



Chapitre 1 – Expressions divers
Diverse Ausdrücke

Exercice Algo 1.1

Dans cet exercice nous voulons essayer quelques éléments en C et nous familiariser avec l'outil Visual Studio Code (VSC).

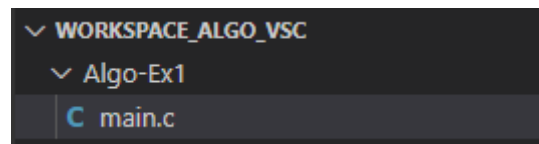
In dieser Aufgabe wollen wir ein paar Elemente in C ausprobieren und erste Schritte mit dem Tool Visual Studio Code (VSC) unternehmen.

Notes/Bemerkungen:

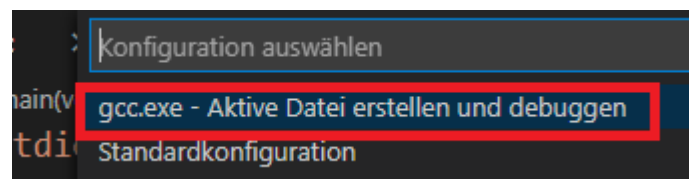
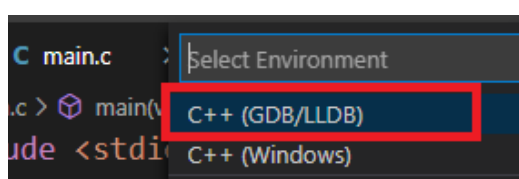
- Pour l'installation, veuillez consulter les informations sous le lien suivant:
Für die Installation bitte die Informationen in folgendem Link befolgen:
<https://cyberlearn.hes-so.ch/mod/folder/view.php?id=1290861>
- Créez un répertoire (par ex. workspace_Algo_VSC) dans lequel vous stockerez ensuite les différents exercices. Dans VSC, vous pouvez ensuite passer au répertoire via Open→Folder.
Erstellen Sie ein Verzeichnis (z.B. workspace_Algo_VSC), in welchem Sie dann die einzelnen Übungen ablegen werden. In VSC können Sie dann über Open→Folder zum Verzeichnis wechseln.
- Pour cet exercice et les autres, il est recommandé de stocker un fichier main.c dans un répertoire distinct pour chaque exercice. Créez donc le répertoire Algo-Ex1 sous le répertoire workspace_Algo_VSC et créez le fichier main.c en utilisant New->File et le contenu suivant :
Für diese und die weiteren Übungen ist zu empfehlen, jeweils eine main.c Datei in einem separaten Verzeichnis zur Übung abzulegen. Erstellen Sie also unter dem Verzeichnis workspace_Algo_VSC das Verzeichnis Algo-Ex1 und erstellen Sie die Datei main.c mittels New->File und folgendem Inhalt:

```
#include <stdio.h>

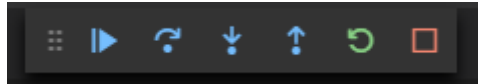
int main(void)
{
    printf("Hello world\n");
    return 0;
}
```



- Testez le programme en appelant Run→Start Debugging.
Testen Sie das Programm, indem Sie Run→Start Debugging aufrufen.



- Maintenant, éditez le fichier **launch.json**, de sorte que le programme soit arrêté lors du débogage dans main.c :
Editieren Sie nun die Datei **launch.json**, so dass das Program beim Debuggen in main.c angehalten wird :
 - "stopAtEntry": **true**,
- Répétez le test du programme en appelant Run→Start Debugging. En haut, il y a maintenant une barre de débogage, que vous pouvez utiliser pour exécuter le programme étape par étape.
Wiederholen Sie den Test des Programms, indem Sie Run→Start Debugging aufrufen. Oben befindet sich nun eine Debug-Leiste, mit welcher sie das Programm Schritt für Schritt ausführen können.



- Pour pouvoir voir la sortie de printf pendant le débogage, modifiez le fichier launch.json
Um die printf-Ausgabe beim Debuggen sehen zu können. editieren Sie nun die Datei launch.json :
 - "externalConsole": **true**,
- Pour spécifier les options du compilateur, une configuration doit être créée dans l'espace de travail : Run→Add Configuration→ C/C++ (gdb) Start. Dans le fichier **tasks.json** maintenant créé, les paramètres du compilateur peuvent maintenant être spécifiés :
Um Compiler-Optionen angeben zu können, muss im Workspace eine Konfiguration erstellt werden : Run→Add Configuration→ C/C++ (gdb) Starten.
In der nun erzeugten Datei **tasks.json** können nun die Compiler-Parameter angegeben werden :

```
"args": [
    "-g",
    "${file}",
    "-o",
    "${fileDirname}\\${fileBasenameNoExtension}.exe",
    "-Wall",
    "-std=c11"
],
```

- a) **Définissez 2 variables** dans la fonction **main** et **initialisez** les. Compilez le programme et lancez le. Que voyez vous dans la fenêtre «Problems»?
Definieren Sie in der Funktion **main 2 Variablen** und **initialisieren** Sie diese. Kompilieren Sie das Programm und führen es aus. Was sehen Sie im Fenster «Problems»?

