الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التعليم العالي والبحث العلمي
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
جامعة محمد بوضياف بالمسيلة
UNIVERSITE MOHAMED BOUDIAFDE M'SILA
كلية الرياضيات والإعلام الآلي
Faculté des Mathématiques et de l'Informatique
جامعة محمد بوضياف - المسيلة
Université Mohamed Boudiaf - M'sila
Département de l'informatique
قسم الإعلام الآلي

رقم: 03/ق.إ.آ/2023

المسيلة في:2023 /11/12

مستخلص من محضر اجتماع اللجنة العلمية لقسم الإعلام الآلي
رقم: 2023/06 المنعقد بتاريخ 2023/11/08 - دورة عادية

في يوم الأربعاء الثامن من شهر نوفمبر لسنة ألفين وثلاثة وعشرون، تم انعقاد اجتماع اللجنة العلمية لقسم الإعلام الآلي على الساعة التاسعة صباحا ، وكان من بين نقاط جدول الأعمال التطرق لمطبوعة الدروس للأستاذ: **صحراوي محمد** المعنونة:

Algorithmique et structure de données

وبعد الاطلاع على تقارير الخبرة الإيجابية، وافقت اللجنة على اعتماد المطبوعة واستعمالها من قبل صاحبها في حدود ما يسمح به القانون.

رفعت الجلسة في نفس اليوم على الساعة الحادية عشر صباحا.

رئيس اللجنة العلمية

رئيس اللجنة العلمية
د. دبي هشام

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université de M'sila
Faculté des Mathématiques et de l'Informatique
Département d'informatique

جامعة المسيلة
كلية الرياضيات والإعلام الآلي
قسم الإعلام الآلي

مطبوعة دروس:

الخوارزميات وهياكل البيانات 1

Algorithmique et Structures de Données 1

من انجاز الأستاذ:

صحراوي محمد

mohamed.sahraoui@univ-msila.dz

موجهة لطلبة:

السنة أولى رياضيات واعلام الي.

السداسي الاول ليسانس

جامعة المسيلة

2022/2023

تعريف بالمادة التعليمية

المادة: الخوارزميات وهياكل البيانات 1

وحدة التدريس الأساسية: UEF121 **السداسي:** 1

الميدان: رياضيات واعلام الي **المستوى:** ليسانس

الرصيد: 6 **المعامل :** 4

أهداف التدريس: تمكين الطالب من اكتساب المفاهيم الأساسية للبرمجة.

المعا رف المطلوبة مسبقا: القارئ مطالب بان يكون لديه المفاهيم الأساسية في الرياضيات ويفضل ان يكون لديه مفاهيم في الاعلام الالي

الحجم الساعي: 105 ساعة (03 ساعات دروس، 01:30 ساعة اعمال موجهة، 03 ساعات اعمال تطبيقية، 3 ساعات عمل فردي اسبوعيا).

طريقة التقييم: (60%) اختبار كتابي، (40%) تقييم مستمر

الفهرس

الفصل الأول: مقدمة

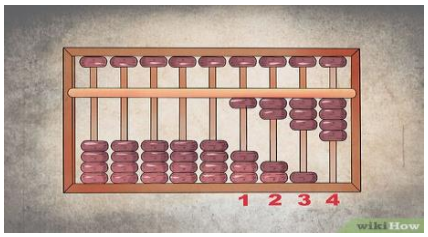
(Introduction)

1. ملخص حول تاريخ الاعلام الالي (Informatique)
2. مقدمة للخوارزميات (Algorithmes)

1. ملخص حول تاريخ الاعلام الالي (Informatique)

يعرف علم الاعلام الآلي بأنه علم المعالجة الآلية للمعلومات ويشمل كل النظريات والتقنيات التي تسمح بمعالجة المعلومة بواسطة الكمبيوتر. ويعتبر بمثابة دعم للمعرفة البشرية والاتصالات في المجالات التقنية والاقتصادية والاجتماعية.

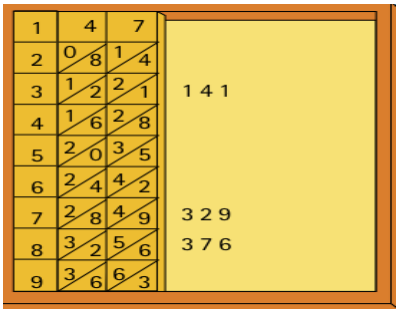
ويرتبط التطور التاريخي للإعلام الآلي بالحاجة للتطور في المعالجة الآلية لأكبر عدد ممكن من المعلومات، والتي تشهد اليوم عدة أنواع مثل الأرقام والأعداد، الأحرف والنصوص، الأشكال والصور، الأصوات والأفلام. حيث اقتصرت قديما على الحاجة لتطوير أدوات للمساعدة على الحسابات التي طالما شهدت تعقيدا متزايدا مرتبطا بالتطورات النظرية في الرياضيات.

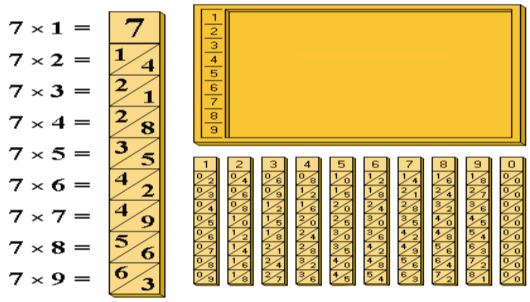


وتعتبر ماكينة المعداد (Abascus) أقدم أجهزة الحساب والتي ترجع الى 2000 سنة قبل الميلاد، وهي عبارة عن إطار مستطيل يرتبط بأجهزة الأسلاك والخرز يستخدم لعمليتي الجمع والطرح في النظام العشري. انشأت هذه الماكنة في الشرق الأوسط والصين والتي استخدمها قدماء المصريين وبلاد ما بين النهرين (العراق) ولا تزال تستخدم الى اليوم في تقوية الحساب الذهني لدى الأطفال. ويرتكز مبدأ عملها على استخدام اليدين الذي هو قديم قدم الانسان، حيث ان الاعمدة تمثل مضاعفات العشرة بينما تختلف دلالة الخرز بين خرز فوق الخط الافقي التي تمثل كل واحدة لرقم 5 وخرز تحت الخط الافقي التي تمثل الرقم 1 لكل منها. وتقوم عملية الحساب على تمثيل الأرقام أولا بتقريب الخرز الى الخط الافقي كما هو موضح في الصورة لرقم 1234، ثم بإضافة الخرز الازمة في عملية الجمع او انقاصها (ابعادها عن الخط الافقي) في عملية الطرح.

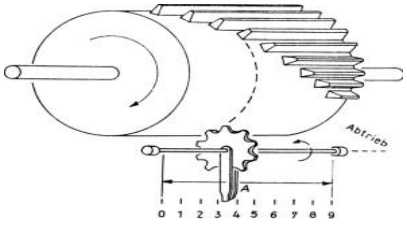
ومنذ ذلك لزم الانتظار حتى القرن السابع عشر لتظهر الآلات الميكانيكية التي تعمل بمساعدة العجلات أو الأسطوانات المسننة حيث كانت المبادئ الفيزيائية التي تم استخدامها تنتسب الى علم الميكانيك (Science de Mécanique) والذي كان يعتبر آخر ما وصل اليه العلم في ذلك الوقت.

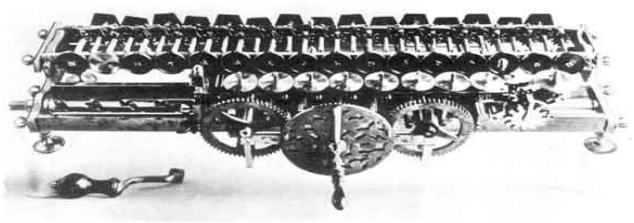
وظهرت أول آلة ميكانيكية للحساب سنة 1614 من طرف الاسكتلندي John Napie كما هي موضحة في الصورة المقابلة، والتي تعتمد على مجموعة من الاعمدة لاستخدامها في عمليات الضرب. الصورتين ادناه توضحان مبدأ عمل آلة نابيي التي تتكون من مصفوفة ضرب لأعداد النظام العشري حيث تنقسم كل نقطة التقاء الى قسمين، القسم الأعلى يمثل العدد المرفوع لضرب العددين العمودي والافقي، والقسم الأسفل يمثل العدد الباقي لنفس الضرب كما هو موضح في صورة اليمين. صورة اليسار توضح امثلة على استخدام هذا المبدئ في حساب عمليات الضرب، حيث نجد نتيجة الضرب 141 للعددين 47 و3 وكذا بالنسبة للنتيجتين 329 و376 بين 47 و7 وبين 47 و8 على التوالي.



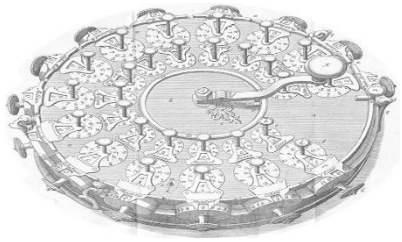


أكمل الفرنسي Blaise Pascal عددًا من هذه الآلات بحلول منتصف القرن السابع عشر، وتبعه الانجليزي Samuel Morland والألماني Gottfried Wilhelm Leibniz الذي ابتكر أسطوانة ليبنز (Tambour étagé de Leibniz) المسننة وهي أسطوانة ذات أطوال غير متساوية واستُخدمت لمدة ثلاثة قرون في أجهزة الحساب للآلات الحاسبة.





دفع انشاء العمليات الحسابية على الآلات الميكانيكية المتكررة الالماني Johann Helfrich von Muller على تصميم محرك يمكنه التعامل مع حسابات أكثر تعقيدًا والذي يهدف إلى حساب جداول اللوغاريتمات، وقام بهذا المشروع في بداية ثمانينيات القرن السابع عشر.



بعد أربعين عامًا، صمم الإنجليزي Charles Babbage آلة جديدة لا تقتصر على القيام بالعمليات الحسابية فقط بل أيضًا طباعتها. لم يستطع باباج إكمال ماكينته في حياته، لكن قام بإكمالها ابنه Henry في سبعينيات القرن التاسع عشر ونجح بجمع عدة أرقام محفوظة في بطاقات مثقبة، وهنا بدا استعمال مفهوم ذاكرة الآلة.

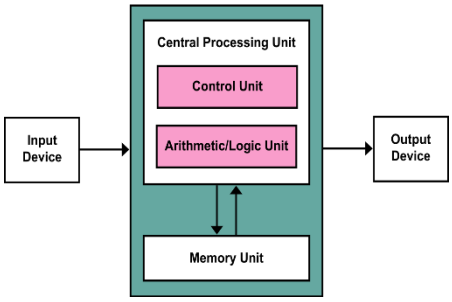
وضعت تصميمات Leibniz، Müller وBabbage الأساس بعد ذلك لإنشاء عدة آلات للحساب ولتخزين المعلومات كما أُدخلت الكهرباء لتكون الأساس لأجهزة الكمبيوتر الرقمية الحالية. ففي سنة 1885 قام الالماني Herman Hollerith بترميز المعطيات على البطاقات المثقوبة، حيث استعملت لأول مرة آلة حساب ميكانيكية للعد والترتيب تعمل بالكهرباء ليبدأ عصر الآلات الالكتروميكانيكية (ميكانيكية أوتوماتيكية) التي تستخدم نتائج علم الالكتروميكانيك (Science d' Electromécanique). حيث شهد هذا العصر عدة آلات لحل المشكلات الرياضية حتى ثلاثينيات القرن الماضي الذي شهد تطويرا هائلا وقفزة نوعية في المعالجة الآلية لتضع أسس علم الاعلام الآلي الحديث.

