

JAN BOT

Developers

- Zakaria AL-mayah
- Yahya ALAlyi
- Ahmed AL-kabsi
- Ahmed AL-razami
- Abdulrahman Dagbas
- Mohammed Jafar
- Hamza Awiden

Under the direct
supervision of
Dr / Najla Abu Talip

مقدمة عن JAN

- "مساعدًا ذكيًا" يعمل داخل كلية أيول الجامعية. يتمثل هدف البرنامج في تقديم إجابات على أسئلة الطلاب واستفساراتهم حول أقسام الكلية ومعلوماتها من خلال واجهة تفاعلية بسيطة تعتمد على إدخال المستخدم

مميزات JAN

1. مساعد ذكي تفاعلي

يعمل البرنامج كمساعد افتراضي يمكنه التفاعل مع المستخدم عبر إدخال الأسئلة والرد بناءً على تحليل تلقائي للمحتوى.

2. تحية مخصصة حسب الوقت

يقوم البرنامج بتحليل الساعة المحلية لتقديم تحية ملائمة مثل "صباح الخير"، "مساء الخير"، إلخ.

3. تحليل ذكي للأسئلة

يفصل بين الكلمات المفتاحية (keywords) وكلمات الربط مثل (what, where, when) لاستخلاص القسم المطلوب والبحث بشكل أكثر دقة.

4. رسوم ASCII جاذبة

يعرض أسماء الأقسام (مثل INFORMATION TECHNOLOGY أو PHARMACY) باستخدام ASCII بطريقة جذابة بصريًا.

5. بحث مخصص لكل قسم

لكل قسم ملف خاص يحتوي على معلوماته. يتم البحث داخله تلقائيًا عند التعرف على القسم في السؤال.

6. خطة بديلة عند الفشل في إيجاد الجواب

في حال لم يجد البرنامج الكلمة المفتاحية في الملفات المحلية، يقوم بعرض خيار للمستخدم لفتح Google أو ChatGPT مباشرةً.

7. عرض تقدير للهيئة التعليمية

يتضمن قسمًا خاصًا لشكر الأساتذة والإدارة على دعمهم، مما يعكس الجانب التقديري للبرنامج.

8. تنظيم بصري واضح

باستخدام مربعات حدودية، وتنسيق مرتب للمخرجات في الشاشة، يسهل على المستخدم القراءة والتفاعل.

===== < يقوم المساعد بتنفيذ عدد من المهام مثل:

- التعرف على الوقت الحالي لتقديم تحية مناسبة.
- التعرف على القسم المطلوب من خلال تحليل الكلمات في السؤال.
- البحث عن كلمات مفتاحية داخل ملفات نصية مخصصة لكل قسم.
- عرض معلومات تقديرية وشكر لأساتذة ومديري الكلية.
- تقديم روابط مساعدة عند عدم توفر إجابة في الملفات المحلية) مثل ChatGPT أو Google).

. شرح الكود :

المكتبات

- `#include <iostream>`. - تستخدم لإجراء عمليات الإدخال والإخراج
- `#include <fstream>` - تُستخدم لقراءة وكتابة الملفات النصية
- `#include <ctime>` . - تُستخدم للتعامل مع الوقت والتاريخ
- `#include <conio.h>` - تُستخدم للتحكم في الإدخال من لوحة المفاتيح (مثل Windows)
- `#include <string>` - دعمًا لسلاسل الأحرف من النصوص
- `#include <cctype>` : - توفر دوال التعامل مع الحروف، مثل
لتحويل الحرف إلى حرف صغير () tolower
- `#include <Windows.h>` - تمكن البرنامج من التفاعل مع النظام مباشرة

الملفات المستخدمة داخل البرنامج

اسم الملف	وظيفته	مكان استخدامه
it.txt	يحتوي على المعلومات المتعلقة بقسم Information Technology	دالة <code>searchInFile()</code>
pharmacy.txt	يحتوي على معلومات قسم Pharmacy	<code>searchInFile()</code>
nursing.txt	يشمل بيانات قسم Nursing	<code>searchInFile()</code>
midwifery.txt	خاص بمحتوى قسم Midwifery	<code>searchInFile()</code>
business.txt	مخصص لقسم Business	<code>searchInFile()</code>
managers.txt	يحتوي معلومات حول إدارة الكلية أو المدراء.	<code>searchInFile()</code>

اسم الملف
aylolb.txt

وظيفته
searchInFile() يحتوي على معلومات عامة عن كلية Aylol.