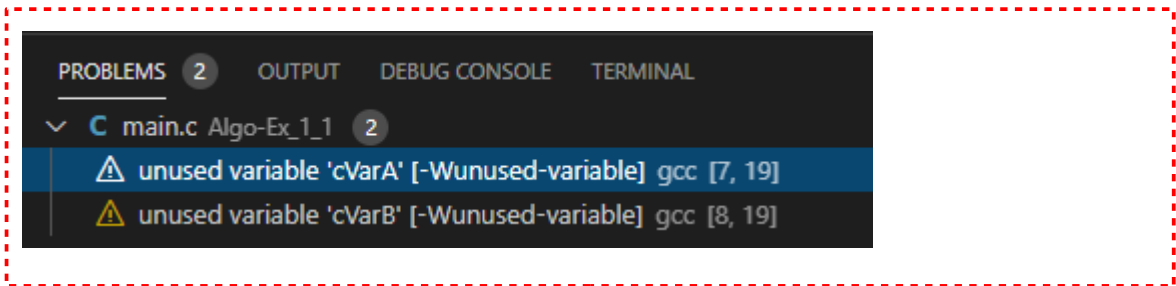


```
#include <stdio.h>

int main(void)
{
    printf("Hello world\n");

    int const cVarA = 12;
    int const cVarB = 10;

    return 0;
}
```



- b) Assignez dans une **expression** la somme de ces deux premières variables à une 3ème variable et affichez le résultat sur la console.
 Weisen Sie in einem **Ausdruck** einer 3. Variable die Summe der beiden ersten Variablen zu und geben Sie das Resultat mittels folgender Anweisung aus:

```
printf("Result is :%d \n", [expression])
```

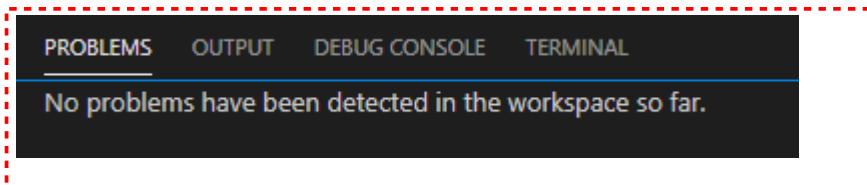
Que voyez vous dans la fenêtre «Problems»?
 Was sehen Sie im Fenster «Problems»?

```
#include <stdio.h>

int main(void)
{
    printf("Hello world\n");
    int const cVarA = 5;
    int const cVarB = 10;
    int const cVarC = cVarA+cVarB;

    printf("Result of addition is: %d\n", cVarC);

    return 0;
}
```



- c) Utilisez différents opérateurs **arithmétiques** et affichez le résultat correspondant sur la console en utilisant la fonction printf
 Verwenden Sie verschiedene **arithmetische** Operatoren und geben Sie das jeweilige Resultat mittels einer printf-Anweisung auf der Konsole aus

```
printf("Result of addition is      : %d\n", cVarA+cVarB);
printf("Result of subtraction is   : %d\n", cVarA-cVarB);
printf("Result of multiplication is: %d\n", cVarA*cVarB);
printf("Result of division is      : %d\n", cVarA/cVarB);
```

- d) **Déterminez** les types et les valeurs des expressions avec les déclarations suivantes:
Ermitteln Sie die Typen und die Werte der folgenden Ausdrücke mit folgenden
 Deklarationen:

```
unsigned char b = 5;    short s = 8;
int i = 9;             float f = 2.5;
double d = 5.2;
```

No	Expression	Type	Valeur
1	d/2	double	2.6
2	s*4	short	32
3	(b+s) > (5*f)	boolean	true
4	i/4 + d	double	7.2
5	1/3	int	0
6	b&6	char	4

```
#include <stdio.h>

int main(void)
{
    unsigned char b = 5;
    short s = 8;
    int i = 9;
    float f = 2.5;
    double d = 5.2;

    printf("Expression 1: d/2 = %.3f \n", d/2);
    printf("Expression 2: s*4 = %i \n", s*4);
    printf("Expression 3: (b+s) > (5*f) = %i \n", (b+s) > (5*f));
    printf("Expression 4: i/4 + d = %f \n", i/4 + d);
    printf("Expression 4: i/4 + d = %i \n", i/4 + d); // -858993459 ->
                                                    // needs type cast to int...

    printf("Expression 5: 1/3 = %i \n", 1/3);
    printf("Expression 5: 1/3 = %f \n", 1/3); // 7.199997 -> needs type cast or 1.0/3
    printf("Expression 6: b&6= %i \n", b&6);
    return 0;
}
```