ففي سنة 1936 قام عالم الرياضيات الإنجليزي Alan Mathison Turing بتصميم آلة تورنج (Machine de Turing) التي وضعت الأسس الأولى للبرمجة والتي تعتبر اول آلة تفصل بين العمليات والمعالجة، فهي عبارة عن جهاز يتعامل مع مجموعة من الرموز المتسلسلة على شريط خارجي، مقسم إلى مربعات، وفق مجموعة من القواعد. حيث يقوم هذا الجهاز إما بالكتابة أو القراءة أو نقل الشريط يمينا او شمالا في كل مربع مما سمح بقراءة وتنفيذ بعض الأوامر. كما يسمح هذا الجهاز على الكشف عن إمكانية تنفيذ العمليات من عدمها والتي تم استعمالها بشكل واسع في مجال دراسة قدرة الحاسوب للعمليات وامكانية تنفيذها، وهو ما يعرف بعلم قابلية الحساب الذي يرجع

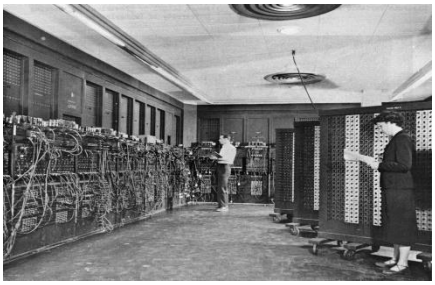
أساسا الى علم الحساب او الجبر الذي أسس له عالم الرياضيات العربي محمد بن موسى الخوارزمي الذي ينتسب اليه اسم خوارزم (Algorithme) (الذي يدور حوله محتوى هاته المطبوعة) من خلال عدة كتب ككتابه "الكتاب المختصر في حساب الجبر والمقابلة" الذي انشأه في بدايات القرن التاسع (830م).

في أواخر الثلاثينيات صمم المهندس الألماني Konrad Zuse سلسة من الآلات الحاسبة الاكتروميكانيكية. حيث تعتبر الآلة الثالثة، المسماة Z3 والمنشأة في 1941، اول آلة قابلة للبرمجة تعمل على أساس مبدأ تورينغ، حيث نفذت التعليمات بالكامل لأحد البرامج المخزنة على شريط خارجي مثقوب، مما جعلها أول كمبيوتر في العالم. وقد تم تصنيعه ب 2000 مُرحّلة

تناوب الكتروميكانيكية (Relais électromécanique)، والتي تمثل المعالج الأول الدي يرتكز في عمله على استخدام مجال مغناطيسي (Champ Magnétique) يتم إنتاجه بواسطة وشيعة (Bobine électromagnétique) عند تطبيق إشارة تحكم عليه.



في 1945 قام النمساوي John von Neumann بتصميم نموذج جديد يصف المكونات الأساسية للكمبيوتر والتي اتُبعت بعد ذلك في انشاء كل الكمبيوترات إلى غاية اليوم. يرتكز هذا التصميم على فصل المعالجة على الذاكرة وتخصيص ذاكرة داخلية لتخزين المعلومات والأوامر معا. فهي نموذج لجهاز كمبيوتر يستخدم بنية تخزين ضمنية للاحتفاظ بكل من التعليمات والبيانات المطلوبة أو الناتجة عن الحساب.



وكان اول كمبيوتر صُنع وصُمم طبقا لمخطط فون نيومان يسمى إينياك (ENIAC: Electronic Numerical Integrator And Computer) في سنة 1946 وهو أول كمبيوتر إلكتروني بالكامل قام بتمويله الجيش الأمريكي.

منذ هذه الفترة شهد التطور العلمي في مجال الاعلام الالي وتيرة متسارعة ونتائج مبهرة. حيث تم استبدال المرحلات الالكتروميكانيكية بأنابيب مفرغة في نهاية الاربعينات وتمت برمجتها عن طريق التلاعب بالقواطع الكهربائية. هذا الجيل من الحواسيب يسمى الجيل الأول ويمتد حتى منتصف الخمسينات ويتميز بضخامة أجهزة الحواسيب ولا توجد به نظام تشغيل (Operating System). كما ظهرت به أول لغة برمجة (Langage de programmation) المتعلقة مباشرة بالآلة (Langage de programmation de bas niveau) وهي لغة التجميع(Langage Assembleur) سنة 1951 ويستغرق وقتا أطول في معالجة البيانات. بدأ في هاته الحقبة الاعتماد على النظام العشري (Système décimale) كما هو الحال في كمبيوترENIAC ثم أُدخل عليه النظام الثنائي (Système binaire) في سنة 1949 من خلال جهازي EDSAC (Electronic Delay Storage Automatic Calculator) التابع لجامعة كامبريدج في المملكة المتحدة و EDVAC (Electronic Discrete Variable Automatic Computer) التابع للجيش الامريكي

بعد ذلك بدأ الجيل الثاني من الحواسيب منذ 1955 والذي امتد الى منتصف الستينات ويتميز بتكنولوجيا الترانزستور(Transistor). في هذا الجيل ظهر مفهوم نظام التشغيل (Système d'exploitation) الذي يمثل الوسيط بين الآلة والبرامج، مما تسبب في ظهور نوع جديد للغات البرمجة يدعى لغات البرمجة المنفصلة عن الالة (Langages de programmation de haut niveau). استمر الاستخدام والتطوير لأنظمة التشغيل والنوع الأخير للغات البرمجة في باقي الاجيال. تمثلت اللغات الجديدة في نوع اللغات العامة كلغة FORTRAN (mathematical FORmula TRANslating system) سنة 1957 و لغة COBOL (COmmon Business Oriented Language) سنة 1959. وكذلك البرمجة الوظيفية (Programmation fonctionnelle) التي ترتكز على حساب الوظائف الرياضية (Fonctions Mathématiques) ممثلة في لغة Lisp سنة 1958. كما تم التحسين في وقت معالجة البيانات الذي سوف يستمر في التحسن بازدياد سرعة معالجة البيانات في كل جيل.

في الجيل الثالث للكمبيوترات (1965-1980) استعملت تكنولوجيا الدوائر المتكاملة (Circuits intégrés). مما جعل أنظمة تشغيل حواسيب هذا الجيل تتميز بتكنولوجيا متعددة المهام (Multitaches) والتي تعني تنفيذ العديد من البرامج في ان واحد. كما ظهرت تقنية مشاركة الوقت (Temps partagé) وهي طريقة تسمح بمشاركة العديد من المستخدمين لنفس المعالج بتوزيع للوقت بينهم حيث أن سرعة المعالجة شهدت تقدما كبيرا إذ أصبح المعالج يستغل الوقت البسيط المستغرق في عملية إدخال أو إخراج البيانات في تولى مهمة أخرى. كما تم تطوير اول شبكة سلكية والتي تدعى ARPANET (Advanced Research Projects Agency Network) من طرف الجيش الأمريكي سنة 1967، وظهرت لغات البرمجة الحتمية (Langages de programmation impératifs) بداية من لغة Pascal سنة 1970 ثم لغة C سنة 1972 وتوالت بعد ذلك عدة لغات تنتمي الى هذا النوع والنوع السابق (Programmation fonctionnelle).

الجيل الرابع (منذ 1980) حيث التكنولوجيا المستعملة هي ميكروبروسيسور الدوائر التكاملية واسعة النطاق (Microprocesseur) التي تحتوي على آلاف الترانزستورات في الممّ2. أدت تكنولوجيا الميكروبروسيسور (المعالج الدقيق) إلى تصنيع ما يسمى بالحواسيب الصغيرة أو الشخصية (Micro ordinateur) وكذا اللوحات الرقمية (Tablets) والهواتف النقالة (Cell phones). كما عُرفت هذه المرحلة بتطوير أنظمة تشغيل رسوميه مختلفة ترتكز على استخدام النوافذ

(Windows)والفأرة (Mouse) ، وشهدت تطوير الشبكات اللاسلكية المختلفة، وكذا نوع جديد للغات البرمجة يدعى البرمجة الشيئية (Programmation orienté-objet) نتج عنه العديد من لغات البرمجة كلغة C++ سنة 1983 و Java سنة 1995 ولغات كثيرة أخرى أُستخدمت، بالإضافة الى لغات الأنواع السالفة الذكر، في مجالات مختلفة للبرمجة كمجال الواب (Programmation Web) ، مجال معالجة الصورة (Traitement d'image) ومجال برمجة الهواتف النقالة (Programmation Android).

2. مقدمة للخوارزميات (Algorithmes)

لعل القارئ للفقرة السابقة يدرك أن كلمة Algorithme ترجع في أصول تسميتها الى اسم عالم الرياضيات والفلك والجغرافيا العربي محمد الخوارزمي (780-850م) الذي كان باحثا في بيت الحكمة في بغداد، وإسمه الخوارزمي لدلالة المكان الذي ينتسب اليه وهو خوارزم الواقعة في أوزبكستان حاليا. حيث انه في سنة 825 ألف الخوارزمي كتابه "الحساب بالأرقام الهندية" الذي أسس فيه للنظام ترقيم جديد الا وهو النظام العشري باستخدام الأرقام الهندية الذي أصبح النظام المعتمد في العالم فيما بعد. تُرجم هذا الكتاب الى اللاتينية تحت عنوان "Algorithmi de numero Indorum" والذي يعني "الخوريتمي على اعداد الهنود" حيث ان كلمة الخوريتمي هي الترجمة اللاتينية لاسم الخوارزمي. تحول مصطلح الخوريتمي ليصبح معناه نظام الأرقام العشري واستُخدم بعد ذلك بصيغ مختلفة تختلف باختلاف اللغة ككلمة "Algorismo" الاسبانية في القرن الثالث عشر، لتليها الكلمتين الفرنسيتين "Augorisme" ثم "Algorisme" حتى انتهى في القرن الخامس عشر الى مصطلح "Algorithm" بالانجليزيه. وفي أواخر القرن التاسع عشر اكتسب المصطلح المعنى الذي اندرج تحته محتوى البرمجة فيما بعد، مما تسبب في ظهور نوعين أساسيين للغات البرمجة، المتعلقة و المنفصلة عن الالة كما ذُكرسابقا، واللذان استمرا في التطور الى يومنا هذا وفقا لمتطلبات كل مرحلة. ارتكز هذا التطور على النوع الثاني مما تسبب في ظهور عدد كبير من لغات البرمجة المختلفة والتي ترجع في الأساس الى مبادئ مشتركة تمخضت عنها تقنيات مشتركة. من هذا المنطلق نستنتج أن الخوارزم في الاعلام الالي يمثل الخلفية التي يرجع اليها البرنامج في تكوينه الأساسي. فهو يمثل مجموعة التقنيات الأساسية التي تخضع لشروط معينة وجب اتباعها من اجل وضع الخطوات الأساسية لبرنامج يحل مشكلة معينة. يوضع هذا البرنامج بعد اكتماله بين ايدي المستخدمين الذين لهم علاقة بهذه المشكلة ليعينهم على حلها. كمثال على ذلك تطوير خوارزم لحساب معدل الطالب للمواد التي أجرى فيها الامتحانات (البكالوريا مثلا) ليكون الاساس لإنشاء برنامج لنفس الغرض. يوضع هذا البرنامج، بعد اكمال الخطوات المتبقية، تحت تصرف الهيئة المعنية (الإدارة مثلا) ليسهل لها حساب معدلات الطلبة. انطلاقا من هذا المثال يتضح لنا العنصرين التاليين:

1- الفرق بين المبرمج الذي يطور البرنامج والمستخدم الذي يستخدمه،

2- الفصل بين الخوارزم والبرنامج بحيث ان الخوارزم يمثل أساس البرنامج، ويستلزم خطوات إضافية متعلقة بلغة البرمجة المختارة، والتي قد تختلف من لغة الى اخرى، ليصبح برنامجا قابلا للتنفيذ.

