

1. شرح محتوى الملف ValidatePassword.h

نبدأ مع تضمين المكتبات :

```
#include<conio.h>
#include<stdio.h>
#include<ctype.h>
#include<windows.h>
```

قمث بتضمين تلك المكتبات لأنني سأستخدم `getch & putchar` من `conio.h` و `printf & scanf` من `stdio.h` و `isdigit & iscntrl` من `ctype.h` و كل من الدالة `SetConsoleTextAttribute` و صديقته المقربة `GetStdHandle` من `windows.h`.

```
#define MAX_PASSWORD_LENGTH 21
```

الثابت `MAX_PASSWORD_LENGTH` عبارة عن الطول الأقصى الذي يمكن أن تصل إليه كلمة المرور.

```
void colorString(int couleurDuTexte, const char* str = NULL)
```

الدالة `colorString` تستقبل رقم اللون و الجملة المراد تلوينها، ثم تقوم بإظهار الجملة باللون المطلوب. الوسيط الثاني للدالة هو وسيط افتراضي لذا يمكننا استدعاء الدالة بوسيط واحد وهو رقم اللون. يتم إظهار محتوى الوسيط الثاني إذا و فقط إذا كان يُمثل سلسلة محارف فعلية.

```
void printErrorPassword(int numError, char charError = '\0')
```

الدالة `printErrorPassword` مسؤولة عن إظهار رسائل الخطأ المتعلقة بإدخال كلمة المرور. تستقبل الدالة وسيطين، أحدهما حقيقي و الآخر افتراضي، الوسيط الأول عبارة عن رقم الخطأ أما الثاني فيمثل المحرف المخالف لقواعد الإدخال. من الطبيعي جداً أن يكون الوسيط الثاني افتراضي لأنه يمثل حالة واحدة فقط (يتعلق برسالة خطأ واحدة) أما في بقية الحالات فسيتم استدعاء الدالة بوسيط واحد وهو رقم الخطأ.

في جسم الدالة نجد القرار `switch` الذي يأتي دوره في اختبار رقم الخطأ و إعطاء التحكم لد `case` المناسبة من أجل إظهار الرسالة التي تُوضح ماهية الخطأ.

و هذه ترجمة سريعة لرسائل الخطأ :

- ✓ **ال case الأولى** : لقد أدخلتم محرف غير مرئي.
 - ✓ **ال case الثانية** : عفوا، الرمز المدخل غير صحيح. (تُخالف لقواعد كلمة المرور)
 - ✓ **ال case الثالثة** : كلمة المرور المدخلة عبارة عن مسافات بيضاء فقط !
 - ✓ **ال case الافتراضية (default)** : رقم الخطأ غير معروف.
- ال case الأخيرة** تأتي فائدتها في مساعدة المُبرمج إذا تم استدعاء الدالة برقم خطأ غير معروف.

void CheckLengthPassword(int numError)

الدالة **CheckLengthPassword** مسئولة عن إظهار رسائل الخطأ المتعلقة بطول كلمة المرور. تستقبل الدالة وسيط واحد وهو رقم الخطأ. الخطأ رقم 1 يعني أن طول كلمة المرور المدخلة أقل من 10 و الخطأ الثاني يعني أن طول الأخيرة أكبر من 30.

فكرة **ال case الأخيرة** ذكرناها آنفا. نأتي الآن إلى ترجمة رسائل الخطأ :

- ✓ **ال case الأولى** : طول كلمة المرور يجب أن لا يقل عن 10 أحرف.
- ✓ **ال case الثانية** : طول كلمة المرور يجب أن لا يزيد عن 30 أحرف.

bool CheckMaxLength(int Length, int lengthSup)

الدالة **CheckMaxLength** تستقبل وسيطين. الأول عبارة عن طول كلمة المرور و الثاني يمثل الطول الأقصى. إذا كان الطول الحالي أكبر من الطول الأقصى ستعيد الدالة **true** و إلا فستعيد **false**.

bool CheckMinLength(int Length, int lengthInf)

نفس فكرة الدالة السابقة مع استبدال الطول الأقصى بالطول الأدنى.

bool HasNoneOf(char c, const char* str)

هذه الدالة تستقبل وسيطين، الأول عبارة عن محرف و الثاني سلسلة محارف، إذا كان الوسيط الأول ينتمي إلى الوسيط الثاني ستعيد الدالة **true** و إلا فستعيد **false**.

bool ValidatePassword(char* password)

هذه هي الدالة الرئيسية التي تتحقق من صحة كلمة المرور.

