanf("%f", &a);

   printf("Entrez l'opérateur (+, -, *, /) : ");
   scanf(" %c", &operateur);

   printf("Entrez le deuxième nombre : ");
```

```c
    scanf("%f", &b);

    switch (operateur) {
        case '+':
            resultat = a + b;
            break;
        case '-':
            resultat = a - b;
            break;
        case '*':
            resultat = a * b;
            break;
        case '/':
            if (b == 0) {
                printf("Erreur : Division par zéro impossible\n");
                return 1;
            } else {
                resultat = a / b;
            }
            break;
        default:
            printf("Erreur : Opérateur invalide\n");
            return 1;
    }

    printf("Le résultat est : %f\n", resultat);

    return 0;
}
```

sommeImpairs.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "../include/header.h"

int sommeImpairs(int n)
{
    int somme = 0;

    for (int i = 1; somme + i <= n; i += 2) {
        somme += i;
    }

    printf("La somme des premiers nombres impairs inférieurs ou égaux à %d est : %d\n", n, somme);

    return 0;
}
```

afficherTableMultiplication.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "../include/header.h"

void afficherTableMultiplication()
{
    int ligne, colonne, produit;


    printf("----");
    for (colonne = 1; colonne <= 10; colonne++) {
        printf("----");
    }
    printf("\n");

    for (ligne = 1; ligne <= 10; ligne++) {
        printf("%2d |", ligne);
        for (colonne = 1; colonne <= 10; colonne++) {
```

```
            produit = ligne * colonne;
            if (produit < 10) {
                printf("  %d ", produit);
            } else if (produit < 100) {
                printf(" %d ", produit);
            } else {
                printf("%d ", produit);
            }
        }
        printf("\n");
    }
}
    printf("    ");
    for (colonne = 1; colonne <= 10; colonne++) {
        printf("%3d ", colonne);
    }
    printf("\n");

    printf("----");
    for (colonne = 1; colonne <= 10; colonne++) {
        printf("----");
    }
    printf("\n");

    for (ligne = 1; ligne <= 10; ligne++) {
        printf("%2d |", ligne);
        for (colonne = 1; colonne <= 10; colonne++) {
            produit = ligne * colonne;
            if (produit < 10) {
                printf("  %d ", produit);
            } else if (produit < 100) {
                printf(" %d ", produit);
            } else {
                printf("%d ", produit);
            }
        }
        printf("\n");
    }
}
```

Résultat :

```
~/CLionProjects/barbet_TP2/PB4
./a.out

Que desirez-vous faire :
(1) Effectuer un calcul.
(2) Calculer la somme des premiers impairs.
(3) Afficher la table de multiplication.
(0) Quitter.
Sélectionnez 0, 1, 2 ou 3 puis appuyez sur Entrée.
--> 3
     1   2   3   4   5   6   7   8   9  10
-------------------------------------------------
 1 |  1   2   3   4   5   6   7   8   9  10
 2 |  2   4   6   8  10  12  14  16  18  20
 3 |  3   6   9  12  15  18  21  24  27  30
 4 |  4   8  12  16  20  24  28  32  36  40
 5 |  5  10  15  20  25  30  35  40  45  50
 6 |  6  12  18  24  30  36  42  48  54  60
 7 |  7  14  21  28  35  42  49  56  63  70
 8 |  8  16  24  32  40  48  56  64  72  80
 9 |  9  18  27  36  45  54  63  72  81  90
10 | 10  20  30  40  50  60  70  80  90 100

Que desirez-vous faire :
(1) Effectuer un calcul.
(2) Calculer la somme des premiers impairs.
(3) Afficher la table de multiplication.
(0) Quitter.
Sélectionnez 0, 1, 2 ou 3 puis appuyez sur Entrée.
--> 0
Au revoir !
```

Makefile :

```
CC = gcc
CFLAGS = -Wall -Wextra -Werror

TARGETS = calculatrice calculatriceAvecFonction sommePremiersImpairs
sommePremiersImpairsAvecFonction tableMultiplication tableMultiplicationAvecProcedure
principalTP2