ولتوضيح الصورة كاملة لابد من إضافة العنصر الثالث الذي يوضح علاقة الكمبيوتر بالعنصرين السابقين:

3- يقتصر عمل المستخدم على البرنامج المنفذ على الكمبيوتر بينما يبدأ عمل المبرمج على الورق وينتهي على الكمبيوتر، بحيث يقوم بتحليل المشكلة وفقا للتقنيات الخوارزمية المكتسبة (والتي تمثل هدف الاعمال الموجهة لهذه المادة الدراسية) لوضع الخوارزم اللازم، ويكون ذلك على الورق، ثم يكمل العمل على الكمبيوتر بإضافة الخطوات اللازمة المتعلقة بلغة البرمجة المختارة والتي تنتهي بالنجاح في تنفيذ البرنامج (يمثل العمل على الكمبيوتر هدف الاعمال التطبيقية لهذه المادة الدراسية).

الفصل الثاني: الخوارزم المتسلسل البسيط (Algorithme séquentiel simple)

1. مفهوم اللغة ولغة الخوارزم (Notion de langage et langage algorithmique)
2. أجزاء الخوارزم (Parties d'un algorithme)
3. المعطيات: متغيرات وثوابت (Variables et constants)
4. أنواع المعطيات (Types de données)
5. العمليات الاساسية (Les opérations de base)
6. التعليمات الاساسية (Les instructions de base)
7. تكوين خوارزم بسيط (Construction d'un algorithme simple)
8. التمثيل البياني للخوارزم (L'organigramme)

1. مفهوم اللغة ولغة الخوارزم (Notion de langage et langage algorithmique)

تنص أحد أهم نظريات علم اللسانيات المتمثلة في نظرية النحو التوليدي (Théorème de grammaire générative) التي أسسها عالم اللسانيات الامريكي Avram Noam Chomsky سنة 1965 أن قواعد اللغة هي عبارة عن نظام عالمي من القواعد التي تولّد تلك المجموعات من الكلمات التي تشكل الجمل النحوية في لغة معينة. تميزت هذه النظرية عن باقي النظريات البنيوية لعلم اللسانيات في ارجاع اللغة الى مجموعة قواعد مشتركة تمثل البنية النحوية، ناتجة عن طفرة جينية في البشركونهم يولدون بقدرة فطرية على اكتساب اللغة، يتولد عن تطبيق هذه البنية النحوية مجموعات من الكلمات والجمل تختلف في شكلها من لغة طبيعية الى أخرى. من هذا المنطلق نجد ان محتوى هذه النظرية يشترك تماما مع مفهوم لغات البرمجة حيث تشترك في بنيتها الاساسية ممثلة في مجموعة التقنيات المكونة للخوارزم وتختلف في شكلها من لغة برمجة الى أخرى. فلكي نصل الى استعمال البرنامج، لا بد للمبرمج ان يمر بالمراحل التالية:

1- تحليل المشكلة: وتتمثل في دراسة وتحديد البيانات (المعطيات)، ثم معالجة هذه البيانات والوصول الى النتائج، يعقبها إيجاد طريقة ترتكز على تقنيات خوارزمية مكتسبة للانتقال من البيانات إلى النتائج. في بعض الحالات، قد تكون هناك حاجة لدراسة نظرية او يكون المشكل غير قابل للبرمجة.

2- كتابة الخوارزم: بعد ايجاد طريقة للانتقال من البيانات إلى النتائج، يشرع المبرمج في انشاء الخوارزم واضح لا لبس فيه.

3- ترجمة الخوارزم إلى لغة برمجة: كما ذكرنا سابقا، تتم الخطوات 1 و2 بدون استخدام الآلة. وإذا أردنا جعل الخوارزمية عملية، فلابد من ترجمتها إلى لغة برمجة وتحميلها على جهاز كمبيوتر. من هنا نجد أن البرنامج عبارة عن خوارزم يتم التعبير عنه بلغة برمجة معينة.

4- تنفيذ البرنامج: عند تقديم البرنامج إلى جهاز الكمبيوتر، يقوم المبرمج بمعالجته على مرحلتين:

1.4- تصحيح الأخطاء الشكلية، وهذا ما يسمى دراسة بناء الجُمل (Etude syntaxique) في مصطلحات البرمجة الذي يخضع للقواعد الشكلية للغة البرمجة المختارة.

2.4- تصحيح الأخطاء الدلالية: ويسمى دراسة دلالة البرنامج (Etude sémantique). فإذا كانت النتائج التي تم الحصول عليها هي تلك المتوقعة، هنا ينتهي تصحيح أخطاء البرنامج ويكتمل تنفيذه. أما إذا لم يتم الحصول على النتائج المرجوة، فسنستنتج أن هناك أخطاء دلالية (منطقية). فاذا كان البرنامج يعطي نتائج غير متوقعة أو نتائج جزئية أو لا يعطي نتائج أصلا، فمن الضروري إجراء مراجعة على سبيل الأولوية على صلاحية الخوارزمية أو الترجمة أو تحليل المشكل.

2. أجزاء الخوارزم (Parties d'un algorithme)

الخوارزم هو عبارة عن مجموعة محددة من التعليمات (Instructions) متسلسلة تسلسلا منطقيا لتنفيذ مجموعة من العمليات لحل مشكل من نوع معين. يتكون الخوارزم من ثلاثة أجزاء رئيسية:

1- الجزء الرأسي (Entête): يتكون هدا الجزء من الكلمة المفتاحية Algorithme التي تقابلها اسم الحوارزم المختار من طرف المبرمج.

2- جزء الاعلانات (Partie de déclaration): يحوي هذا الجزء المعطيات ممثلة في متغيرات محدودة.

3- الجسم (Le corps) : يبدأ هذا الجزء بالتعليمة début (يمثل الرمز "{" في لغة C) ثم سلسلة للتعليمات التي تعبر عن تقنيات مستخدمة تؤدي إلى تنفيذ مجموعة من العمليات لحل المشكل المطروح والوصول في الأخير الى النتائج المرجوّة وينتهي في الأخير بالتعليمة fin (يمثل الرمز "}" في لغة C). يمثل الجدول التالي المكونات الأساسية للخوارزم وكذا برنامج لغة البرمجة C .

Parties	Algorithme	Programme C
Entête	**Algorithme:** nom de l'algorithm	**include<stdio.h>** **main()** **{**
Partie de déclaration	Déclaration de variable1 Déclaration de variable2 . . .	Déclaration de variable1 Déclaration de variable2 . . .
Corps	**début** Instruction1 Instruction2 . . . **fin**	Instruction1 Instruction2 . . . **}**

يتضح جليا من الجدول ان هناك فرق في ترتيب المكونات الأساسية بين الخوارزم و برنامج C يمكن ان يتلخص في نقطتين:

1- ليس هناك تسمية لبرنامج Cلان تسميته تتم عند حفظ ملف البرنامج،

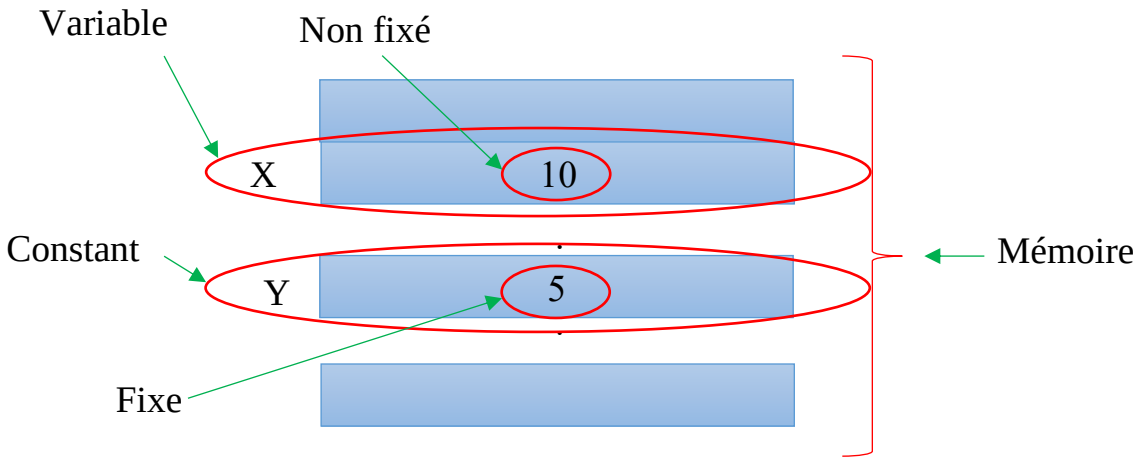
2- يتكون الجزء الرأسي من استدعاء لملف المكتبة stdio.h التي تحوي التعليمات الأساسية للبرمجة والتعليمة ()main اللازمة لبدئ البرنامج مع إشارة البدئ ("{"). وذلك لان جزء الاعلانات يكون داخل المحتوى بخلاف الخوارزم. يمكن استدعاء مكتبات أخرى عند الضرورة.

3. المعطيات: متغيرات وثوابت (Variables et constants)

المتغير (Variable) عبارة عن مكان محدد في الذاكرة (Mémoire) الداخلية لجهاز الكمبيوتر يكفي لتحميل قيمة واحدة من نوع معين. يُعرّف المتغير بنوعه الذي ينتمي الى أحد الأنواع المحددة في الفقرة الرابعة (الموالية)، واسمه الفريد في البرنامج الذي ينتمي اليه. يُحدّد اسم المتغير ونوعه من طرف المبرمج في جزء الاعلانات ويتعرف البرنامج على مكان هذا المتغير من خلال اسمه حيث يستطيع الدخول إلى محتوى هذا المكان في كل مرة (وفقا للتعليمات المخولة بذلك) لكتابة قيمة من نفس النوع فيه أو لقراءة ما فيه من قيمة ان لم يكن فارغا.