في البداية تم الإعلان عن سلسلة حرفية باسم `not_accepted_chars` و التي تحتوي على مجموعة الرموز المُنحالفة لقواعد كلمة المرور.

```
do {  
    ...  
} while (int(*password++) != 13);
```

يتم الخروج من الحلقة `do while` عندما يضغظ المستخدم على `.enter`.

المتغير `passwordLength` يمثل الطول الحالي لكلمة المرور بينما يتم تخزين عدد المسافات البيضاء التي تحتوي عليها كلمة المرور الحالية داخل المتغير `CheckSpace`.

تكون قيمة المتغير المنطقي `IsCntrl` مُساوية لـ `true` إذا وفقط إذا كان الرمز المُدخل عبارة عن أحد الرموز الغير مرئية. أما المتغير `CheckPasswordError` فتكون قيمته `true` إذا كان الرمز المُدخل غير مرئي أو ينتمي إلى الأحرف المحظورة في كلمة المرور.

في كل مرة، تتم قراءة الحرف الذي أدخله المستخدم ثم يُخزن هذا الحرف داخل سلسلة المحارف `password` ، بعد ذلك تعرض الدالة `ValidatePassword` على الشاشة إشارة نجمة.

تتم القراءة هنا بواسطة الدالة `getch` و بالتالي لن يظهر الحرف المُدخل.

هذا الشرط :

```
if(*password != ' ' & (int) *password != 13 & CheckPasswordError)
```

يعني : إذا كان الرمز المُدخل عبارة عن أحد الحروف المحظورة أو يمثل حرف غير مرئي باستثناء المسافة البيضاء و الـ `.enter`.

هذا الشرط :

```
if(!IsCntrl)
```

يعني أن الرمز المُدخل ليس رمز غير مرئي و بالتالي فهو أحد الحروف المحظورة لذا قمنا باستدعاء الدالة `printErrorPassword` بوسيطين، الوسيط الأول هو الخطأ رقم 2 أما الوسيط الثاني فهو الرمز المُنحالف.

else تعني أن الرمز المدخل عبارة عن محرف غير مرئي و في هذه الحالة يتم استدعاء الدالة **printErrorPassword** بالخطأ رقم واحد.

بعد ذلك، سيتم تخزين الرمز **\0** الذي يدل على نهاية السلسلة الحرفية داخل الخانة الحالية بدلا من الحرف المخالف، ثم تعيد الدالة **false**.

```
if (CheckMaxLength(passwordLength, 30))
```

في كل مرة أيضا سيتم فحص طول كلمة المرور، إذا تجاوز عدد حروفها 30 سيتم استدعاء الدالة **CheckLengthPassword** و إظهار الرسالة المتعلقة بالخطأ رقم 2 ثم تعيد الدالة **false**.

نأتي الآن لما بعد الحلقة **do while** :

الخروج من الحلقة يعني أن المستخدم ضغط على **enter** كما يعني أيضا أن طول كلمة المرور المدخلة أقل من الطول الأقصى. بقيت الآن حالتين للتحقق :

➤ هل طول كلمة المرور أقل من الطول الأدنى ؟

➤ هل كلمة المرور المدخلة تحتوي على مسافات بيضاء فقط ؟

قبل أن نبدأ التحقق يجب أن نضع الرمز **\0** في نهاية السلسلة، و هذا ما فعلناه في هذا السطر :

```
* (password - 1) = '\0';
```

السبب في كتابة **password - 1** بدلا من **password** بسيط جدا، وهو أنه بالخروج من الحلقة **do while** سيتم إزاحة المؤشر إلى الخانة الموالية و بالتالي سيصبح المؤشر **password** يشير إلى الخانة الموجودة بعد الخانة التي يوجد فيها رمز الـ **enter**.

هذا الشرط :

```
if (CheckSpace & CheckSpace == passwordLength - 1)
```

يعني : إذا كانت قيمة المتغير **CheckSpace** تختلف عن الصفر (أو لنقل أن المتغير غير معدوم و هذا معناه أن كلمة المرور غير خاوية) و عدد المسافات البيضاء يساوي عدد خانات كلمة المرور (هذا معناه أن كلمة المرور عبارة عن مسافات بيضاء فقط) فقم باستدعاء الدالة **printErrorPassword** برقم الخطأ 3. في هذه الحالة ستعيد الدالة **false**.

هذا الشرط :

```
if(CheckMinLength(passwordLength, 10))
```

يعني : إذا كان طول كلمة المرور المدخلة أقل من الطول الأدنى فقم باستدعاء الدالة **CheckLengthPassword** بالخطأ رقم 1. في هذه الحالة ستعيد الدالة **false**.

لم تبق الآن سوى حالة واحدة :

```
else return true
```

و تعني : كلمة المرور المدخلة صحيحة و توافق كافة الشروط، لذا ستعيد الدالة **true**.

