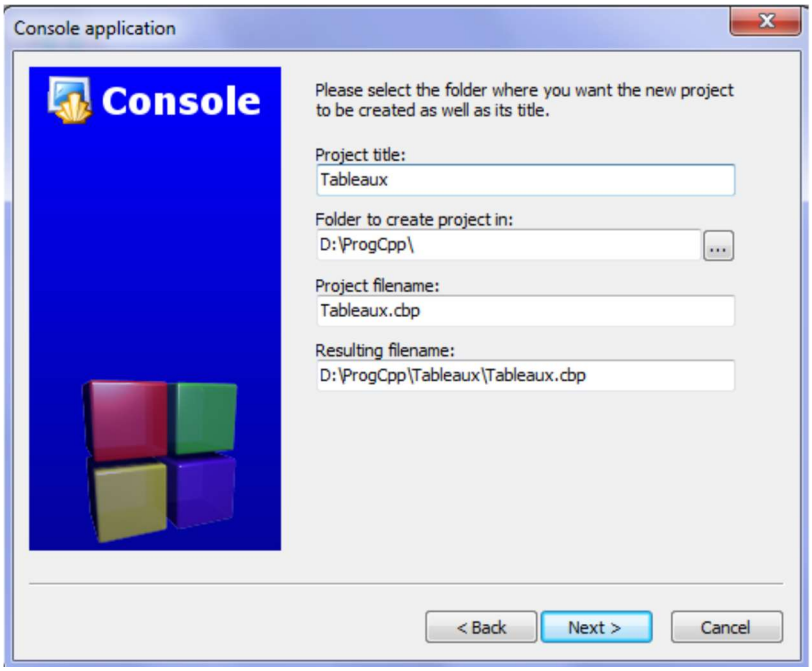
CPP2	Initiation au langage C avec Code Block – Les tableaux et les pointeurs	
Outil : Code Block	**Durée : 3 heures**	**Travail individuel**

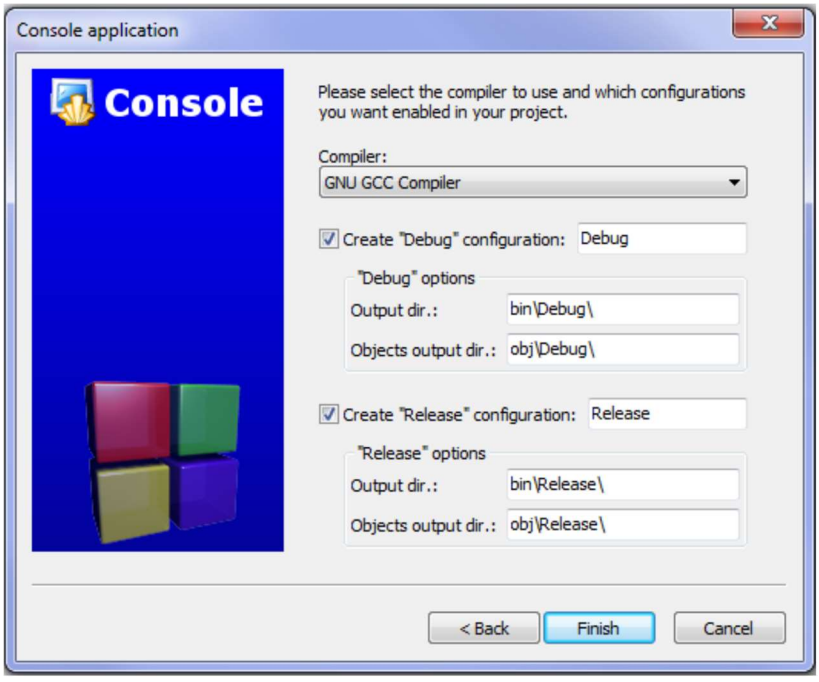
Objectifs: ⇨ Se familiariser avec la notion de tableau et de pointeur en langage C

1 – Configuration du projet

⇨ Lancer le logiciel Code Bock
⇨ Créer un nouveau projet : File / New / Project
⇨ Sélectionner le mode console : Console application / Go (puis Next)
⇨ Sélectionner le langage C : C / Next
⇨ Donner un titre au projet et définir le répertoire de sauvegarde :



⇨ Choisir le compilateur GCC (par défaut), les modes DEBUG et RELEASE. Puis Finish



2 – Fonction printf

Par défaut, Code Blocks met le fichier « main.c » comme fichier source :

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

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

La fonction printf est l'écriture sur la sortie standard en langage c (l'écran par défaut).

⇨ Tester les différents programmes ci-dessous et répondre aux questions.