مكان استخدامه

التقنيات المستخدمة :

التقنية	التوضيح
Strtok	لتقسيم الجملة إلى كلمات فردية .
Strcmp	للمقارنة بين الكلمة المدخلة والكلمات المستهدفة (مثل الأقسام).
find()	للبحث عن الكلمة المفتاحية داخل كل سطر من الملف.
system("start ...")	لفتح الروابط في المتصفح عند الحاجة.
ifstream	لقراءة محتوى الملفات النصية.
ctime	للحصول على الوقت الحالي وعرض التحية.
Windows.h	فقط Windows لتضمين أوامر النظام الخاصة بنظام.

أهم المميزات التحليلية في البرنامج:

- تحليل سياق السؤال بشكل يدوي (بدائي).
- تمييز الأقسام والربط بين كلمات السؤال ومحتوى الملفات.
- دعم تعدد الأقسام.
- فتح موارد خارجية في حالة عدم العثور على الإجابة.
- واجهة بسيطة وسلسلة.

* الدوال

رسالة شكر في نهاية البرنامج `greatToAylol() - 1`

وظيفتها: 🔍

عرض رسالة شكر وإمتنان لإدارة كلية "Aylol" عند انتهاء البرنامج أو خروج المستخدم.

💡 متى تُستدعى؟

عند خروج المستخدم من حلقة `question()` بكتابة "exit" أو "exite".



الوظيفة العامة:

`openHelpOptions ()` -2

- تُستخدم هذه الدالة عندما لا يتم العثور على الكلمة المفتاحية في ملفات الأقسام، فتُعطى المستخدم خيار فتح مواقع مفيدة للمساعدة) مثل ChatGPT أو Google).

```
char choice[10]; // متغير لتخزين اختيار المستخدم.
cout << ... // يطلب من المستخدم اختيار مصدر خارجي للمساعدة.
cin.getline(choice, 10); // يأخذ الإدخال من المستخدم (إما "1" أو "2").
if (strcmp(...)) // يقارن الإدخال بالسلاسل النصية لتحديد الموقع الذي سيتم فتحه.
system("start ...") // يأمر النظام بفتح الرابط في المتصفح الافتراضي.
```

متى تُستدعى هذه الدالة؟

- عند عدم العثور على كلمة مفتاحية في ملفات البيانات. (`searchInFile()`)

الفائدة:

الميزة
 بديل ذكي في حال فشل البحث
 توفير مصادر خارجية قوية

الفائدة
 لا يُترك المستخدم في حيرة
 يُحول المستخدم إلى أدوات بحث متقدمة تلقائياً

`showAppreciation()` -3

عرض رسالة شكر وتقدير لأشخاص محددين (الدكتورة نجلاء و الدكتور وليد)

متى يتم استدعاؤها؟

تُستدعى داخل دالة `question()` قبل إظهار نتيجة البحث وقبل طباعة العنوان الخاص بالقسم.

4- الدوال المسؤولة عن طباعة العناوين: (print...Title)

اسم الدالة	الوظيفة الأساسية
<code>printITTitle()</code>	طباعة عنوان قسم "Information Technology" بأسلوب ASCII Art.
<code>printPharmacyTitle()</code>	طباعة عنوان قسم "Pharmacy" باستخدام رموز تمثل الاسم بشكل فني.
<code>printNursingTitle()</code>	طباعة عنوان قسم "Nursing" بأسلوب مرئي مميز.
<code>printmidwiferyTitle()</code>	طباعة عنوان قسم "Midwifery" بخطوط ASCII.
<code>printbusinessTitle()</code>	طباعة عنوان قسم "Business" بشكل بصري لافت.
<code>printAylolTitle()</code>	طباعة عنوان اسم الكلية "AYLOL UNIVERSITY COLLEGE" بطريقة فنية جذابة.