2. شرح محتوى الملف **ValidateEmail.h**

في البداية، قمنا بتضمين الملف **ValidatePassword.h** مما يُتيح لنا استخدام جميع المكتبات و الدوال الموجودة في الملف.

```
#define MAX_EMAIL_LENGTH 31
```

الثابت **MAX_EMAIL_LENGTH** عبارة عن الطول الأقصى الذي يمكن أن تصل إليه اسم المستخدم، (أو لنقل الجزء الموجود قبل **@**).

```
void printErrorMessage(int numError, char charError = '\0')
```

فكرة هذه الدالة مُطابقة تماماً لفكرة الدالة **printErrorMessage**، لذا سننتقل إلى ترجمة رسائل الخطأ :

✓ **ال case الأولى و الثانية :** نفس رسائل الخطأ التي شرحناها سابقاً.

✓ **ال case الثالثة :** البريد الإلكتروني يجب أن ينتهي بـ **@yahoo.fr**

✓ **ال case الافتراضية (default) :** رقم الخطأ غير معروف.

```
void CheckLengthEmail(int numError)
```

فكرة هذه الدالة مُطابقة تماماً لفكرة **CheckLengthPassword** و رسائل الخطأ أيضاً متطابقة !، غير أن الطول الأدنى لعنوان البريد الإلكتروني هو 6 و الطول الأقصى يساوي 20.

```
bool ValidateAtDotComOrDotFr(char *str, const char *AtDotComOrDotFr)
```

هذه الدالة مسئولة عن التحقق من الجزء الموجود بعد **@**، النسخة الفرنسية يجب أن تنتهي بـ **yahoo.fr** و الإنجليزية تنتهي بـ **yahoo.com**.

تستقبل الدالة وسيطين، الوسيط الأول عبارة عن سلسلة محارف، و الثاني عبارة عن سلسلة محارف ثابتة. تقوم الدالة بفحص الحرف الذي أدخله المستخدم بعد تخزينه طبعاً في الوسيط الأول، إذا كان الحرف المدخل لا يساوي الحرف الأول من سلسلة المحارف الثابتة (أو الوسيط الثاني إن شئت) ستعيد الدالة **false**. أما إذا كان الحرف المدخل مساوي للحرف الأول من الوسيط الثاني فسيتم الانتقال إلى **الخانة الثانية** من الوسيط الأول و الثاني أيضاً، ثم تُكرر الدالة نفس العملية من جديد و هكذا، حتى تصل إلى نهاية سلسلة المحارف الثابتة، عندها سيتم فحص آخر حرف من الوسيط الأول، إذا كان هذا الأخير ينتهي بالحرف **r** أو **m** فهذا معناه أن السلسلة المدخلة (أو الوسيط الأول عن شئت) مُساوية للوسيط الثاني، و إلا فهذا معناه أن الحرف الأخير من السلسلة المدخلة لا يُساوي **m** و لا **r** أيضاً، مما يعني أن محتوى الوسيط الأول يختلف عن محتوى الوسيط الثاني.

```
bool ValidateEmail(char* name)
```

هذه هي الدالة الرئيسية التي تتحقق من صحة اسم المستخدم (الجزء الموجود قبل **@**).

في البداية تم الإعلان عن سلسلة حرفية باسم **accepted_chars** و التي تحتوي على مجموعة الرموز المُوافقة لقواعد اسم المستخدم.

المتغير **nameLength** يمثل الطول الحالي لإسم المستخدم.

```
do {  
    ...  
} while (int(*name++) != 13 & *name != '@');
```

يتم الخروج من الحلقة **do while** عندما يضغط المستخدم على **enter** أو يُدخل الحرف **@**.

تكون قيمة المتغير المنطقي **IsCtrl** مُساوية لـ **true** إذا وفقط إذا كان الرمز المدخل عبارة عن أحد الرموز الغير مرئية. أما المتغير **CheckNameError** فتكون قيمته **true** إذا كان الرمز المدخل غير مرئي أو ينتمي إلى الأحرف المحظورة في كلمة المرور.

في كل مرة، تتم قراءة الحرف الذي أدخله المستخدم ثم يُخزن هذا الحرف داخل المتغير **name**، بعد ذلك تعرض الدالة **ValidateEmail** على الشاشة إشارة نجمة.

هذا السطر :

```
if ((int) *name != 13 & CheckNameError)
```

يعني : إذا كان الرمز المدخل عبارة عن أحد الحروف المحظورة أو يمثل حرف غير مرئي باستثناء ال **enter**.