1

```c
int main()
{
    printf("Hello world!\n");
    printf("Bonjour le monde \n");
    return 0;
}
```

2

```c
int main()
{
    printf("Hello world!");
    printf("Bonjour le monde \n");
    return 0;
}
```

3

```c
int main()
{
    char a=65;
    char b=66;
    printf("Valeur de a: %d\n",a);
    printf("Valeur de b: %d\n",b);
    return 0;
}
```

4

```c
int main()
{
    char a=65;
    char b=0x42;
    printf("Valeur de a: %x\n",a);
    printf("Valeur de b: %x\n",b);
    return 0;
}
```

5

```c
int main()
{
    char a=65;
    char b=0x42;
    printf("Valeur de a: %c\n",a);
    printf("Valeur de b: %c\n",b);
    return 0;
}
```

2.1 – Donner le rôle de \n

2.2 - Donner le rôle de %d

2.3 - Donner le rôle de %x

2.4 - Donner le rôle de %c

3 – Déclaration de tableaux

⇨ Tester le programme ci-dessous (utiliser copier « Ctrl +C » et coller « Ctrl + V »)

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

int main()
{
    unsigned char data[5]={5,0x6A,17,56,98};

    printf("Valeur de data[0]: %d\n",data[0]);
    printf("Valeur de data[1]: %d\n",data[1]);
    printf("Valeur de data[2]: %d\n",data[2]);
    printf("Valeur de data[3]: %d\n",data[3]);
    printf("Valeur de data[4]: %d\n",data[4]);
    printf("Taille du tableau: %d\n",sizeof(data));
    return 0;
}
```

3.1 – Modifier le programme pour initialiser data[3] à la valeur de 45, lors de la déclaration du tableau.

3.2 – Modifier le programme pour afficher les valeurs du tableau en hexadécimal.

⇨ Tester le programme ci-dessous :

```c
int main()
{
    unsigned char data[5]={5,0x6a,17,56,98};
    int i;
    for (i=0;i<5;i++)
    {
        printf("Valeur de data[%d]: %d\n",i, data[i]);
    }
    printf("Taille du tableau: %d\n",sizeof(data));
    return 0;
}
```

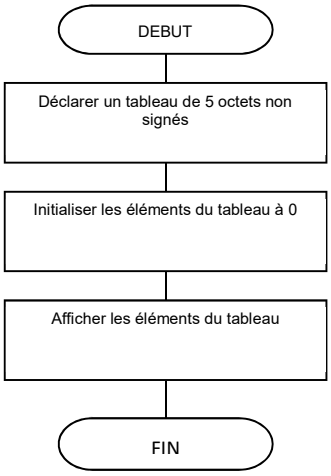
3.3 – Modifier la taille du tableau à 10 éléments (initialisés à une valeur quelconque).

3.4 – Modifier le programme pour afficher les valeurs du tableau en hexadécimal (l'indice reste en décimal).

L'initialisation des éléments d'un tableau peut se faire au moment de la déclaration du tableau (avec des accolades), comme précédemment, soit de la manière suivante :

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

int main()
{
    unsigned char data[5];
    int i;
    data[0]=65;
    data[1]=0x45;
    data[2]=34;
    data[3]='A';
    data[4]=98;
    for (i=0;i<5;i++)
    {
        printf("Valeur de data[%d]: %d\n",i, data[i]);
    }
    printf("Taille du tableau: %d\n",sizeof(data));
    return 0;
}
```

3.5 – Modifier le programme pour initialiser tous les éléments du tableau à la valeur 0, en utilisant une boucle « for » :



4 – Chaîne de caractères

Une chaîne de caractère est un cas particulier de tableau.

Le compilateur place par défaut la valeur 0 à la fin de la chaîne de caractère.

Une chaîne de caractère peut être initialisée de la manière suivante :

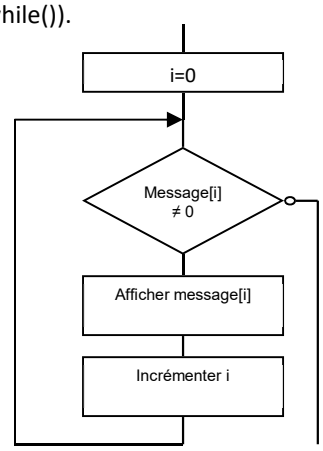
```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    char message[]="BONJOUR";
    int i;
    for (i=0;i<8;i++)
    {
        printf("Valeur de message[%d]: %d\n",i, message[i]);
    }
    printf("Taille du tableau: %d\n",sizeof(message));
    return 0;
}
```

La taille de la chaîne de caractère n'est pas précisée ici.

⇨ Tester le programme

4 .1 – Relever la valeur du dernier élément du tableau message

4.2 – Modifier le programme pour afficher tous les caractères en testant cette fois le dernier élément du tableau, comme sur l'organigramme ci-dessous (remplacer le for par un while()).