ملاحظة

- كل ملف نصي يجب أن يُجهز مسبقاً ويوضع بجانب البرنامج التنفيذي.
- تستخدم الملفات كقاعدة بيانات للرد على الأسئلة المطروحة من المستخدم.
- كل ملف يُفترض أن يحتوي على فقرات تنتهي بعلامة " ". التي تعني نهاية الجزء المطلوب طباعته.

void toLowerCase(char* str) -5

- تعريف دالة اسمها toLowerCase.

تستقبل معامل واحد وهو:

->(char* str) مؤشر (pointer) إلى سلسلة أحرف (string) من نوع (C) أي مصفوفة من (char).

- الهدف من الدالة: تحويل جميع الأحرف في السلسلة إلى أحرف

for (int i = 0; str[i]; ++i)

- حلقة for تبدأ من الحرف الأول في السلسلة.
- تستمر بالعمل حتى تصل إلى نهاية السلسلة (أي حتى يكون str[i] == '\0').
- شرط str[i] في هذا السياق يعني " طالما لم نصل لنهاية السلسلة."

void detectDepartment(char nnn[], int &c)-6


- تعريف دالة باسم detectDepartment.
 - المعاملات:
 - nnn[]: مصفوفة من الأحرف (سلسلة نصية) تحتوي على النص الذي يحتوي على اسم القسم.
 - c: متغير من نوع int يتم تمريره بالمرجعية (reference). سيتم تعديل هذا المتغير في الدالة ليعكس الرقم المقابل للقسم.
 - ٢. **const char* nnnm[] = {"it", "pharmacy", "nursing", "midwifery", "business", "managers"};**
 - مصفوفة من السلاسل النصية، كل عنصر فيها يمثل اسمًا لقسم مختلف.
 - هذه الأقسام هي IT، الصيدلة، التمريض، التوليد، الأعمال، والإدارة.
 - الهدف من هذه المصفوفة هو مقارنة النص المدخل مع هذه الأسماء لتحديد القسم.
 - ٣. **const int mm = 6;**
 - تعيين المتغير mm الذي يحمل عدد الأقسام المتاحة في المصفوفة (6) nnnm أقسام.
 - ٤. **char* hh = strtok(nnn, " ");**
 - استخدام دالة (strtok) لتقسيم النص المدخل (في nnn) إلى أجزاء بناءً على المسافات.
 - strtok
- تقطع السلسلة النصية nnn إلى كلمات منفصلة، ويتم تخزين الكلمة الحالية في hh.

- ٥. `while (hh != nullptr) {`
- الحلقة `while` تتكرر طالما أن دالة `strtok` تعيد كلمة غير فارغة.
- `hh != nullptr` يعني أن هناك كلمة أخرى في النص للمعالجة.
- ٦. `for (int i = 0; i < mm; i++) {`
- حلقة `for` تفحص كل قسم في مصفوفة الأقسام `nnnm`.
- ٧. `if (strcmp(hh, nnnm[i]) == 0) {`
- استخدام دالة `strcmp` للمقارنة بين الكلمة التي تم استخراجها من النص (`hh`) وبين الأقسام في المصفوفة `nnnm`.
- إذا كانت الكلمة المدخلة (`hh`) تتطابق مع أي من الأقسام، سيتم تنفيذ الكود التالي.
- ٨. `c = i + 1;`
- إذا تم العثور على تطابق بين الكلمة المدخلة وأحد الأقسام، يتم تعيين قيمة المتغير `c` إلى `(i + 1)` حيث أنه فهرس القسم في المصفوفة.
- يتم إضافة 1 لأننا نريد أن يكون العدد الذي نعيده من ١ إلى ٦ وليس من ٠ إلى ٥.
- ٩. `break;`
- إذا تم العثور على القسم الصحيح، نكسر الحلقة الداخلية فوراً لأننا لا نحتاج لمزيد من الفحص.
- ١٠. `hh = strtok(nullptr, " ");`
- إعادة استدعاء `strtok` لالتقاط الكلمة التالية في النص المدخل.
- المعامل `nullptr` يعني أن `strtok` ستستمر من حيث توقفت في المرة السابقة.
- ١١. `}`
- غلق الحلقات.