هذا الشرط :

```
if (!IsCntrl & *name != ' ' & *name != '@')
```

يعني أن الرمز المدخل ليس رمز غير مرئي و ليس مسافة بيضاء أيضا و لا يمثل محرف @ الذي يدل على نهاية اسم المستخدم. و بالتالي فالمحرف سيكون أحد الحروف المحظورة لذا قمنا باستدعاء الدالة **printErrorMessage** بوسيطين، الوسيط الأول هو الخطأ رقم 2 أما الوسيط الثاني فهو المحرف المخالف.

هذا الشرط :

```
else if (*name != '@')
```

يعني أن الرمز المدخل يختلف عن @ إذا فهو محرف غير مرئي و في هذه الحالة يتم استدعاء الدالة **printErrorMessage** بالخطأ رقم واحد.

```
*name = '\\0';
```

في كلتا الحالتين السابقتين، سيتم تخزين الرمز \0 الذي يدل على نهاية السلسلة الحرفية داخل الخانة الحالية بدلا من الحرف المخالف، ثم تعيد الدالة **false**.

```
if (CheckMaxLength(nameLength, 20))
```

في كل مرة أيضا سيتم فحص طول اسم المستخدم، إذا تجاوز عدد حروفه 20 سيتم استدعاء الدالة **CheckLengthEmail** و إظهار الرسالة المتعلقة بالخطأ رقم 2 ثم تعيد الدالة **false**.

نأتي الآن لما بعد الحلقة **do while** :

الخروج من الحلقة يعني أن المستخدم أدخل المحرف @ أو ضغط على **enter** كما يعني أيضا أن طول اسم المستخدم أقل من الطول الأقصى. بقيت الآن حالتين للتحقق :

➤ هل أدخل المستخدم الرمز **enter** ؟

➤ هل طول اسم المستخدم أقل من الطول الأدنى ؟

قبل أن ندخل في التفاصيل، هناك قضية مشتركة بين هذه الحالات وهي وضع الرمز **0** في نهاية السلسلة، و هذا ما فعلناه في هذا السطر :

```
*(name - 1) = '\0';
```

السبب في كتابة **name - 1** بدلا من **name** شرحناه سابقا.

هذا الشرط :

```
if ((int) *(name - 1) == 13)
```

يعني إذا كان الرمز المدخل هو **enter** فقم باستدعاء الدالة **printErrorMessage** بالخطأ رقم 3.

```
if(CheckSpace & CheckSpace == passwordLength - 1)
```

else تعني أن الرمز المدخل يختلف عن **enter** إذا فهو مساوي ل **@**.

هذا الشرط :

```
if (CheckMinLength(nameLength, 6))
```

يعني : إذا كان طول اسم المستخدم أقل من الطول الأدنى فقم باستدعاء الدالة **CheckLengthEmail** بالخطأ رقم 1. في هذه الحالة ستعيد الدالة **false**.

لم تبق الآن سوى حالة واحدة (إذا وصل البرنامج إلى هذه النقطة فهذا يعني أن طول اسم المستخدم موافق تماما لقواعد الإدخال) :

```
else {  
    if (Validate) {  
        ...  
    }  
    else ...  
}
```

هذه الحالة تحتوي بدورها على حالتين :

هل سيتم إدخال الجزء الموجود بعد **@** بصفة صحيحة أم لا ؟

قبل أن ندخل في التفاصيل يجب أن نُخزن اسم المستخدم في سلسلة محارف جديدة (لأن المتغير **name** مُخصص لاسم المستخدم و ليس الإيميل بالكامل) و هذا ما فعلناه في هذه الخطوة :

```
char* AtDotCom = name;
```

في هذه الخطوة :

```
bool Validate = ValidateAtDotComOrDotFr (AtDotCom, "yahoo.fr");
```

قمنا باستدعاء الدالة **ValidateAtDotComOrDotFr** بوسيطين، الأول هو السلسلة التي تحتوي على اسم المستخدم و الوسيط الثاني هو الامتداد (**إن صح التعبير**) الذي يجب أن ينتهي به البريد الإلكتروني حسب النسخة الفرنسية. (يمكننا بكل بساطة تغيير **fr** إلى **com** لكي يدعم البرنامج النسخة الإنجليزية بدلا من الفرنسية).

هذا الشرط :

```
if (Validate)
```

يعني أن إيميل المستخدم ينتهي بـ **yahoo.fr** و في هذه الحالة ستعيد الدالة **true**.

أما الحالة العكسية :

```
else printErrorMessage(3)
```

فتعني أنه حدث خطأ في كتابة **yahoo.fr** و بالتالي سيتم استدعاء الدالة **printErrorMessage** بالخطأ رقم 3 ثم تعيد الدالة **false**.