4.3 – Tester en modifiant la valeur initiale de la chaîne de caractère (remplacer « BONJOUR » par autre chose). Afficher les caractères, et non plus leur code ASCII.

5 – Adresses des éléments d'un tableau

Pour obtenir l'adresse d'une variable, il faut ajouter & devant le nom de la variable.

⇨ Tester le programme suivant :

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

int main()
{
    unsigned char data[6]={11,22,33,44,55,66};
    int i;
    i=0;
    for (i=0;i<6;i++)
    {
        printf("Valeur de data[%d]: %d",i, data[i]);
        printf(" adresse  %x:\n",&data[i]);
    }
    printf("Taille du tableau: %d\n",sizeof(data));
    return 0;
}
```

5.1 – En déduire les adresses occupées par les éléments du tableau.

⇨ Remplacer la déclaration d'un tableau d'octets (unsigned char) par un tableau d'entiers (unsigned int). Tester

5.2 – En déduire le nombre d'octets occupés par un entier.

⇨ Ajouter la ligne : printf("Valeur de data %x\n",data); comme ci-dessous :

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

int main()
{
    unsigned int data[6]={11,22,33,44,55,66};
    int i;
    i=0;
    for (i=0;i<6;i++)
    {
        printf("Valeur de data[%d]: %d",i, data[i]);
        printf(" adresse  %x:\n",&data[i]);
    }
    printf("Taille du tableau: %d\n",sizeof(data));
    printf("Valeur de data %x\n",data);
    return 0;
}
```

5.3 – Préciser à quoi correspond la valeur de data (sans indice).

6 – Pointeur

Un pointeur est une case mémoire qui contient l'adresse d'une variable.

Un pointeur se déclare avec une *.

Dans l'exemple ci-dessous :

unsigned char *p : déclaration d'un pointeur vers des octets non signés

p=&a : initialisation du pointeur avec l'adresse de la variable a

⇨ Tester le programme

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

int main()
{
    unsigned char a;
    unsigned char *p;
    a=0x58;
    p=&a;

    printf("Valeur de a: %x",a);
    printf(" adresse de a %x\n",&a);
    printf("Valeur de p: %x",p);
    printf(" adresse de p: %x\n",&p);

    return 0;
}
```

6.1 – Préciser ce que contient la variable p

⇨ Tester le programme ci-dessous :

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

int main()
{
    unsigned char a[4]={0x11,0x22,0x33,0x44};
    unsigned char *p;
    int i;
    p=&a[0];

    for (i=0;i<4;i++)
    {
        printf("Valeur de la donnee: %x",*p);
        printf (" Valeur du pointeur: %x\n",p);
        p++;
    }
    return 0;
}
```

⇨ Modifier le programme par une déclaration d'un tableau de unsigned int comme ci-dessous :

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

int main()
{
    unsigned int a[4]={0x11,0x22,0x33,0x44};
    unsigned int *p;
    int i;
    p=&a[0];

    for (i=0;i<4;i++)
    {
        printf("Valeur de la donnee: %x",*p);
        printf (" Valeur du pointeur: %x\n",p);
        p++;
    }
    return 0;
}
```

6.2 – Indiquer comment est incrémenté p (instruction p++) dans les 2 cas ?

7 – Retour de valeur par pointeur

Un sous programme ne peut retourner qu'une seule valeur.

Dans l'exemple ci-dessous, la fonction somme retourne la somme des 2 nombres d'entrée.

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

int somme(unsigned char a, unsigned char b);

int main()
{
    int i;
    i=somme(5,8);
    printf("Somme: %d\n",i);
    return 0;
}

int somme(unsigned char a, unsigned char b)
{
    return a+b;
}
```

⇨ Tester le programme ci-dessus.

Lorsqu'on souhaite retourner plusieurs valeurs, on peut utiliser des pointeurs :

Dans l'exemple ci-dessous, la fonction calcul retourne la somme et la différence de 2 nombres.

Dans cet exemple, les arguments fournis à la fonction calcul sont : la valeur de x (5), la valeur de y (8), l'adresse de i (où sera stocké le résultat de la somme) et l'adresse de j (où sera stocké le résultat de la différence).

⇨ Tester le programme ci-dessous.

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

void calcul(unsigned char a, unsigned char b, int *somme, int *diff);

int main()
{
    int i,j;
    unsigned char x,y;
    x=5;
    y=8;
    calcul(x,y,&i,&j);
    printf("Somme: %d\n",i);
    printf("Difference: %d\n",j);
    return 0;
}

void calcul(unsigned char a, unsigned char b, int *somme, int *diff)
{
    *somme=a+b;
    *diff=a-b;
}
```

7.1 – De la même manière, proposer un programme comportant un sous-programme qui retourne (par pointeur) la somme et la moyenne de 3 nombres.