



تقرير مشروع المترجمات

بيئة تطوير لغة البرمجة العربية

الجمهورية اليمنية

جامعة إب - كلية العلوم التطبيقية

قسم علوم الحاسوب وتقنية المعلومات


مترجمات

المشرف: د/ خالد الكحسہ


الطلاب:

- أيمن محمد ناجي قحان
- حازم هزام جمال العمري
- ضياء فضل الحضرمي
- طارق فضل علي محمد العمري
- علي محمد أحمد القواس

المقدمة والملخص

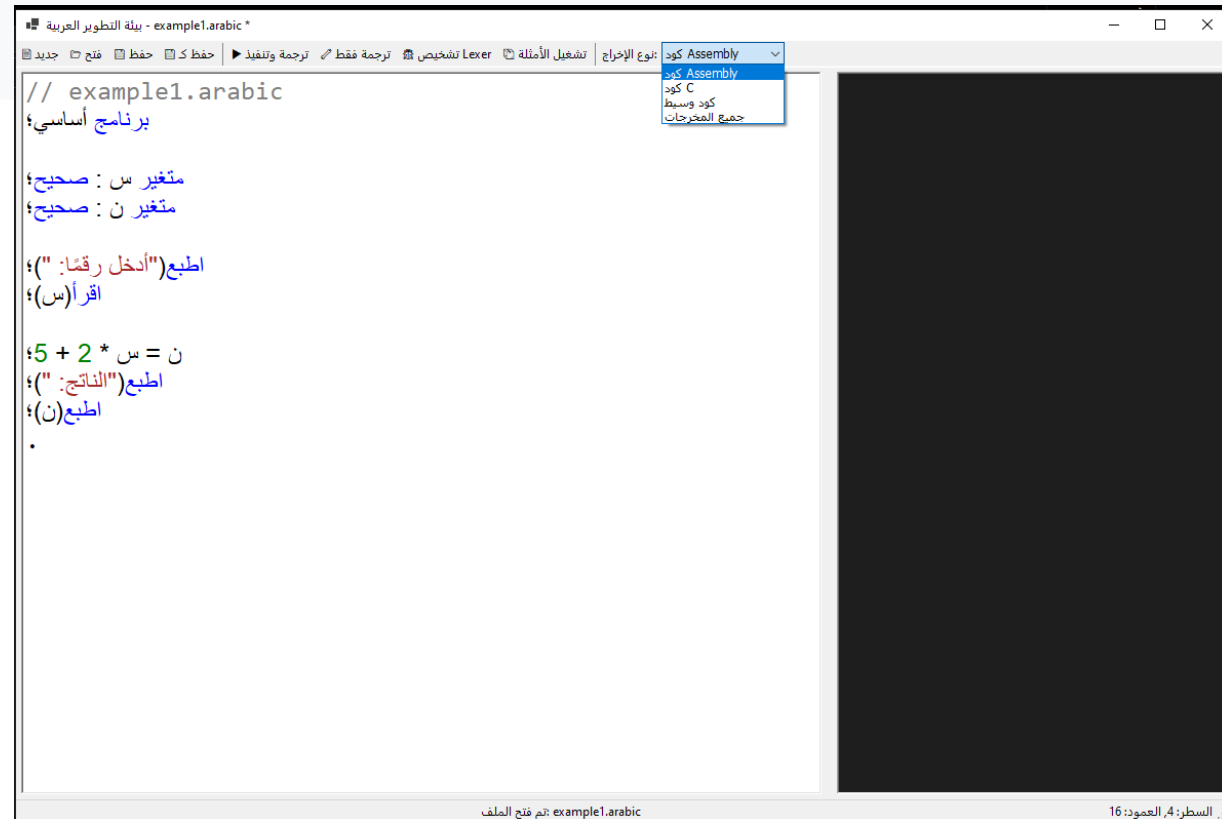
 المشروع: لغة برمجة عربية مع بيئة تطوير متكاملة (IDE) ومترجم (Compiler).

 الهدف: تمكين كتابة الكود بالعربية وتحويله إلى C ثم تنفيذ البرنامج.