ويُعرّف الثابت (Constant) بكونه Variable من نوع خاص، تكمن خصوصيته في كونه يبقى ثابتا على قيمة واحدة لا تتغير بخلاف الVariable الذي يمكن ان يغير قيمته كلما استدعت الضرورة.

تسمى هاته القيمة بالثابت (Constant). توضح الصورة الموالية الVariable X و الConstant Y ذوو النوع صحيح (Entier) وقيمتهما في الذاكرة 10 و 5 على التوالي.



الجدول التالي يوضح طريقة الاعلان للاثنين في الخولرزم و كذا في برنامج C.

Type	Algorithme	Programme C
Variable	**Variable** Nom_variable: Type	Type Nom_variable**;**
Constant	**Constant** Nom_variable ← valeur	**Const** Type Nom_variable = valeur **;**

تخضع التسمية في البرمجة سواء كانت للVariable أو للConstant أوحتى للخوارزم الى الشروط التالية:

- لا بد ان يبدأ اسم المتغير بحرف (Lettre) ،
- يتكون اسم المتغير على الأقل من حرف (Lettre) ،

- بعد حرف البداية، لا يمكن ان يحتوي إلا على حروف او اعداد او رمز واحد الا وهو رمز الشرطة سفلية "_" (Underscore) لربط أجزاء الاسم،
- لا يسمح باستخدام الرموز الأخرى مثل (+ او/ او* او ، ...الخ) ولا الحروف ذات الشرطات (Accents) مثل é او à او ê.

أمثلة:

Type	Algorithme	Programme C
Variables	**Variable** X: entier **Variable** val1: entier **Variable** m2_max: entier	**int** X**;** **int** val1**;** **int** m2_max**;**
Constant	**Constant** PI ← 3.14	**Const float** PI= 3.14**;**

ملاحظة:

يمكن جمع المتغيرات ذات النوع الواحد فيصبح الاعلان السابق كالتالي:

Type	Algorithme	Programme C
Variables	**Variables** X, val1, m2_max: entier	**int** X,val1**,** m2_max **;**

لا بد للقارئ ان يركز على ال كلمات المفتاحية (mots clés) المكتوبة بالخط العريض وكذا أماكن الفواصل (Virgule) والفواصل المنقوطة (Point virgule) وذلك لأهميتها البالغة خاصة في برنامج C.

4. أنواع المعطيات (Types de données)

يمثل الجدول التالي حصر لأنواع المعطيات الأساسية التي يمكن استخدامها في البرمجة، مع الصيغة المخصصة لها في كل من الخوارزم وبرنامج C:

Types de données	Algorithme	Programme C
الاعداد الصحيحة	**entier**	**int**
الاعداد الحقيقية	**réel**	**float**
حرفي (يمثل رمز واحد ينتمي لكل ما يمكن إدخاله من لوحة المفاتيح)	**caractère**	**char**
نصي (سلسلة حروف أو سلسلة من الرموز محددة بالعدد الذي تأخذه فيمة size)	**chaine**[**size**]	**char**[size]
منطقي (يأخذ إحدى قيمتين true أو false)	**booléen**	**boolean**

مثال:

Algorithme	Programme C
Variables: A**: entier** b**: réel** flag1,flag2**: caractère** student_name**: chaine[**25**]** Inf**: booléen**	 **int** A**;** **real** b**;** **char** flag1,flag2**;** **char[**25**]** student_name**;** **boolean** Inf**;**

5. العمليات الاساسية (Les operations de base)

يمثل الجدول التالي حصر لأنواع العمليات الاساسية التي يمكن استخدامها في البرمجة، مع الكلمات المفتاحية التابعة لها في كل من الخوارزم و برنامج C وكذلك الامثلة:

Type d' opération	Algorithme	Programme C	Exemples
العمليات الحسابية Operations arithmétiques	+ - * / **Div** **Mod**	+ - * / **%**	5+7=12 7-5=2 3*4=12 9/2=4.5 10 Div 3=3 10 Mod 3=1
المقارنة Comparaison	= ≠ < > ≤ ≥	= = **!=** < > <= >=	5=4(faux), 'ah'='ahc'(faux) 3≠6 (vrai), 'ca' ≠ 'cb' (vrai) 2<2 (faux), 'a' < 'd' (vrai) 8>5 (vrai), 'b' > 'x' (faux) 4≤4 (vrai), 'e' ≤ 'y' (vrai) 9≥10 (faux), 'b' ≥ 'c' (faux)
العمليات المنطقية Operations logiques	**et** **ou** **Non** **Xor**	**&&** || **!** ^	
الجمع بين النصوص (Concaténation de chaines)	+	+	'good' + ' ' + 'moorning' = 'good moorning'

بعد الاطلاع على الجدول لابد من توضيح النقاط التالية:

1- كل العمليات المذكورة تستلزم وجود طرفين الا عمليتين: العملية المنطقية (Non) التي تستلزم طرفا واحدا، وعملية الجمع بين النصوص التي تستلزم طرفين فما أكثر، بالإضافة الى ان كل أطراف العملية لا بد أن تكون من نفس النوع. قد يكون كل طرف بسيط كالأطراف الموجودة في الامثلة، كما قد يكون مركبا كالمقارنة التالية:

(((5+3)*2)≤(((2/4)*3)Div 3) والتي تنتهي الى الإجابة faux

2- لا تُستعمل العمليات الحسابية إلا بين طرفين entier أو réel. يستثنى من ذلك عمليتي Div و Mod بين طرفين réel والّتان تعنيان حاصل القسمة الصحيح وباقي القسمة الصحيح على التوالي. كما يلاحظ ان عملية Div لا يقابلها شيء في برنامج C وذلك لانه ليس لديها كلمة مفتاحية خاصة وانما تُنجز بطريقة غير مباشرة سنشرحها في الفقرة 6.1.

3- تُستعمل عمليات المقارنة بين طرفين كل الانواع. يستثنى من ذلك العمليات (<,>,≥,≤) بين طرفين booléens.

4- لا تُستعمل العمليات المنطقية إلا بين طرفين booléens ولضرب امثلة على هذه النوع من العمليات لا بد من استخدام الجدول المنطقي (جدول الحقيقة) التالي لطرفين Booléens A و B يأخذان كل القيم المنطقية الممكنة لنرى نتيجة العمليات المنطقية السالفة الذكر.

A	B	Non A	Non B	A et B	A ou B	A Xor B
false	false	true	true	false	false	false
false	true	true	false	false	true	true
true	false	false	true	false	true	true
true	true	false	false	true	true	false

بالإضافة الى الجدول، يوجد قانون Morgan والذي كثيرا ما يستخدم في البرمجة والذي ينص على الاتي: Non (A et B) ⇔ (Non A) ou (Non B)

Non (A ou B) ⇔ (Non A) et (Non B)

6. التعليمات الاساسية (Les instructions de base)

تنقسم التعليمات الأساسية الى نوعين: إسناد القيمة (Affectation) وتعليمات الادخال والاخراج (instructions d'entrée sortie).

6.1 إسناد القيمة (Affectation)

لاحظنا في الفقرة الثالثة ان القيم المباشرة توضع في الذاكرة ولا سبيل للوصول اليها الا عن طريق الVariables أو الConstants، لذلك فان الاحتفاظ بأي قيمة مباشرة لغرض ما، كاستخدامها في عملية مّا مثلا، لا تتم الا عن طريق استخدام الVariables و/أو الConstants من نفس النوع. بالإضافة الى ذلك، توضع النتائج، التي يُراد حفضها لغرض ما كاستخدامها مرة أخرى مثلا، في Variables لنفس النوع. وتتم هذه الطريقة عن طريق عملية الإسناد التي يعبر عنها باستخدام الرمز (←) في الخوارزم والرمز (=) في برنامج C (الرمز = لا يرمز لعلامة يساوي في برنامج C كما هو موضح في الفقرة الخامسة). فإسناد القيم يتم عبر الطرق الثلاث التالية:

1- اسناد قيمة مباشرة: وتستخدم للVariables و الConstants الذين هم من نفس القيمة، كما يمكن عمل ذلك في جزء الاعلانات أو في الجسم. مع وجود فرق بين الVariable و الConstant يكمن في ان الVariable يمكن اسناد اليه عدة قيم متتالية حسب الضرورة (القيمة الجديدة تأخذ مكان القيمة التي قبلها لان المكان في الذاكرة لا يسع الا لقيمة واحدة)، بينما الConstant لا يمكن اسناد اليه الا قيمة واحدة ولمرة واحدة.

أمثلة:

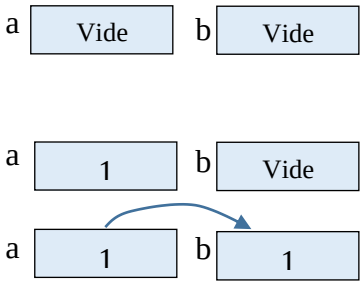
الحالة	Algorithme	Programme C	Contenants
اسناد قيمة لVariable في جزء الاعلانات	**Variables:** a ← 1: **entier** **…**	**include<stdio.h>** **main()** **{** **int** a=1; **…** **}**	a 1
اسناد قيمة لConstant في جزء الاعلانات	**Variables:** **constant** PI ← 3.14 **…**	**include<stdio.h>** **main()** **{** **const float** PI=3.14; **…** **}**	PI 3.14
اسkاد قيمة لVariable في الجسم	**Variables:** a: **entier** **début** a ← 1 **fin**	**include<stdio.h>** **main()** **{** **int** a; a=1; **}**	a Vide a 1
اسkاد قيمة لConstant في الجسم	**Variables:** **constant** PI **début** PI ← 3.14 **fin**	**include<stdio.h>** **main()** **{** **cons float** PI; PI=3.14; **}**	PI Vide PI 3.14

2- اسناد قيمة من Variable الى Variable: ولا يتم هذا الا بين الVariables، كما لا يمكن ان يكون ذلك الا في الجسم. حيث يتم عن طريق هذه العملية نسخ القيمة المباشرة الموجودة في Variable االعطاء (الموجود على اليمين) ووضعها في Variable الاستقبال (الموجود على

اليسار). وهنا لا بد من الإشارة الى أمر مهم الا وهو ان Variable العطاء (أو أي Variable سنأخذ منه قيمة ثابتة) لا بد ان لا يكون فارغا أي بمعنى أُسندت اليه قيمة ثابتة من قبل، وتسمى هذه العملية بإعطاء القيمة البدئية (Initialisation) (في المثال: Variable العطاء "a" أخذ القيمة 1 قبل ان يستخدم في عملية الاسناد كمعطي للقيمة التي يمتلكها والتي هي 1 لـVariable الاستقبال "b" لكي يصبح هو أيضا ممتلك لنفس القيمة 1).