— 7- `greetBasedOnTime()` . التحية حسب الوقت

 وظيفتها:

تقوم هذه الدالة بتحليل الوقت الحالي من الجهاز، ثم تعرض تحية مناسبة (صباح، ظهر، مساء، ليل) بناءً على الساعة.

 كيف تعمل:

١. تستخدم `time_t` و `localtime` للحصول على الوقت الحالي.
٢. تستخرج الساعة من الوقت (`tm_hour`).
٣. بناءً على قيم الساعة:
- بين ٥ صباحاً و ١٢ ظهراً "GOOD MORNING"
- بين ١٢ و ٥ مساءً "GOOD AFTERNOON"
- بين ٥ مساءً و ٩ مساءً "GOOD EVENING"
- باقي الأوقات "GOOD NIGH"

8-showCollegeInfo

تقوم بفتح ملف نصي (`ayl0lb.txt`) وتقرأ سطره واحداً تلو الآخر، ثم تقوم بطباعة كل سطر على الشاشة. إذا فشل فتح الملف، تُطبع رسالة خطأ على الشاشة. وفي النهاية، يتم إغلاق الملف بعد الانتهاء من القراءة.

•

الهدف الأساسي: عرض المعلومات المخزنة في ملف نصي يحتوي على بيانات تتعلق بالكلية للمستخدم

تقوم بفتح ملف نصي معين وتبحث فيه عن كلمة (أو جملة) معينة، وإذا وجدت هذه الكلمة، تقوم بعرض الأسطر التي تحتوي عليها، وإذا لم تجدها، تعرض خيارات مساعدة للمستخدم. هنا هو الشرح المجمع لكل سطر:

- أولاً، يتم تعريف متغير `search` وتخزين الكلمة التي يريد المستخدم البحث عنها في هذا المتغير:
- ```
string search;
```
- ```
search = keyword;
```
- ثم يقوم البرنامج بفتح الملف الذي تم تمريره كعامل للدالة باستخدام `ifstream`، وإذا فشل في فتح الملف، يقوم بإظهار رسالة خطأ ويعود بدون القيام بأي إجراء:
- ```
ifstream file(filename);
```
- ```
if(!file) {
```
- ```
cerr << "not open " << endl;
```
- ```
return;
```
- ```
}
```
- بعد التأكد من فتح الملف بنجاح، يبدأ البرنامج في قراءة الملف سطراً سطراً باستخدام حلقة `while` مع دالة `getline` التي تقرأ السطور من الملف:
- ```
file.seekg(0);
```
- ```
string line;
```
- ```
bool found = false;
```
- ```
while (getline(file, line)) {
```
- هنا، إذا لم يكن قد تم العثور على الكلمة بعد `found` (يساوي `false`)، فإنه يتحقق إذا كانت السطر يحتوي على الكلمة أو لا باستخدام دالة `find` الخاصة بـ `string`:
- ```
if (!found) {
```
- ```
if (line.find(search) != string::npos) {
```
- ```
found = true;
```
- ```
cout << endl;
```
- ```
} else {
```
- ```
continue;
```
- ```
}
```
- ```
}
```
- إذا وجد الكلمة في السطر، يتم تغيير متغير `found` إلى `true` ويتم إضافة سطر فارغ قبل الطباعة. إذا لم يتم العثور على الكلمة في السطر، يتم تخطي هذا السطر باستخدام `continue`.
- بعد ذلك، إذا كانت السطر يحتوي على الكلمة، يتم طباعة السطر:
- ```
if (line == ".") {
```
- ```
break;
```
- ```
} else {
```
- ```
cout << line << endl;
```
- ```
}
```
- إذا كان السطر هو نقطة ("."), يتم إنهاء عملية القراءة باستخدام `break`. أما إذا كان السطر يحتوي على نص آخر، يتم طباعة السطر باستخدام `cout`.
- بعد إتمام القراءة، إذا لم يتم العثور على الكلمة (`found`) يبقى (`false`)، يتم استدعاء دالة `openHelpOptions` لعرض خيارات المساعدة:
- ```
if (!found) {
```
- ```
openHelpOptions();
```
- ```
}
```
- وأخيراً، بعد الانتهاء من البحث في الملف وطباعة السطور (إن وُجدت)، يتم إغلاق الملف:
- ```
file.close();
```
-