 المخرجات التعليمية: فهم خط أنابيب المترجم، بناء شجرة النحو المجرد (AST)، توليد كود وسيط ونهائي، وتصميم واجهة IDE عربية.

مشروع المترجمات - 2025

جامعة إب - قسم علوم الحاسوب وتقنية المعلومات



The screenshot shows a window titled "example1.arabic - بيئة التطوير العربية". The menu bar includes options like "نوع الإخراج" (Output Type), "تشغيل الأمللة" (Run), "ترجمة فقط" (Compile Only), "ترجمة وتنفيذ" (Compile and Run), "Lexer تشخيص" (Lexical Analysis), "حفظ ك" (Save As), "حفظ" (Save), "فتح" (Open), and "جديد" (New). A dropdown menu is open under "نوع الإخراج", showing options: "كود Assembly", "كود C", "كود وسيط", and "جميع المخرجات". The code editor contains the following Arabic code:

```
// example1.arabic
برنامج أساسي؛

متغير س : صحيح؛
متغير ن : صحيح؛

اطبع ("أدخل رقمًا: ")؛
اقرأ(س)؛

ن = س * 2 + 5؛
اطبع ("النتيجة: ")؛
اطبع(ن)؛
.
```

The status bar at the bottom indicates "السطر: 4, العمود: 16" and "example1.arabic: تم فتح الملف".

مكونات المشروع

المتراجم (C++)

✓ التحليل اللغوي والنحوي للكود المصدري العربي

✓ بناء شجرة النحو المجرد (AST)

✓ توليد الكود الوسيط (IR)

✓ توليد الكود النهائي بلغة C

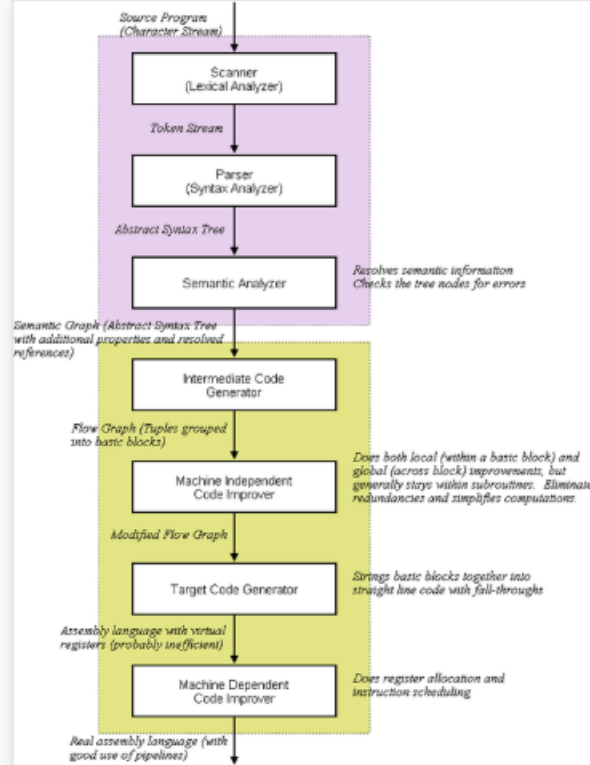
بيئة التطوير (#C)

✓ واجهة رسومية مبنية باستخدام NET 6.0 و Windows Forms

✓ محرر نصي مع دعم كامل للغة العربية

✓ تنفيذ الكود واستعراض نتائج كل مرحلة

✓ إدارة ملفات وتكامل مع المترجم والأدوات الخارجية



بيئة التطوير المتكاملة (IDE) – الوظائف

إدارة الملفات

فتح، حفظ، حفظ باسم، وإنشاء ملفات جديدة

اختصارات: Ctrl+N، Ctrl+O، Ctrl+S

التشغيل والتنفيذ

استدعاء المترجم → توليد كود C → استدعاء GCC/TCC

تشغيل البرنامج وعرض المخرجات (F5)

عرض مراحل الترجمة

التحليل المعجمي - عرض الرموز (Tokens)

التحليل النحوي - عرض الشجرة (AST)