مثال:

Algorithme	Programme C	Contenants

Algorithme exemple_1
Variables:
a,b: **entier**
 	
début
 	
a ← 1
 	
b ← a
 	
fin
	include<stdio.h>
	
	main()
	{
	int a,b ;
	
	a=1;
	
	b=a;
	
	}
		

3- اسناد نتيجة عملية الى Variable: ولا يتم هذا الا باستخدام Variables فقط او باستخدام لـVariable مع قيم ثابتة. كما لا يمكن ان يكون ذلك الا في الجسم. حيث يتم هنا حساب العملية أولا ثم يتم وضع نتيجتها في Variable الاستقبال (الموجود على اليسار). في المثال قمنا بعمليتي قسمة تعطيان نفس النتيجة: الأولى باستخدام Variable العطاء "x" مع قيمة ثابتة 5 ووضعت النتيجة في الـVariable "y"، والثانية باستخدام لمتغيرين "x" و "b" ووضعت النتيجة في الـVariable "z"، بعد ان قمنا بإعطاء القيمة البدئية لكلتا المتغيرين ("x" في جزء الاعلانات باعطائها القيمة 2 و "b" في الجسم باعطائها القيمة 5). كما يجدر لفت الانتباه الى نوع Variables الاستقبال "y" و "z" اللذان كانا من نوع réel وذلك لان حاصل القسمة قد يكون عشريا (بالفاصلة). فلو كانا من نوع entier لكان حاصل القسمة الجزء الصحيح فقط أي مايقابل عملية Div في الخوارزم.

مثال:

Algorithme	Programme C	Contenants

Algorithme exemple_2
Variables:
x ← 2,b: **entier**
y,z: real
…
début
y ← 5/x
 	
b ← 5
 	
z ← b/x
 	
fin
	include<stdio.h>
	main()
	{
	int x=2,b ;
	float y,z;
	
	y=5/x;
	
	b=5;
	
	z= b/x;
	
	}
		x 2 b Vide
		y Vide z Vide
		
		x 2 b Vide
		y 2.5 z Vide
		
		x 2 b 5
		y 2.5 z Vide
		
		x 2 b 5
		y 2.5 z 2.5

4- لإسناد قيم مباشرة لمتغير من نوع Caractère أو chaine لابد من استخدام علامة خاصة ممثلة في مزدوجة (' ') ووضع القيمة التي هي من نفس النوع داخلها (بالنسبة لبرنامج C تُستبدل المزدوجة بمزدوجتين (" ") في نوع chaine). بالإضافة الى ذلك، فان كل ما قيل في النقطتين 1 و 2 ينطبق على هذين النوعين من البيانات، بينما النقطة الثالثة لا تتم الا بين عمليات المقارنة التي توضع في متغير من نوع booléen أو عملية الConcaténation التي توضع في chaine.

مثال:

Algorithme	Programme C	Contenants

Algorithme exemple_3
Variables:
p ← 'm',n1: **caractère**
result: **Booléen**
word:**chaine[3]**
 	
début
 	
result ← (p='a')
 	
n1 ← 'e'
 	
word ← p+n1
fin
	include<stdio.h>
	main()
	{
	char p='m',n1;
	boolean result;
	char[2] word;
	
	result= (p = = 'a');
	
	n1='e';
	
	word= p+n1;
	
	}
		p m n1 Vide
		result Vide word Vide
		
		p m n1 Vide
		result faux word Vide
		
		p m n1 e
		result faux word Vide
		
		p m n1 e
		result faux word me

6.1. تعليمات الادخال والاخراج (Les instructions d'entrée sortie)

المقصود بقراءة المدخلات (Les entrées) هو كيفية إدخال القيم المكتوبة من طرف المستخدم إلى البرنامج لكي يتعامل معها، كأن نكون بصدد القيام بعملية الجمع ونطلب منه أن يدخل رقمين، فعملية الإدخال هذه تحتاج إلى قراءة قيم مختلفة من طرف البرنامج في كل تنفيذ له ليستخدمها كمدخلات لتنفيذ العملية، وتكون عملية قراءة المدخلات عبراستخدام التعليمة "lire(variable)" في الخوارزم ويتم كتابة الvariable الذي سيستقبل القيمة مباشرة. كما تستخدم التعليمة "scanf("%x",&variable)" في برنامج C حيث يضاف الى الvariable علامة &، ويُستخدم الرمز "%x" للدلالة على نوع البيانات، فالحرف x يختلف باختلاف نوع البيانات المراد إدخالها حيث نضع مكانه حرف d حينما نريد ادخال entier أو حرف f حينما نريد ادخال réel أو حرف c حينما نريد ادخال caratctère أوحرف s حينما نريد ادخال chaine. في المثال الموالي يرمز # الى Vide (فارغ).

مثال:

Algorithme	Programme C	قبل التنفيذ			بعد التنفيذ		
		a	b	c	a	b	c
Algorithme exple_4 **Variables:** a,b,c: **entier** **début** **read**(a) **read**(b) c ← a+b **fin**	**include<stdio.h>** **main()** **{** **int** a,b,c ; **scanf**("%d",&a); **scanf**("%d",&b); c= a+b; **}**	#	#	#	#	#	#
		#	#	#	**2**	#	#
		#	#	#	**2**	**5**	#
		#	#	#	**2**	**5**	**7**
					نفرض ان المستخدم اختار a=2 و b=5		

يقصد بإخراج البيانات (Les sortie) عملية طباعة نص معين أو قيمة variable على شاشة المستخدم ليراها، ففي الخوارزم نستخدم الامر (التعليمة) "ecrire(content)" ونفتح القوسين، ثم نضع ال variable المراد إظهارقيمته للمستخدم كما نستطيع إضافة نص او اكثر بين مزدوجة لإظهاره للمستخدم. بينما في برنامج C نستخدم الامر "printf("phrase+%x",variables)" حيث ان phrase تدل على النص المراد إخراجه حرفيا بينما %x تدل على نوع البيانات التي تختلف باختلاف نوع البيانات المراد إخراجها، بنفس الطريقة التي ذكرت سابقا في المدخلات ، والتي تأخذ قيمتها من الvaraiables الموجودة في الاخيربعد المزدوجة الاخيرة والفاصلة.

مثال:

Algorithme	Programme C	على الشاشة بعد التنفيذ
Algorithme exple_5 **Variables:** a,b,c: **entier** **début** **ecrire**('donner a:') **lire**(a) **ecrire**('donner b:') **lire**(b) c ← a+b **ecrire**('L'addition est:',c) **end**	**include<stdio.h>** **main()** **{** **int** a,b,c ; **printf**("donner a:"); **scanf**("%d",&a); **printf**("donner b:"); **scanf**("%d",&b); c= a+b; **printf**("L'addition est:%d",c); **}**	donner a: يتم ادخال رقم من طرف المستخدم 3 مثلا donner b: يتم ادخال رقم من طرف المستخدم 7 مثلا L'addition est:10

7. تكوين خوارزم بسيط (Construction d'un algorithme simple)

في هذه الفقرة سننشئ خوارزم بسيط يهدف لحساب عمليتي الجمع والطرح لأي عددين صحيحين. للقيام بذلك، سنتبع الخطوات الثلاث الأولى من بين الاربع المذكورة والمشروحة في الفقرة الاولى، كون الخطوة الرابعة تستلزم العمل على جهاز الكمبيوتر.

1. تحليل المشكلة:

المعطيات: أيّ عددين صحيحين، اذن لا يوجد تحديد لعددين محددين مما يستدعي قراءة المدخلات باستخدام الامر "lire(variable)". وبما أنه يوجد عددين إذن لابد من استخدام متغيرين اثنين مختلفين ولكل واحد قراءة المُدخل الخاص به.

المعالجة: تجرى على هذين العددين عمليتي الجمع والطرح وهما عمليتين متوفرتين في الخوارزم كماهو مذكور في الفقرة الخامسة (الجدول)

المخرجات: نتيجتين، الاولى لعملية الجمع والثانية لعملية الطرح. توضع كل نتيجة في متغير مختلف ثم نقوم بإخراج محتوى كل متغير على الشاشة باستخدام التعليمة "ecrire(content)" لكل متغير.

2. كتابة الخوارزم:

Algorithme exemple_6
Variables:
a,b,c,d: **entier**
début
eccrire('donner a:'**)**
lire(a**)**
ecrire('donner b:'**)**

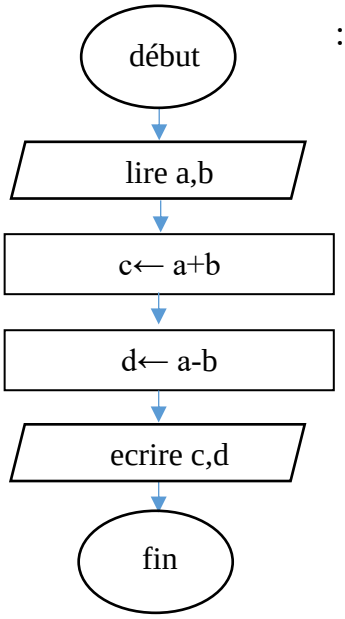
```
lire(b)
c ← a + b
d ← a - b
ecrire('L'addition est: ' ,c)
ecrire('La soustraction est: ' ,d)
fin
```

3. ترجمة ال خولرزم إلى لغة برنامج C:

```
include<stdio.h>
main()
{
int a,b,c,d ;
printf("donner a:");
scanf("%d",&a);
printf("donner b:");
scanf("%d",&b);
c= a + b;
d= a - b;
printf("L'addition est: %d",c);
printf("La soustraction est: %d",d);
}
```

8. التمثيل البياني للخوارزم (L'organigramme)

هو تمثيل مصور للخوارزمية يوضح خطوات حل المشكلة من البداية إلى النهاية مع إخفاء التفاصيل لإعطاء الصورة العامة للحل وتسمى كذلك بالخريطة الانسيابية (Organigramme). تُوضع في هذا التمثيل الdébut و الfin في دائرتين ثم يتم توضيح المدخلات و المخرجات في شكل متوازي اضلاع، بينما توضع العمليات المنجزة في أُطر. كمثال على ذلك، التمثيل البياني للخوارزم البسيط للفقرة السابقة والذي يكون كالتالي:



الفصل الثالث: الهياكل الشرطية

(Strutures conditionnelles)

1. مقدمة (Intrduction)
2. الهيكل الشرطي البسيط (Struture conditionnelle simple)
3. الهيكل الشرطي المتعدد الاختيارات (Struture conditionnelle de choix multiple)
4. تعليمات الانتقال (Branchement)

1. مقدمة (Intrduction)

في كثير من الأحيان يتطلب الخوارزم إجراء تحقق من وجود شرط معين خلال عملية التحليل، فإن كانت نتيجة التحقق ايجابية ينفذ الخوارزم عملية أو عمليات معينة، وإن كان العكس ينفذ عملية أو عمليات أخرى، وهذا مايسمى بالهياكل الشرطية (Strutures conditionnelles). على سبيل المثال، نريد إنشاء برنامج يتكون من نافذة لتسجيل الدخول مثل تلك التي في برنامج Skype أو برنامج Messenger، فنحن ملزمين بالتحقق من اسم المستخدم وكلمة السر المدخلتين، إن كانا صحيحين سمحنا بعملية الدخول، وإن كان أحدهما خاطئا أظهرنا رسالة خطأ.