SRC_CALCULATRICE = PB1/calculatrice.c
SRC_CALCULATRICE_AVEC_FONCTION = PB1/calculatriceAvecFonction.c
SRC_SOMMEPREMIERSIMPAIRS = PB2/sommePremiersImpairs.c
SRC_SOMMEPREMIERSIMPAIRS_AVEC_FONCTION = PB2/sommePremiersImpairsAvecFonction.c
SRC_TABLEMULTIPLICATION = PB3/tableMultiplication.c
SRC_TABLEMULTIPLICATION_AVEC_PROCEDURE = PB3/tableMultiplicationAvecProcedure.c
SRC_PRINCIPALTP2 = PB4/principalTP2.c
SRC_AFFICHERTABLEMULTIPLICATION = PB4/afficherTableMultiplication.c
SRC_SOMMEIMPAIRS = PB4/sommeImpairs.c
SRC_EFFECTUERCALCUL = PB4/effectuerCalcul.c

OBJ_CALCULATRICE = $(SRC_CALCULATRICE:.c=.o)
OBJ_CALCULATRICE_AVEC_FONCTION = $(SRC_CALCULATRICE_AVEC_FONCTION:.c=.o)
OBJ_SOMMEPREMIERSIMPAIRS = $(SRC_SOMMEPREMIERSIMPAIRS:.c=.o)
OBJ_SOMMEPREMIERSIMPAIRS_AVEC_FONCTION = $
(SRC_SOMMEPREMIERSIMPAIRS_AVEC_FONCTION:.c=.o)
OBJ_TABLEMULTIPLICATION = $(SRC_TABLEMULTIPLICATION:.c=.o)
OBJ_TABLEMULTIPLICATION_AVEC_PROCEDURE = $
(SRC_TABLEMULTIPLICATION_AVEC_PROCEDURE:.c=.o)
OBJ_PRINCIPALTP2 = $(SRC_PRINCIPALTP2:.c=.o)
OBJ_AFFICHERTABLEMULTIPLICATION = $(SRC_AFFICHERTABLEMULTIPLICATION:.c=.o)
OBJ_SOMMEIMPAIRS = $(SRC_SOMMEIMPAIRS:.c=.o)
OBJ_EFFECTUERCALCUL = $(SRC_EFFECTUERCALCUL:.c=.o)

all: $(TARGETS)

calculatrice: $(OBJ_CALCULATRICE)
	$(CC) $(CFLAGS) -o $@ $^

calculatriceAvecFonction: $(OBJ_CALCULATRICE_AVEC_FONCTION)
	$(CC) $(CFLAGS) -o $@ $^

sommePremiersImpairs: $(OBJ_SOMMEPREMIERSIMPAIRS)
	$(CC) $(CFLAGS) -o $@ $^

sommePremiersImpairsAvecFonction: $(OBJ_SOMMEPREMIERSIMPAIRS_AVEC_FONCTION)
	$(CC) $(CFLAGS) -o $@ $^

tableMultiplication: $(OBJ_TABLEMULTIPLICATION)
	$(CC) $(CFLAGS) -o $@ $^

tableMultiplicationAvecProcedure: $(OBJ_TABLEMULTIPLICATION_AVEC_PROCEDURE)
	$(CC) $(CFLAGS) -o $@ $^

principalTP2: $(OBJ_PRINCIPALTP2) $(OBJ_AFFICHERTABLEMULTIPLICATION) $
(OBJ_SOMMEIMPAIRS) $(OBJ_EFFECTUERCALCUL)
	$(CC) $(CFLAGS) -o $@ $^

%.o: %.c
	$(CC) $(CFLAGS) -c $< -o $@

clean:
```

```
	rm -f $(OBJ_CALCULATRICE) $(OBJ_CALCULATRICE_AVEC_FONCTION) $
(OBJ_SOMMEPREMIERSIMPAIRS) $(OBJ_SOMMEPREMIERSIMPAIRS_AVEC_FONCTION) $
(OBJ_TABLEMULTIPLICATION) $(OBJ_TABLEMULTIPLICATION_AVEC_PROCEDURE) $
(OBJ_PRINCIPALTP2) $(OBJ_AFFICHERTABLEMULTIPLICATION) $(OBJ_SOMMEIMPAIRS)

fclean: clean
	rm -f $(TARGETS)

re: fclean all

.PHONY: all clean fclean re
```