عرض الكود الوسيط (IR) والكود النهائي (C)

```
calculator.arabic * -  ×
جديد فتح حفظ حفظ ك ترجمة فقط Lexer تشخيص تشغيل الأمللة أنواع الإخراج جميع المخرجات
// آلة حاسبة بسيطة
برنامج آلة حاسبة؛

متغير رقم1 = 15؛
متغير رقم2 = 3؛
متغير ناتج = 0؛

// عملية الجمع
ناتج = رقم1 + رقم2؛
اطبع "الجمع:"؛
اطبع ناتج؛

// عملية الطرح
ناتج = رقم1 - رقم2؛
اطبع "الطرح:"؛
اطبع ناتج؛

// عملية الضرب
ناتج = رقم1 * رقم2؛
اطبع "الضرب:"؛
اطبع ناتج؛

// عملية القسمة
ناتج = رقم1 / رقم2؛
اطبع "القسمة:"؛
اطبع ناتج؛

// عملية الباقي
ناتج = رقم1 % رقم2؛
اطبع "الباقي:"؛
اطبع ناتج؛

الترجمة تمت بنجاح
السطر: 1، العمود: 4
AM 14:12 28/10/2025 بث مباشر
```

المرحلة 1: التحليل المعجمي (Lexical Analysis)

ما هو التحليل المعجمي؟

✓ المرحلة الأولى في عملية الترجمة التي تقوم بقراءة الكود المصدري كتيار من الحروف وتقسيمه إلى وحدات أولية

✓ تحويل النص إلى رموز مميزة (Tokens) مثل: كلمات محجوزة، معرفات، ثوابت، مشغلات

✓ استبعاد التعليقات والمسافات غير الضرورية في هذه المرحلة

</> أمثلة على الرموز (Tokens)

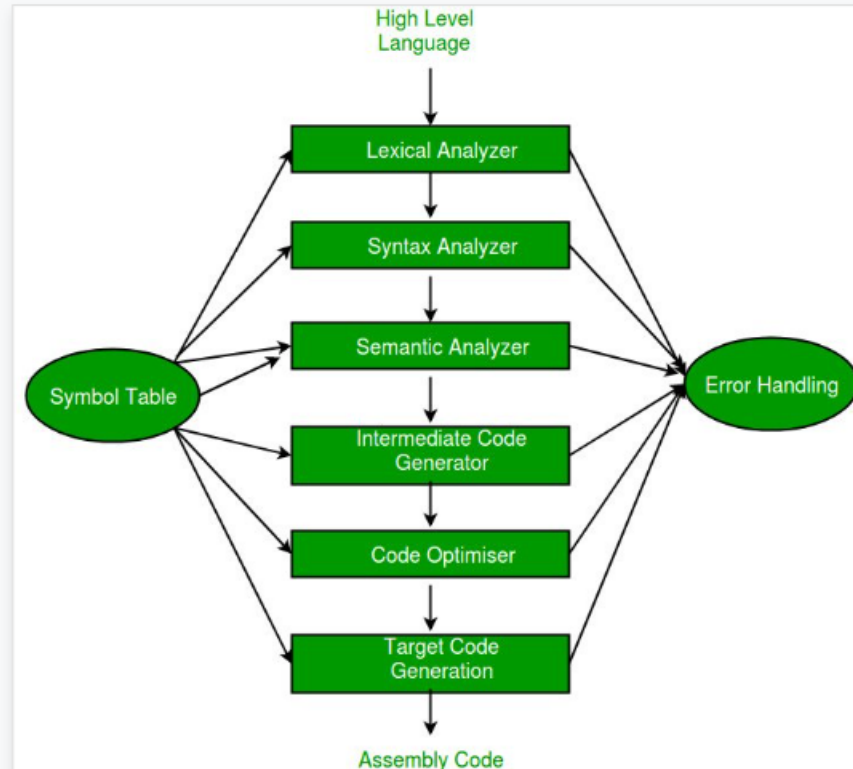
TOKEN_KEYWORD_VAR - "متغير"

TOKEN_IDENTIFIER - "س"

TOKEN_OPERATOR - "="

TOKEN_NUMBER - "10"

TOKEN_KEYWORD_IF - "إذا"



المرحلة 2: التحليل النحوي وبناء AST

التحليل النحوي (Syntax Analysis)