2. الهيكل الشرطي البسيط (Struture conditionnelle simple)

يسمح الهيكل الشرطي البسيط بتنفيذ تعليمة أو أكثر إذا تحقق شرط معين، ويقوم بتنفيذ تعليمة أُخرى أو أكثر إذا تحقق عكس ذلك الشرط. لابد من الإشارة إلى ان استخذام عكس الشرط لتنفيذ تعليمة أُخرى أو أكثر يبقى اختياريا، أي بمعنى يمكن الاستغناء عنه اذا لم يكن هناك داع لاستخدامه. تكون بنية الهيكل الشرطي البسيط كالاتي:

Algorithme	Programme C

si condition **alors**
liste des instructions 1
sinon
liste des instructions 2
fin si
	if (condition)
	{
	instruction list1
	}
	else
	{
	instruction list2
	}

حيث ان الcondition يمثل عبارة منطقية تنتهي الى الإجابة ب vrai أو faux وتتكون من عمليات منطقية و/أو عمليات المقارنة السابقة الذكر في الفصل السابق. كما تكون مجموعتي ال liste des instructions 1 و ال2 liste des instructions مختلفتين وتتكون كل واحدة منهما على الأقل من تعليمة واحدة وتكون متسلسلة اذا كانت اكثر من واحدة.

مثال:

في هذا المثال سيقوم الخوارزم بالقيام بطلب عدد صحيح من المستخدم ثم يقوم بإبلاغه فيما اذا كان العدد معدوما (null) ام لا (non null).

Algorithme	C programme

Algorithme exemple_7
Variables:
val:**entiers**
début
ecrire('donner un entier: ')
lire(val)
si val = 0 **alors**
ecrire(val, " est null")
sinon
ecrire(val, " est non null")
fin si
fin
	include<stdio.h>
	main()
	{
	int val**;**
	printf("donner un entire: ")**;**
	scanf("%d ", &val)**;**
	if (val = = 0)
	{
	printf("%d est null", val)**;**
	}
	else
	{
	printf("%d est non null", val)**;**
	}
	}

1.2. العبارات المنطقية

1.1.2. العبارة المنطقية البسيطة:

الشرط البسيط هو مقارنة تعبيرين من نفس النوع، مثلا.(a<0) نوع صحيح أو حقيقي، (s=" op") نوع حرف.

لمقارنات الأحرف، يتم استخدام ترتيب ASCII ، والذي يتبع الترتيب الأبجدي .سيكون الحرف الذي يتم وضعه قبل الآخر حسب الترتيب الأبجدي أقل من الآخر مثلا "a " أقل من "b "، لكن "s "أكبر من "m".

لابد من الإشارة الى ان الشرط البسيط لا يعني أنه دائما قصير. فيمكنه أن يكون طويلا كالمقارنة بين عمليتين.

مثال: $(a + b - 3) * c \leq (5 * y - 2) / 3$

2.1.2. العبارة المنطقية المركبة:

يمكن أن تكون العبارة المنطقية مركبة أيضا، أي تتكون من عدة شروط بسيطة أو متغيرات منطقية مرتبطة ببعضها البعض بواسطة عمليات منطقية et، ou، Xor، non.

على سبيل المثال: شرط يتكون من شرطين بسيطين مرتبطين بـ: et، (b <0 et a <0) يكون صحيحا إذا كان a <0 صحيحا وإذا كان b <0 صحيحا.

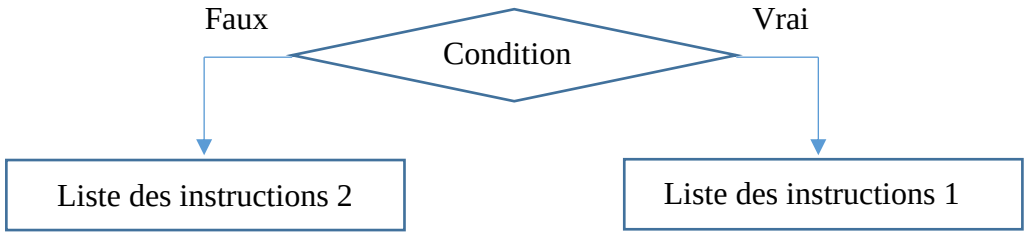
ملاحظة: استخدام الأقواس يجعل من الممكن تعديل المشاكل المحتملة للأولويات المنطقية.

3.1.2. المتغيرات المنطقية

قد تتطلب الضرورة استخدام متغير منطقي في الشرط الذي يحل محل العبارة المنطقية، كما يمكننا إسناد عبارة منطقية إلى متغير منطقي ثم استخدامه في الشرط.

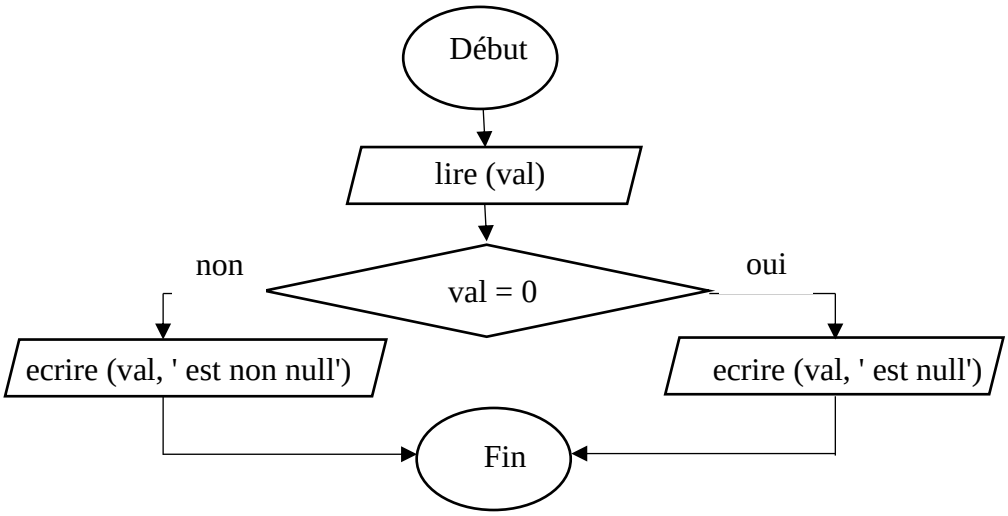
2.2. التمثيل البياني للهيكل الشرطي البسيط

يكون التمثيل البياني للهيكل الشرطي البسيط كالتالي:



مثال:

التمثيل البياني للمثال الأول (العدد الصحيح المعدوم ام لا) يكون كالتالي:



2. الهيكل الشرطي المركب (Struture conditionnelle composée)

في كثير من الاحيان لا يكون لدينا حالتين فقط (الشرط ونقيضه)، بل تكون هناك ثلاث حالات فما أكثر، مما يستلزم استخدام هياكل شرطية متداخلة تسمى هيكل شرطي مركب.

مثال:

اكتب الخوارزم الذي يقوم بطلب عدد صحيح من المستخدم ثم يقوم بإبلاغه فيما إذا كان العدد معدوما (null) او إيجابيا (positif) او سلبيا (négatif).

Algorithme	C programme

Algorithme exemple_8
Variables:
val:**entiers**
début
lire(val)
si val = 0 **alors**
ecrire(val, " est null")
sinon
si val > 0 **alors**
ecrire(val, " est positif")
sinon
ecrire(val, " est négatif")
fin si
fin si
fin
	include<stdio.h>
	main()
	{
	int val**;**
	printf("give integer")**;**
	scanf("%d ", &val)**;**
	if (val = = 0)
	{
	printf("%d is null", val)**;**
	}
	else
	{
	if (val > 0)
	{ **printf**("%d is positive", val)**;** }
	else
	{ **printf**("%d is negative", val)**;** }
	
	}
	}

3. الهيكل الشرطي المتعدد الاختيارات (Struture conditionnelle de choix multiple)

يجعل هذا الهيكل الشرطي من الممكن اختيار التنفيذ بين عدة كتل من التعليمات وفقًا لتغير قيمة متغير معين. فهو يتيح الاختيار من بين أكثر من احتمالين لكنها متعلقة بتغير قيمة متغير واحد.

1.3. الشكل العام

Algorithme	Programme C

selon le cas (variable)
variable = valeur 1 : liste d'instructions 1
variable = valeur 2 : liste d'instructions 2
…
variable = valeur n : liste d'instructions n
sinon
liste d'instructions
fin Si
	switch(variable) {
	case valeur 1: liste d'instructions 1**; break;**
	case valeur 2: liste d'instructions 2**; break;**
	…
	case valeur n: liste d'instructions n**; break;**
	default:
	liste d'instructions **;**
	}

يجدر الإشارة الى ان المعالجة تختلف باختلاف قيمة القيمة من valeur 1 الى valeur n، حيث يتم تنفيذ سلسلة التعليمات المقابلة للقيمة المحققة ويتم تجاهل جميع سلاسل التعليمات الأخرى. أما إذا كان المتغير ليس له قيمة بين كل تلك القيم، فسيتم تنفيذ سلسلة التعليمات المتعلقة ب sinon(default en C) والممثلة باسم liste d'instructions.

لكي يتم تجاهل جميع سلاسل التعليمات الأخرى في حالة تحقق احدى القيم لابد من إضافة تعليمة break بعد كل سلسلة تعليمات متعلقة بقيمة المتغير في برنامج C.

مثال:

اكتب الخوارزم الذي يقوم بطلب عدد صحيح من المستخدم ثم يقوم بإبلاغه باليوم المقابل له في الاسبوع فيما إذا كان العدد بين 1 و 7، اما اذا كان العدد خارج هذا النطاق يقوم بابلاغه ان العدد لا يوافق أي يوم.