١. `int b = 0;`
 - التعريف: تعريف متغير `b` من النوع `int` لتخزين القيمة التي تمثل القسم الذي يرتبط بالسؤال.
 - الهدف: تحديد القسم بناءً على الكلمات في السؤال التي تكون ذات صلة.
٢. `const char* word[] = {"what", "where", "why", "is", "are", "when", "?", "who"};`
 - التعريف: مصفوفة من السلاسل النصية التي تحتوي على الكلمات التي تعتبر استعلامات (مثل: "what", "where", "when", "who").
 - الهدف: تحديد الكلمات الاستعلامية التي يمكن أن يستخدمها المستخدم في السؤال.
٣. `const int num_word = 8;`
 - التعريف: تعريف متغير `num_word` لتحديد عدد الكلمات الاستعلامية في المصفوفة.
 - الهدف: لتحديد حجم المصفوفة `word`.
٤. `char input[250];`
 - التعريف: تعريف مصفوفة `input` لتخزين السؤال الذي يدخله المستخدم.
 - الهدف: قراءة السؤال من المستخدم.
٥. `char keyword[250];`
 - التعريف: مصفوفة `keyword` لتخزين الكلمات المهمة في السؤال (الكلمات التي ليست جزءاً من الكلمات الاستعلامية).
 - الهدف: استخراج الكلمات التي يمكن استخدامها للبحث.
٦. `char onthword[250];`
 - التعريف: مصفوفة `onthword` لتخزين الكلمات الاستعلامية (مثل: "what", "where").
 - الهدف: استخراج الكلمات الاستعلامية من السؤال.
٧. `while (true)`
 - التعريف: حلقة لتمكين المستخدم من إدخال أسئلة بشكل مستمر.
 - الهدف: السماح للمستخدم بطرح أسئلة عدة.
٨. `cin.getline(input, 250);`
 - التعريف: قراءة السؤال من المستخدم وتخزينه في المصفوفة `input`.
 - الهدف: جمع السؤال الذي أدخله المستخدم.
٩. `toLowerCase(input);`
 - التعريف: تحويل السؤال إلى أحرف صغيرة لتوحيد البيانات.
 - الهدف: ضمان عدم التأثير على المعالجة بسبب اختلاف الحالة (كبيرة أو صغيرة).
١٠. `if (strcmp(input, "exite") == 0 || strcmp(input, "exit") == 0)`
 - التعريف: التحقق إذا كان المستخدم قد أدخل كلمة "exit" أو "exite" للخروج.
 - الهدف: السماح للمستخدم بالخروج من الحلقة والدالة عند إدخال `exit`.
١١. `strtok(input, " ");`
 - التعريف: تقسيم السؤال إلى كلمات باستخدام المسافة كفاصل.
 - الهدف: التعامل مع الكلمات المفردة في السؤال.
١٢. `detectDepartment(keyword, b);`
 - التعريف: دالة `detectDepartment` تحلل الكلمات في `keyword` وتحدد القسم الذي يتعلق بالسؤال بناءً على الكلمات المدخلة.
 - الهدف: تحديد القسم المناسب بناءً على الكلمات الرئيسية في السؤال.
١٣. `system("cls");`
 - التعريف: مسح الشاشة بعد معالجة السؤال.
 - الهدف: جعل واجهة المستخدم أكثر نظافة بعد معالجة السؤال.

ملاحظة:

الجزء من الكود المتعلق بـ `detectDepartment` يقوم بتحديد القسم الذي يتعلق بالسؤال بناءً على الكلمات المدخلة، وهذا يحدد أيضًا الملف الذي سيتم البحث فيه. الدالة تستمر في التكرار حتى يقوم المستخدم بإدخال "exit".