✓ التحقق من القواعد النحوية للغة البرمجة العربية

✓ تحويل قائمة الرموز (Tokens) إلى هيكل شجري

✓ اكتشاف الأخطاء النحوية وتقديم رسائل دقيقة

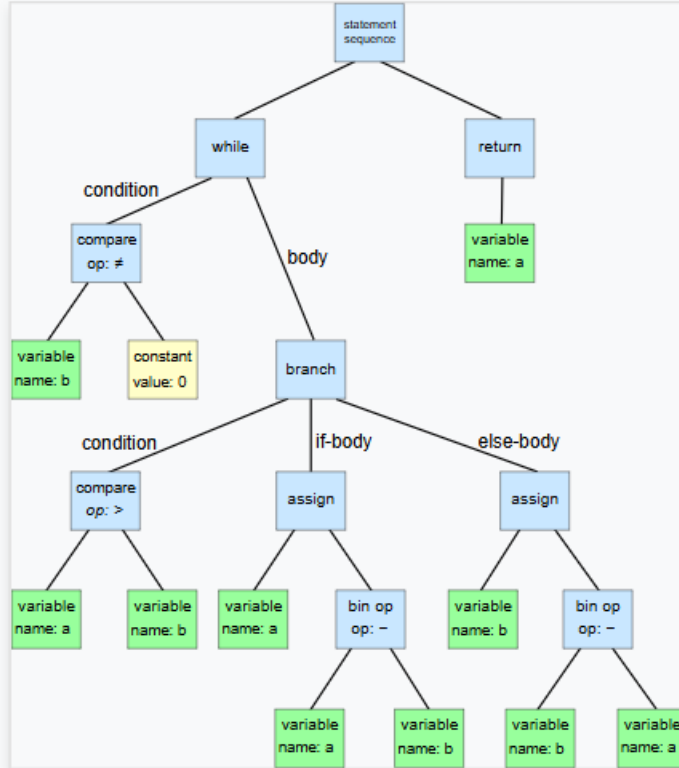
شجرة النحو المجرد (AST)

✓ تمثيل بنية البرنامج بشكل هرمي ومنطقي

✓ العقد تمثل العمليات والمتغيرات والثوابت والتعبيرات

✓ تساعد في توليد الكود الوسيط بشكل فعال

✓ تحتفظ بالتسلسل المنطقي للكود بدون تفاصيل النحو الزائدة



example1.arabic * - بيئة التطوير العربية

جديد

فتح

حفظ

حفظ ك

ترجمة وتنفيذ

ترجمة فقط

Lexar تشخيص

تشغيل الأمثلة

جميع المخرجات

نوع الإخراج

```
// example1.arabic
برنامج أساسي؛

متغير س : صحيح؛
متغير ن : صحيح؛

اطبع ("أدخل رقمًا: ")؛
اقرأ(س)؛

ن = س * 2 + 5؛
اطبع ("الناتج: ")؛
اطبع(ن)؛
```

تم توليد كود C: C:\Users\USER\AppData\Local\Temp\tmpB763_output.

تم توليد كود التجميع في C: \Users\USER\AppData\Local\Temp\tmpB763_out

تم توليد كود التجميع C: \Users\USER\AppData\Local\Temp\tmpB763_outpu

الكود الوسيط المولد:

=====

0: PRINT STRING, str_0
1: READ س
2: LOAD t0, س
3: LOAD t1, 2
4: MUL t2, t0, t1
5: LOAD t3, 5
6: ADD t4, t2, t3
7: STORE ن, t4
8: PRINT STRING, str_1
9: PRINT VARIABLE, ن
10: HALT

إحصائيات:

عدد التعليمات: 11
عدد المتغيرات المؤقتة: 5
عدد العلامات: 0
عدد السلاسل النصية: 2
عدد الرموز في جدول الرموز: 2

الترجمة تمت بنجاح

الملفات الناتجة:

C:\Users\USER\AppData\Local\Temp\tmpB763_output_intermed
C:\Users\USER\AppData\Local\Temp\tmpB763_output.c (كود C)
C:\Users\USER\AppData\Local\Temp\tmpB763_output.asm (كود A

الترجمة تمت بنجاح

السطر: 1, العمود: 1

يمثل الكود الوسيط تمثيلاً داخلياً أبسط للبرامج وأكثر استقلالية عن اللغة المصدر. سيتم استخدامه لاحقاً لتوليد الكود النهائي أو إجراء تحسينات اختيارية.