Algorithme	Programme C
Algorithme exemple_9 **Variables:** jour:**entier** **Début** **Lire**(jour) **selon le cas** (jour) jour = 1 : **ecrire** ("c'est le Samedi") jour = 2 : **ecrire** ("c'est le Dimanche") jour = 3 : **ecrire** ("c'est le Lundi") jour = 4 : **ecrire** ("c'est le Mardi") jour = 5 : **ecrire** ("c'est le Mercredi") jour = 6 : **ecrire** ("c'est le Jeudi") jour = 7 : **ecrire** ("c'est le Vendredi") **sinon** **ecrire** ("n'est pas conforme à aucun jour") **fin selon** **fin**	**include <stdio.h>** **main()** **{** **int** jour**;** **scanf**("%d",&jour)**;** **switch** (jour) **{** **case** 1: **printf**("c'est le Samedi")**;** **break;** **case** 2: **printf**("c'est le Demanche")**;** **break;** **case** 3: **printf**("c'est le Lundi")**;** **break;** **case** 4: **printf**("c'est le Mardi")**;** **break;** **case** 5: **printf**("c'est le Mercredi")**;** **break;** **case** 6: **printf**("c'est le Jeudi")**;** **break;** **case** 7: **printf**("c'est le Vendredi")**;** **break;** **default:** **printf**(" n'est pas conforme à aucun jour ")**;** **}** **}**

4. تعليمات الانتقال (Branchement)

تسمح تعليمات الانتقال بمقاطعة التسلسل الطبيعي للبرنامج عن طريق القفز من مكان في البرنامج إلى مكان آخر لإكمال تنفيذ البرنامج. تتمثل هذه التعليمات أساسا في تعليمة aller à (goto) التي ينبني عليها ثلاث تعليمات أُخرى ممثله في التعليمات break ، و continue ، و return. سنركز في هذه الفقرة على التعليمة الاساسية aller à مع شرح عام للتعليمات الأخرى كون فهم التعليمات الثلاث الأخرى بصفة دقيقة يحتاج الى تقدم الطالب في الدروس.

1.4. تعليمة aller à

تتيح تعليمة aller à (goto) الانتقال من مكان معين في البرنامج الى مكان اخر حددناه مسبقًا. للقيام بهذا التحديد، لا بد من استخدام الملصقة (étiquette) والتي تسمى أيضا بالانجليزية (label). فالملصقة ليست أكثر من اسم نختاره متبوعًا بنقطتين " :" ويوضع في المكان الذي يراد الانتقال اليه في البرنامج.

مثال:

اكتب الخوارزم الذي يطلب من المستخدم رقما صحيحا ثم يضيف اليه اثنين إذا كان ذلك العدد أقل من عشرة، بينما يضيف اليه ثلاثة إذا كان أكثر أو يساوي عشرة.

Algorithme	Programme C

Algorithme exemple_10
Variables:
a:**entier**
Début
  Lire(a)
  si a < 10 **alors**
  aller à 1
  fin si
  a ← a+1
1: a← a+2
  ecrire(a)
fin
	include <stdio.h>
	main()
	{
	int a**;**
	scanf("%d", &a)**;**
	  if (a<10) **{**
	  goto 1**;**
	  }
	  a = a+1**;**
	1: a = a+2**;**
	  printf("%d",a)**;**
	}

فهنا المتغير a يُضاف اليه 1 ثم 2 إذا كان أكبر او يساوي 10، بينما ينتقل الى إضافة 2 فقط إذا كان اقل من 10.

2.4. تعليمة break

لقد رأينا تعليمة break في الهيكل الشرطي متعدد الاختيارات. فتعليمة break في الواقع لها تأثير الخروج مباشرة من المكان الذي هي فيه الى اقرب نهاية (fin) ثم متابعة تنفيذ البرنامج. ففي مثال أيام الأسبوع تقفز التعليمة break الى نهاية switch(jour) ثم تتابع تنفيذ البرنامج لتجد بعدها النهاية العامة للبرنامج.

3.4. تعليمة continue

تسمح لك تعليمات المتابعة continue، التي لا يمكن استخدامها إلا في الحلقات، بإيقاف تنفيذ التكرار الحالي والانتقال مباشرة إلى التكرار التالي.

4.4. تعليمة return

تنهي تعليمة return تنفيذ البرنامج مباشرة وتعطي كمخرج القيمة التي تحملها سواء كانت مباشرة او عن طريق متغير، على سبيل المثال return(0) تنهي التنفيذ وتعطي كمخرج القيمة 0. ويعتبر أساس استخدام التعليمة return في برمجة الوظائف (les fonctions).

الفصل الرابع: الحلقات (Les boucles)

1. مقدمة (Intrduction)
2. الحلقة Tant que
3. الحلقة Répéter
4. الحلقة Pour

1. مقدمة (Intrduction)

أحيانا نحتاج إلى تكرار أمر برمجي معيّن أكثر من مرة كإدخال كلمة المرور (mot de passe). فعملية الادخال هنا قد تتكرر إذا كانت الكلمة المُدخلة خاطئة. فإذا وُجد هذا التكرار فلا بد من استخدام الحلقات التي تسهل لنا برمجة هذا النوع من الحالات. تنقسم الحلقات الى ثلاثة أنواع: Tant que و Répéter و Pour.

2. الحلقة Tant que

وهي صيغة تبدأ بشرط مسبوق بالكلمة tant que والتي تقابل كلمة "مادام" باللغة العربية و "while" باللغة الانجليزية، وتعني أنه مادام هذا الشرط متحققا فقم بتكرار الأوامر التابعة للحلقة، وحينما لا يتحقق الشرط يتم الخروج من الحلقة والانتقال إلى ما بعدها من اوامر. وتتميز حلقة الtant que بميزتين:

1- مراقبة تحقق الشرط يكون في البداية أي قبل الدخول للأوامر التابعة للحلقة. فإذا كان الشرط محققا، تنفذ الأوامر التابعة للحلقة ثم تعاد الكرّة، وإذا لم يكن كذلك ينتقل التنفيذ الى مابعد الأوامر التابعة للحلقة وينتهي التكرار.

2- عدد التكرارات غير معلوم لأنه متعلق بتحقق الشرط وبالتالي قد يكون صفرا إذا لم يتحقق شرط التكرار منذ البداية.

1.2. الشكل العام:

Algorithme	Programme C
Tant que condition **faire** Liste des instructions **Fin tant que**	**while** (condition) **{** Liste des instructions**;** **}**

حيث يمثل الcondition عبارة منطقية تخضع الى نفس الخصائص التي شرحناها في الفقرة السابقة.

مثال:

اكتب الخوارزم الذي يطلب من المستخدم ادخال كلمة المرور والتحقق في ما إذا كانت مساوية للكلمة "user123".

Algorithme	Programme C
Algorithme exemple_11 **Variables:**	**include <stdio.h>** **main()**

mot_passe: **chaine**
Début
Ecrire('Entrer le mot de passe:')
Lire(mot_passe)
  Tant que mot_passe ≠ 'user123' **faire**
  Ecrire(' Mot de passe incorrecte')
  Lire(mot_passe)
  Fin tant que
Ecrire ('Mot de passe juste')
fin
	{
	char[7] mot_passe**;**
	printf("Entrer le mot de passe")**;**
	scanf("%s", &mot_passe)**;**
	  while (mot_passe != 'user123')
	  { printf("Mot de passe incorrecte")**;**
	  scanf("%s", &mot_passe)**;**
	  }
	printf("Mot de passe juste")**;**
	}

من خلال المثال يتضح ان عملية ادخال كلمة المرور تتكرر بعد كتابة جملة Mot de passe incorrecte للمستخدم كلما أدخل كلمة لا تساوي لكلمة "user123" ، بينما يتوقف التكرار إدا أدخل المستخدم كلمة "user123" فيصبح حينئذ الشرط غير محقق أي 'user123' ≠ mot_passe يعطي نتيجة "faux" وبالتالي ينتقل التنفيذ الى التعليمة الأخيرة ليكتب البرنامج جملة Mot de passe juste للمستخدم.

3. الحلقة Répéter

وتعمل هذه الحلقة بنفس الطريقة التي تعمل بها الحلقة tant que مع وجود اختلافين بينهما. الاختلاف الأول يشترك فيه الخوارزم وبرنامج C بينما يخص الاختلاف الثاني الخوارزم فقط:

1- تضمن الحلقة répéter على الأقل تنفيذ واحد للأوامر التابعة لها لان مراقبة تحقق الشرط يكون في اخر الحلقة بخلاف الحلقة tant que الذي يكون في الأول وبالتالي ليس هناك أي ضمان لتنفيذ الأوامر التابعة لحلقة tant que.

2- تستخدم الحلقة répéter الشرط الذي بتحقُّقه يتم الخروج من الحلقة وانتهاء التكرار وبعدم تحققه يبقى التكرار ساريا، بخلاف الحلقة tant que التي تستخدم الشرط الذي بتحققه يبقى التكرار وبعدم تحققه يتم الخروج من الحلقة وانتهاء التكرار.

1.3. الشكل العام:

Algorithme	Programme C
répéter Liste des instructions **Jusqu'à** condition	**do** **{** Liste des instructions**;** **} while** (condition);

مثال:

في هذا المثال سنستخدم الحلقة répéter في حل المثال السابق.

Algorithme	Programme C

Algorithme exemple_12
Variables:
mot_passe: **chaine**
Début
  répéter
  Ecrire('Entrer le mot de passe: ')
  Lire(mot_passe)
  Jusqu'à mot_passe = 'user123'
Ecrire ('Mot de passe juste')
fin
	include <stdio.h>
	main()
	{
	char[7] mot_passe**;**
	  do {
	  printf("Entrer le mot de passe: ")**;**
	  scanf("%s", &mot_passe)**;**
	  } while (mot_passe != 'user123');
	printf("Mot de passe juste")**;**
	}

من هذا المثال تتضح النقاط التالية:

1- تغير الشرط من استخدام للعملية "≠" في الحلقة tant que الى العملية "=" في الحلقة répéter وذلك لان الاول شرط بقاء التكرار والثاني شرط الخروج كما ذكرنا سابقا.

2- تم حذف الجملة "Mot de passe incorrecte" وذلك لان الحلقة répéter تضمن الدخول على الأقل لمرة واحدة والتي قد تكون فيها كلمة المرور صحيحة وبالتالي لابد من حذف هذه الجملة.

4. الحلقة Pour

تختلف حلقة pour عن الحلقتين السابقتين في امرين اثنين وهما:

1- عدد التكرارات معروف لأجل هذا فإن حلقة pour تمتلك لعدّاد التكرارات (compteur).

2- شرط الخروج من الحلقة مرتبط بانتهاء عدد التكرارات.

1.4. الشكل العام

Algorithme	Programme C
Pour i allant de 1 jusqu'à n **faire** Liste des instructions **Fin pour**	**for** (i=1; i<=n; i++) **{** Liste des instructions**;** **}**

حيث يمثل i عدّاد التكرارات الذي يبدأ بالقيمة الابتدائية (valeur initiale) التي تكون اختيارية والتي اخترنا لها قيمة 1 وينتهي عند القيمة النهائية (valeur finale) الاختيارية أيضا والتي اخترنا لها القيمة n التي تكون معروفة (إما قيمة ثابتة أو متغير من نفس النوع يحمل قيمة ثابتة). في كل تكرار يُضاف واحد

الى قيمة العداد i حتى يصل الى القيمة n التي تمثل التكرار الأخير ومن ثم يخرج من الحلقة ويكمل تنفيذ باقي الأوامر.

مثال: اكتب الخوارزم الذي يقوم بجمع الاعداد العشرة الطبيعية الأولى.