- e) Indiquer la **valeur** des **variables** après chaque ligne de cette séquence d'instructions et indiquer aussi l'ordre d'évaluation des expressions (en ajoutant des parenthèses et en mentionnant la **précédence** et l'**associativité** des opérateurs). Vérifier vos résultats avec le debugger en Code-Blocks

Geben Sie, nach der Ausführung jeder Code-Zeile der folgenden Code-Sequenz, den **Wert der Variablen** wie auch die Reihenfolge der Evaluierung der Ausdrücke (durch Hinzufügen von Klammern und Angabe der **Präzedenz** und **Assoziativität** des Operators). Verifizieren Sie Ihre Lösungen mit Hilfe des Debuggers von Code-Blocks.

```
printf("Expression 1: a=%i b=%i j=%i k=%i\n",a,b,j,k);
```

No	Expression	a	b	j	k	Évalué comme
1	int j = 7, k = 4; bool a = true, b = true;	1	1	7	4	-
2	b = a != true;	"	0	"	"	Prio: 2 9 Ass: (l←r) (l→r) b = (a != true);
3	a = a && b && true;	0	"	"	"	Prio: 5 5 Ass: (l→r) (l→r) a && (b && true);
4	a = 7 >= 3 7 == 3;	1	"	"	"	Prio: 10 6 9 Ass: (l→r) (l→r) (l→r) (7 >= 3) (7 == 3);
5	b = k++ >= 5 j-- == 7;	"	1	6	5	Prio: 16 10 4 16 9 Ass: (lr) (lr) (lr) (lr) (lr) (k++ >= 5) (j-- == 7); évalué comme / wie folgt evaluiert : (k >= 5) (j == 7); k++; j--;
6	k = k++ + ++j;	"	"	"	12	Prio: 16 12 15 Ass: (lr) (lr) (rl) (k++) + (++j); évalué comme / wie folgt evaluiert 1): ++j; k = k + j;

1) Nicht konform gemäss C-Standard. Der Ausdruck kann durch die Implementierung des Kompilers sowohl als

```
++j;  
k = k + j;
```

als auch als

```
++j;  
k++;  
k = k + j;
```

ausgewertet werden, so dass das Resultat von k dann unterschiedlich sein kann.

gcc gibt mit -Wall auch das entsprechende Warning aus:

```
warning: operation on 'k' may be undefined [-Wsequence-point]
```

```

#include <stdio.h>
int main(void)
{
    int j = 7, k = 4;
    bool a = true, b = true;

    printf("Expression 1: a=%i b=%i j=%i k=%i\n",a,b,j,k);
    b = a != true;
    printf("Expression 2: a=%i b=%i j=%i k=%i\n",a,b,j,k);
    a = a && b && true;
    printf("Expression 3: a=%i b=%i j=%i k=%i\n",a,b,j,k);
    a = 7 >= 3 | 7 == 3;
    printf("Expression 4: a=%i b=%i j=%i k=%i\n",a,b,j,k);
    b = k++ >= 5 || j-- == 7;
    printf("Expression 5: a=%i b=%i j=%i k=%i\n",a,b,j,k);
    k = k++ + ++j;
    printf("Expression 6: a=%i b=%i j=%i k=%i\n",a,b,j,k);

    return 0;
}

```

Example for Terminal output:

a) Definition and initialization

b) Simple expression

Result of addition is: 15

c) Arithmetic operators

Result of subtraction is : 2

Result of multiplication is: 120

Result of division is : 1

d) Types and values

Expression 1: d/2 = 2.600

Expression 2: s*4 = 32

Expression 3: (b+s) > (5*f) = 1

Expression 4: i/4 + d = 7.200000

Expression 4: i/4 + d = 7

Expression 5: 1/3 = 0

Expression 5: 1/3 = 0.333333

Expression 6: b&6= 4

e) Precedence and associativity

Expression 1: a=1 b=1 j=7 k=4

Expression 2: a=1 b=0 j=7 k=4

Expression 3: a=0 b=0 j=7 k=4

Expression 4: a=1 b=0 j=7 k=4

Expression 5: a=1 b=1 j=6 k=5

Expression 6: a=1 b=1 j=7 k=12

Process returned 0 (0x0) execution time : 0.049 s