مشروع المترجمات - 2025

جامعة إب - قسم علوم الحاسوب وتقنية المعلومات

المرحلة 4: توليد الكود النهائي (C Transpilation)

</> مقتطف كود C المولد

```

X  □  -
                                     tmpB763_output.c - المفكرة
ملف  تحرير  تنسيق  عرض  تعليمات

// مولد للغة العربية C كود
// =====

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

int main() {
    int 0 = س;
    int 0 = ن;
    int t0 = 0;
    int t1 = 0;
    int t2 = 0;
    int t3 = 0;
    int t4 = 0;
    char* str_0 = "أدخل رقمًا ";
    char* str_1 = "النتيجة: ";

    printf("%s\n", str_0);
    scanf("%d", &س);
    t0 = س;
    t1 = 2;
    t2 = t0 * t1;
    t3 = 5;
    t4 = t2 + t3;
    ن = t4;
    printf("%s\n", str_1);
    printf("%d\n", ن);
    return 0;
}

<  >
UTF-8  Windows (CRLF)  110%  Ln 1, Col 1
```

يتم تحويل الكود العربي إلى لغة C القياسية، مع تحويل أسماء المتغيرات العربية (مثل "س") إلى أسماء متوافقة في C (مثل "var"). يتم استخدام مترجم C خارجي (مثل GCC أو TCC) لإنتاج الملف التنفيذي النهائي (.exe).

* example3.arabic - بيئة التطوير العربية

غيل الأمثلة Lexer تشخيص ترجمة فقط ترجمة وتنفيذ حفظ ك حفظ حفظ فتح جديد

```
// example3.arabic
```

```
برنامج حلقة؛
```

```
متغير العداد : صحيح؛
```

```
متغير المجموع : صحيح؛
```

```
المجموع = 0؛
```

```
العداد = 1؛
```

```
طالما (العداد => 10) فان
```

```
    المجموع = المجموع + العداد؛
```