الهدف الرئيسي للدالة

- استلام أسئلة من المستخدم.
- تحديد إذا كانت الكلمات مدخلة عبارة عن كلمات استعلامية أم لا.
- تقسيم السؤال إلى جزئين: الكلمات الاستعلامية والكلمات المهمة.
- تحديد القسم الذي يتعلق بالسؤال باستخدام الدالة `detectDepartment`.
- مسح الشاشة وإعادة التفاعل مع المستخدم بعد كل عملية.

هذه الدالة تقوم بالتحقق من القسم الذي ينتمي إليه السؤال الذي قام المستخدم بطرحه، وذلك بناءً على الكلمة المفتاحية في السؤال. بعد تحديد القسم، يتم عرض عنوان القسم المناسب، ثم البحث في الملف الخاص بهذا القسم باستخدام الكلمة المفتاحية التي أدخلت. إذا لم يكن هناك قسم معين يطابق السؤال، يتم عرض خيارات مساعدة للمستخدم.

١. عرض رسالة تقدير: باستخدام الدالة `showAppreciation()`، يتم عرض رسالة تقدير للأشخاص المعنيين.
٢. اختيار القسم بناءً على الكلمة المفتاحية: يتم التحقق من الكلمة المفتاحية المدخلة لتحديد القسم (مثل "IT" أو "Pharmacy").
٣. عرض عنوان القسم: بناءً على القسم المحدد، يتم عرض العنوان المناسب باستخدام دالة مثل `printITTitle()` أو `printPharmacyTitle()`.
٤. البحث في الملف: بعد عرض العنوان، يتم البحث في الملف المرتبط بالقسم (مثل `lit.txt` أو `pharmacy.txt`) باستخدام الكلمة المفتاحية.
٥. التعامل مع الحالات الأخرى: إذا لم يتطابق السؤال مع أي قسم، يتم عرض خيارات مساعدة للمستخدم باستخدام دالة `openHelpOptions()`.

الفكرة الرئيسية هي التعامل مع الأسئلة بشكل ذكي وتوجيه المستخدم للملفات المناسبة بناءً على القسم الذي يهتم به.

choose()_11

تهدف إلى تقديم خيار للمستخدم لتحديد ما إذا كان يريد معرفة المزيد من المعلومات حول كلية "أيلول" أو لا. بناءً على اختيار المستخدم، تقوم الدالة باتخاذ إجراء معين.

وظيفة الدالة:

١. سؤال المستخدم:
 - تعرض الدالة رسالة تسأل المستخدم إذا كان يريد معرفة المزيد عن الكلية أم لا، مع توجيه تعليمات واضحة بأن المستخدم يمكنه إدخال "yes" للموافقة أو "no" للرفض.
٢. قراءة إجابة المستخدم:
 - يتم تخزين إجابة المستخدم في متغير `bb` باستخدام دالة `cin.getline()`، والتي تقرأ النص المدخل من المستخدم.
٣. تحويل الإجابة إلى أحرف صغيرة:
 - باستخدام دالة `toLowerCase(bb)`، يتم تحويل النص المدخل (سواء كان "YES" أو "yes") إلى أحرف صغيرة لضمان أن المقارنة لا تتأثر بحالة الأحرف.
٤. تنظيف الشاشة:
 - بعد الحصول على الإجابة، يتم مسح محتويات الشاشة باستخدام الأمر `system("cls")` في بيئات `Windows`. هذا يضمن أن شاشة الكونسول تكون نظيفة للمرحلة التالية من التنفيذ.
٥. التحقق من الإجابة:

- إذا كانت الإجابة "yes" ، يتم تنفيذ الكود داخل الجملة الشرطية .if وإذا كانت الإجابة "no" أو أي شيء آخر، يتم تنفيذ الكود داخل الجملة الشرطية .else
- 6. إجراءات في حالة: "yes"
 - إذا كان المستخدم قد اختار "yes" ، يتم أولاً عرض عنوان الكلية باستخدام دالة `printAylolTitle()`.
 - ثم يتم عرض معلومات عن الكلية باستخدام دالة `showCollegeInfo()`.
 - بعد ذلك، يتم السماح للمستخدم بطرح أسئلة باستخدام دالة `question()`.
- 7. إجراءات في حالة: "no"
 - إذا كانت الإجابة "no" ، يتم مباشرة استدعاء دالة `question()` لبدء مرحلة طرح الأسئلة دون عرض أي معلومات إضافية عن الكلية.