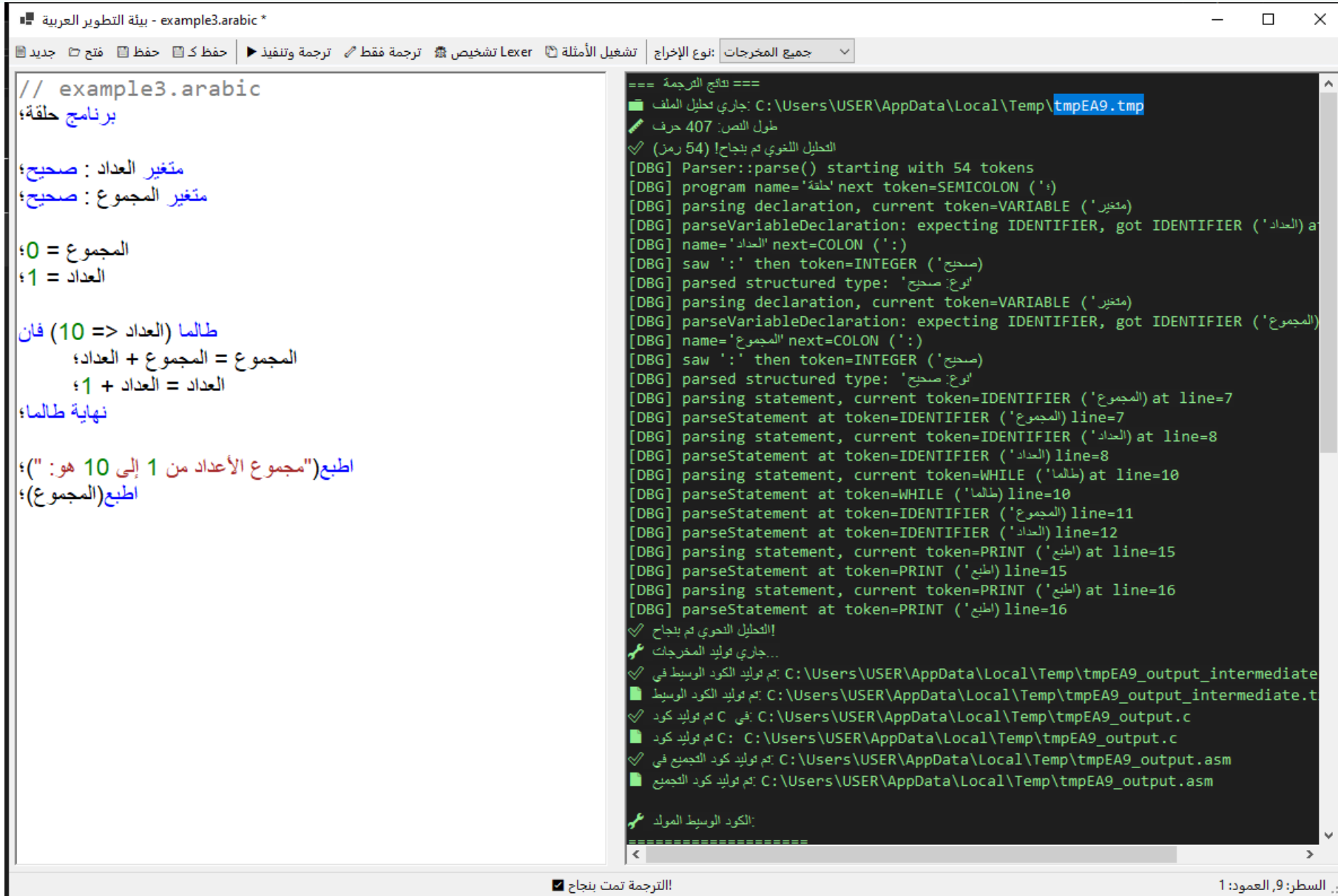
```
    العداد = العداد + 1؛
```

```
نهاية طالما؛
```

```
اطبع("مجموع الأعداد من 1 إلى 10 هو: ")؛
```

```
اطبع(المجموع)؛
```

مثال تطبيقي كامل - من المصدر إلى التنفيذ



مثال تطبيقي كامل - من المصدر إلى التنفيذ

example3.arabic * - بيئة التطوير العربية

جديد فتح حفظ حفظ ك تشغيل الأمتلة Lexer تشخيص ترجمة فقط ترجمة وتنفيذ نوع الإخراج جميع المخرجات

```
// example3.arabic
برنامج حلقة؛

متغير العداد : صحيح؛
متغير المجموع : صحيح؛

المجموع = 0؛
العداد = 1؛

طالما (العداد >= 10) فان
    المجموع = المجموع + العداد؛
    العداد = العداد + 1؛
نهاية طالما؛

اطبع ("مجموع الأعداد من 1 إلى 10 هو: ")؛
اطبع (المجموع)؛
```

الكود الوسيط المولد:

```
=====
0: LOAD t0, 0
1: STORE المجموع, t0
2: LOAD t1, 1
3: STORE العداد, t1
4: L0:
5: LOAD t2, العداد
6: LOAD t3, 10
7: LOAD t4, 0
8: CMP t2, t3
9: JLE L2
10: JMP L3
11: L2:
12: LOAD t4, 1
13: L3:
14: JZ t4, L1
15: LOAD t5, المجموع
16: LOAD t6, العداد
17: ADD t7, t5, t6
18: STORE المجموع, t7
19: LOAD t8, العداد
20: LOAD t9, 1
21: ADD t10, t8, t9
22: STORE العداد, t10
23: JMP L0
24: L1:
25: PRINT STRING, str_0
26: PRINT VARIABLE, المجموع
27: HALT
```

إحصائيات:

- عدد التعليمات: 28
- عدد المتغيرات المؤقتة: 11
- عدد العلامات: 4
- عدد السلاسل النصية: 1
- عدد الرموز في جدول الرموز: 2
- إلترجمة تمت بنجاح

الملفات الناتجة:

- (الكود الوسيط) C:\Users\USER\AppData\Local\Temp\tmpEA9_output_intermediate.txt
- C (كود) C:\Users\USER\AppData\Local\Temp\tmpEA9_output.c
- Assembly (كود) C:\Users\USER\AppData\Local\Temp\tmpEA9_output.asm

الترجمة تمت بنجاح

السطر: 9, العمود: 1

مثال تطبيقي كامل - من المصدر إلى التنفيذ

Temp

29/10/2025 4:19 PM: تاريخ التعديل:

الحجم: 1.52 كيلوبايت

tmpEA9_output.asm

C:\مستخدمون\USER\AppData\Local\Temp

29/10/2025 4:19 PM: تاريخ التعديل:

الحجم: 988 بايت

Assembler Source: النوع:

C Source File: النوع:

tmpEA9_output.c

C:\مستخدمون\USER\AppData\Local\Temp

tmpEA9_output_intermediate.txt

ملف تحرير تنسيق عرض تعليمات الكود الوسيط المولد:

tmpEA9_output.c

ملف تحرير تنسيق عرض تعليمات كود C مولد للغة العربية

tmpEA9_output_intermediate.txt

تاريخ التعديل: 29/10/2025 4:19 PM

```

=====
LOAD t0, 0 :0
t0 المجموع, STORE :1
LOAD t1, 1 :2
t1 العداد, STORE :3
:L0 :4
LOAD t2 :5
LOAD t3, 10 :6
LOAD t4, 0 :7
CMP t2, t3 :8
JLE L2 :9
JMP L3 :10
:L2 :11
LOAD t4, 1 :12
:L3 :13
JZ t4, L1 :14
LOAD t5 :15
العداد, LOAD t6 :16
ADD t7, t5, t6 :17
t7 المجموع, STORE :18
العداد, LOAD t8 :19
LOAD t9, 1 :20
ADD t10, t8, t9 :21
t10 العداد, STORE :22
JMP L0 :23
:L1 :24
PRINT STRING, str_0 :25
العداد, PRINT VARIABLE :26
HALT :27
جدول الرموز:

```

```

===== //
#include <stdio.h>
#include <stdlib.h>

} ()int main
;0 = العداد int
;0 = المجموع int
;int t0 = 0
;int t1 = 0
;int t2 = 0
;int t3 = 0
;int t4 = 0
;int t5 = 0
;int t6 = 0
;int t7 = 0
;int t8 = 0
;int t9 = 0
;int t10 = 0
;str_0 = "مجموع الأعداد من 1 إلى 10 هو:";
;t0 = 0
;t0 = المجموع
;t1 = 1
;t1 = العداد
:L0
العداد = t2
;t3 = 10
;t4 = 0

```

```

# كود تجميعي مولد للغة العربية
# =====

.data
العداد : .word 0
المجموع : .word 0
t0: .word 0
t1: .word 0
t2: .word 0
t3: .word 0
t4: .word 0
t5: .word 0
t6: .word 0
t7: .word 0
t8: .word 0
t9: .word 0
t10: .word 0
newline: .asciiz "\n"
int_format: .asciiz "%d"
str_format: .asciiz "%s"
str_0: .asciiz "مجموع الأعداد من 1 إلى 10 هو"

.text
.globl main
main:
li $t0, 0
sw $t0, t0
lw $t0, t0

```

التقنيات والأدوات المستخدمة

C++17 </> المترجم الأساسي (ArabicCompiler.exe) - معالجة الكود العربي وتحليله وتوليد الكود الوسيط والنهائي.

C#/.NET 6.0 WinForms 🖥️ بيئة التطوير المتكاملة (ArabicIDE.exe) - واجهة المستخدم الرسومية العربية.


CMake 📦 إدارة عملية بناء المترجم وتوليد ملفات المشروع للمنصات المختلفة.

GCC/TCC ⚙️ مترجم خارجي لتحويل كود C المؤلّد إلى ملف تنفيذي نهائي.

أنظمة التشغيل: Windows 🖥️ للـ IDE، مع دعم بناء المترجم على Windows/Linux/macOS.

هيكل المشروع (ملفات ومجلدات)


أخرى


CMakeLists.txt 


/output (الملفات الناتجة) 

/build (اختياري) 


/IDE

MainForm.cs 


Program.cs 


ArabicIDE.csproj 


Compiler/src


Lexer.h/.cpp 

Parser.h/.cpp 

AST.h 

Compiler.h/.cpp 

main.cpp 

 **ملاحظة:** يعتمد المشروع على بنية منظمة تفصل بوضوح بين مكونات المترجم (C++) وواجهة المستخدم (C#) مع الحفاظ على بساطة التصميم وقابلية الصيانة.

الكلمات المحجوزة والميزات اللغوية

الكلمات المحجوزة الأساسية:

الكلمة	الوظيفة	الكلمة	الوظيفة
برنامج	بداية تعريف البرنامج	اطبع	أمر الطباعة للإخراج
متغير	تعريف متغير جديد	اقرأ	أمر القراءة من الإدخال
ثابت	تعريف ثابت	طالما	حلقة التكرار while
إذا	بداية الجملة الشرطية	كرر	بداية حلقة repeat
فان	الجزء التنفيذي عند تحقق الشرط	حتى	شرط إنهاء حلقة repeat
وإلا	الجزء التنفيذي عند عدم تحقق الشرط	نهاية	نهاية الكتلة البرمجية

الخصائص اللغوية:


أسماء عربية للمتغيرات مع تحويل آمن للأسماء في كود C.


تعبيرات حسابية ومنطقية قياسية متوافقة مع المعايير العالمية.

كتل وتعليمات تحكم (if/else، loops) بصيغة عربية سهلة الاستخدام.

تحويل مباشر إلى لغة C مع الحفاظ على بنية الكود وأدائه.

أنواع المخرجات المدعومة

 **Assembly (اختياري/توضيحي):** تمثيل منخفض المستوى للبرنامج باستخدام تعليمات شبيهة بلغة التجميع MIPS، يساعد في فهم عملية الترجمة.

 **C (الكود النهائي للتنفيذ):** يتم توليد كود C كمرحلة نهائية، ويمكن تنفيذه باستخدام مترجم C مثل GCC أو TCC، مما يوفر مرونة وكفاءة في التنفيذ.

 **الكود الوسيط IR:** تمثيل مستقل عن اللغة يتوسط بين الكود المصدري والكود النهائي، يساعد في تطبيق التحسينات وتحليل البرنامج.

ملحوظات:

✓ إمكانية عرض كل نوع في نافذة مخرجات IDE مع تلوين للأخطاء والتحذيرات.

▶ دعم تنفيذ مباشر من الواجهة باستخدام F5.

النتائج والإنجازات

✓ بناء مترجم أولي: يعمل بكفاءة مع قواعد اللغة العربية المحددة في المشروع.

💻 تطوير IDE عربي: بيئة تطوير متكاملة كاملة الوظائف لعرض جميع مراحل الترجمة بواجهة سهلة الاستخدام.

</> نجاح التوليد والتنفيذ: توليد كود C بشكل صحيح والتنفيذ الناجح عبر مترجمات GCC/TCC.

✓ أمثلة عملية: تطوير نماذج توضيحية تبين الدورة الكاملة من الكود المصدري العربي إلى المخرجات النهائية.

≡ قابلية التوسع: تصميم مرن يسمح بإضافة تحسينات وتحليلات مستقبلية لتطوير اللغة.

الخلاصة والشكر

✓ **الخلاصة:** المشروع يبرهن إمكانية إنشاء منظومة تطوير عربية شاملة (IDE + Compiler) مع الترجمة إلى C، مما يسهل برمجة التطبيقات باستخدام اللغة العربية.

A **أعمال مستقبلية:** تحسين المُحلّلات، إضافة تحسينات (Optimizations)، دعم المزيد من تراكيب اللغة وإدارة المكتبات والحزم، وتطوير واجهة مستخدم أكثر تكاملاً مع أدوات التطوير الأخرى.

🏆 **الشكر والتقدير:** نتقدم بجزيل الشكر للدكتور / خالد الكحس على توجيهاته القيمة وإشرافه على المشروع، ولإدارة جامعة إب وكلية العلوم التطبيقية وقسم علوم الحاسوب وتقنية المعلومات على دعمهم المستمر، ولكل من ساهم في إنجاح هذا المشروع.