`main()` _12

هي نقطة البداية في البرنامج حيث يتم تنفيذ العمليات الأساسية عند تشغيل البرنامج.

1. طباعة رسالة ترحيبية:

- يقوم البرنامج بطباعة رسالة ترحيب تعرف المستخدم بالشخصية "جان" الذي يمثل المساعد الخاص في كلية "أيلول". كما يوضح أنه مستعد للمساعدة ويطلب من المستخدم إخبار المساعد بما يحتاجه.

2. قراءة مدخلات المستخدم:

```
char nn[10];
cout<<"==>";
cin.getline(nn,10);
```

- يقوم البرنامج هنا بتعريف مصفوفة `nn` بحجم ١٠ لاحتواء إجابة المستخدم.
- يتم استخدام `cin.getline(nn, 10)` لقراءة سطر من المدخلات النصية من المستخدم (حتى ١٠ أحرف).

3. التحية بناءً على الوقت:

```
greetBasedOnTime();
```

- هذه الدالة تقوم بتحية المستخدم بناءً على الوقت الحالي (صباحًا، ظهرًا، مساءً، أو ليلاً). سيتم استدعاء دالة `greetBasedOnTime()` التي تتحقق من الوقت الحالي وتعرض التحية المناسبة.

4. اختيارات للمستخدم:

```
choose();
```

- هنا يتم استدعاء دالة `choose()` التي تقدّم للمستخدم خيارًا لمعرفة المزيد عن الكلية أو طرح أسئلة. بناءً على اختياره، ستتم إجراءات معينة مثل عرض معلومات عن الكلية أو الانتقال مباشرة إلى مرحلة طرح الأسئلة.

5. إرجاع القيمة:

```
return 0;
```

- في النهاية، يتم إرجاع القيمة ٠ من دالة `main()`، مما يعني أن البرنامج قد تم تنفيذه بنجاح وأنه لا يوجد أخطاء

• سير البرنامج من البداية إلى النهاية:

المرحلة	شرح ما يحدث
1 بدء التنفيذ	يبدأ تنفيذ البرنامج في <code>main()</code> . تُطبع رسالة ترحيب، ثم تُستدعى دالة <code>greetBasedOnTime()</code> لتحية المستخدم حسب الوقت.
2 اختيار المستخدم	بعدها تستدعي <code>choose()</code> لتحديد رغبة المستخدم في معرفة معلومات عن الكلية.
3 عرض معلومات الكلية	إذا اختار "yes"، تُعرض معلومات الكلية من ملف <code>aylolb.txt</code> ، باستخدام <code>showCollegeInfo()</code> .
4 وضع الأسئلة	بعد ذلك، البرنامج يدخل في حلقة الأسئلة داخل دالة <code>question()</code> . هنا يبدأ الجزء التفاعلي والتحليلي.
5 تحليل السؤال	يتم تحليل سؤال المستخدم إلى كلمات مفتاحية وأخرى مساعدة. يتم تحديد القسم المناسب عبر <code>detectDepartment()</code> .
6 البحث والرد	بعد تحديد القسم، يتم استخدام <code>searchInFile()</code> للبحث عن الإجابة داخل ملف نصي خاص بالقسم.
7 بديل في حال عدم وجود إجابة	إذا لم يتم العثور على الإجابة، يتم عرض خيار للمستخدم لاستخدام Google أو ChatGPT .
8 لخروج	المستخدم يستطيع الخروج بكتابة <code>exit</code> أو <code>exite</code> ، وهنا تُطبع رسالة شكر من <code>greatToAylol()</code> .

•