Algorithme	Programme C
Algorithme exemple_13 **Variables:** i,s: **entier** **Début** s ← 0 **Pour** i allant de 1 jusqu'à 10 **faire** s ← s+ i **fin pour** **ecrire** ('la somme est: ', s) **fin**	**include <stdio.h>** **main()** **{** **int** i,s=0; **for** (i=1; i<=10; i++) **{** s = s + i; **}** **Printf**("la somme est: %d", s); **}**

من خلال المثال يتضح تكرار الحلقة pour لتسع مرات باستخدام العدّاد i الذي ينطلق من القيمة 1 وينتهي عند القيمة 10 بحيث يضيف في كل مرة قيمته الى المجموع المتواجد في المتغير s. يجدر الإشارة الى اننا قمنا بإعطاء قيمة أولية للمتغير s محددة في الصفر لكيلا يكون فارغا في اول استخدامه في عملية الجمع ولا يغير من المجموع شيئا بالقيمة الصفر. بعد انتهاء كل التكرارات ينتقل التنفيذ من الحلقة pour الى الأوامر التي بعدها والمحددة في التعليمة ecrire ('la somme est: ', s) والتي من شأنها اخراج النتيجة على شاشة المستخدم. الجدول التالي يوضح عملية تنفيذ الحلقة pour التي تنتهي بالنتيجة 55.

i	s	i + s	i	s	i + s
1	0	1	6	15	21
2	1	3	7	21	28
3	3	6	8	28	36
4	6	10	9	36	45
5	10	15	10	45	55
				55	

5. الحلقات المتداخلة (Boucles imbriquées)

الحلقات المتداخلة عبارة عن حلقتين او أكثر متداخلة فيما بينها لضرورة برمجية استدعت ذلك كاستخدام حلقة pour داخل حلقة tant que او حلقة pour داخل حلقة pour أخرى. فعلى سبيل المثال لو أردنا إعداد برنامج يعطي الاعداد الأولية بين 2 و10 لوجدنا ان الامر يستلزم استخدام حلقتي pour متداخلة. حيث تستخدم حلقة pour للانتقال بين الاعداد من 2 (القيمة الابتدائية) الى 10 (القيمة النهائية) كما هو موضح بالإطار الاحمر، وتستخدم حلقة pour ثانية داخل الأولى للتحقق من أولية كل عدد في هذا المجال بإجراء القسمة المتكررة من 1 (القيمة الابتدائية) إلى هذا العدد (القيمة النهائية) كما هو موضح بالإطار الأخضر. ثم التحقق من وجود قاسمين فقط (وبالتالي عدد أولي) أو أكثر يستلزم استخدام هيكل شرطي داخل الحلقة الأولى لأنه يتكرر مع كل عدد. الخوارزم مع برنامج C موضحان في الجدول التالي:

Algorithme	Programme C

Algorithme exemple_14
Variables:
i,j,d : entier
Début
pour i allant de 2 jusqu'à 10 **faire**
d ← 0
pour j allant de 1 jusqu'à i **faire**
si (i mod j) = 0 **alors**
d ← d+1
fsi
fin pour
si d = 2 **alors**
Ecrire (i,' est nombre premier')
fin si
fin pour
fin
	include <stdio.h>
	main()
	{
	int i,j,d;
	for (i=2 ; i<=10 ; i++) **{**
	d=0;
	for (j=1 ; j<= i ; j++) **{**
	if ((i % j)= =0) **{**
	d = d + 1;
	}
	}
	if (d = = 2) **{**
	printf("%d est nombre premier", i);
	}
	}
	}

يجدر الإشارة الى اننا استخدمنا ثلاث متغيرات صحيحة i و j و d وذلك لان المتغيرين i و j يمثلان العدادين اللازمين للحلقتين الأولى والثانية على التوالي، بينما استخدمنا المتغير d لحساب عدد القواسم لكل عدد حيث يبدأ بالقيمة 0 مع كل عدد جديد ثم يزداد واحد في كل مرة يجد فيها باقي القسمة 0 بين العدد الجديد واعداد الحلقة الثانية، وعند الخروج من الحلقة الثانية (مع بقائه في الحلقة الأولى) يتحقق من عدد القواسم الموجودة فيما إذا كان يساوي اثنان وبالتالي الإعلان على انه عدد اولي، أما اذا كان أكثر فلا يهم البرنامج لأنه لن يتم الإعلان عن أي شيء.

الفصل الخامس: الجداول وسلاسل الاحرف (Les tableaux et les chaines de caractères)

1. مقدمة (Intrduction)
2. نوع الجدول (Type tableau)
3. الجداول متعددة الابعاد (Les tableaux multidimentionnels)
4. سلاسل الاحرف (Les chaines de caractères)

1. مقدمة (Intrduction)

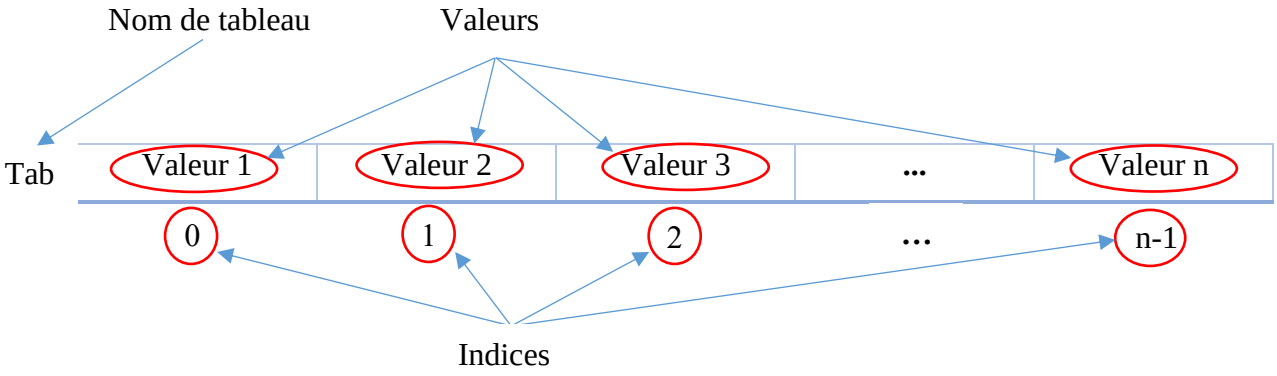
خلال كل ماسبق كنا نستخدم المتغيرات، ورأينا كيف تسمح لنا بتخزين القيم في الذاكرة، ورأينا كذلك أن كل متغير مخصص لاحتواء قيمة واحدة، ولا يمكن للمتغير الواحد أن يستقبل أكثر من قيمة في وقت واحد.

لذلك لو افترضنا أننا نريد إنشاء برنامج لإدخال أسماء الطلبة لاستلزم الامر استخدام عدد كبير من المتغيرات وبالتالي فإن هذه الطريقة ليست مجدية لأننا سنجد صعوبة في استحضار أسماء المتغيرات، ناهيك عن صعوبة تتبع التعليمات وقراءتها.

لحل مثل هذه المشاكل وُجد مفهوم الجداول (Les tableaux)، بحيث نستخدمها حينما نود تخزين مجموعة من القيم المنتمية لنفس نوع البيانات.

2. نوع الجدول (Type tableau)

يمثل الجدول هيكل تخزين له اسم مُختار ومجموعة محددة من أماكن في الذاكرة تمثل عناصر الجدول وتحتوي على قيم من نفس النوع. لكل عنصر منها مؤشر خاص ويحتوي على قيمة واحدة. كما هو موضح في الشكل، يرتكز مفهوم الجدول على ركيزتين أساسيتين هما:



القيم (Valeurs): وهي القيم المراد تخزينها في عناصر الجدول، لو أخذنا مثلا جدول لتخزين نقاط الطلاب في المواد فإن النقاط هي القيم، حيث أن كل نقطة هي عبارة عن قيمة يتم تخزينها في عنصر معين من عناصر جدول المواد.

الفهرس (Indice): ويمثل مؤشر العنصر في الجدول، ويبدأ بصفر وينتهي برتبة آخر عنصر ناقص واحد.

مثلا لو أردنا تخزين نقاط خمسة مواد في جدول نعطيه إسم Notes فإن التمثيل الفعلي سيكون بالشكل التالي:

Notes	10	12	8	14	11
	0	1	2	3	4

1.2. إعلان الجدول (Déclaration de tableau)

يتم الإعلان عن الجدول في جزء الاعلانات (Partie de déclaration) حيث نختار له أمرين اثنين وهما:

1- الاسم: ويخضع لنفس شروط تسمية المتغيرات المذكورة في الفصل الثاني، ويمثل عنوان الجدول في الذاكرة إذ يرتبط مع العنصر الأول في الجدول.

2- عدد العناصر: وهنا تختلف الصيغة بين الخوارزم وبرنامج C. حيث انه في الخوارزم يبدأ بالفهرس الأصغر ثم الفهرس الأكبر يفصل بينهما نقطتين، بينما نكتفي بذكر عدد العناصر في برنامج C. لابد من الإشارة الى اختيار العدد الكافي لتخزين كل القيم المُراد تخزينها إن كان عددها معروف، فإن لم يكن معروف فلابد من اختيار العدد الأحوط.

ويكون الشكل العام للتبليغ عن الجدول كالتالي:

Algorithme	Programme C
Algorithme declare_tableau **Variables:** Nom_de_tableau: **tableau**[indice_initiale..indice_finale] **de** Type_d'éléments **…**	**include<stdio.h>** **main()** **{** Type_d'éléments Nom_de_tableau[Nombre_d'éléments] **;** **…**

مثال: في هذا المثال نقوم بالإبلاغ عن الجدول السابق Notes.

Algorithme	Programme C
Algorithme exemple_15 **Variables:** Notes: **tableau**[0..4] **d'entiers** **…**	**include<stdio.h>** **main()** **{** **int** Notes[5] **;** **…**

2.2. إدخال قيمة لعنصر في الجدول

يتم إدخال قيمة لعنصر في الجدول عن طريق استخدام اسم الجدول مع مؤشر العنصر المراد ادخال القيمة فيه. كما تتم بنفس الطريقة تغيير قيمة عنصر معين بإدخال القيمة الجديدة التي ستمحي القيمة القديمة وتحل محلها. ويكون الشكل العام لإدخال قيمة ما في عنصر معين بالشكل التالي:

Algorithme	Programme C
Nom_de_tableau[indice_de_l'élément]← valeur	Nom_de_tableau[indice_de_l'élément] = valeur**;**

لابد من الإشارة أن القيمة (valeur) اما ان تكون ثابتة أو عن طريق استخدام متغير من نفس النوع.

مثال: في هذا المثال نقوم بإدخال قيم النقاط في الجدول السابق Notes عنصرا بعنصر.

Algorithme	Programme C
Algorithme exemple_16 **Variables:** Notes: **tableau**[0..4] **d'entiers** **Début** Notes[0] ← 10 Notes[1] ← 12 Notes[2] ← 8 Notes[3] ← 14 Notes[4] ← 11 **fin**	**include<stdio.h>** **main()** **{** **int** Notes[5] **;** Notes[0] = 10 **;** Notes[1] = 12 **;** Notes[2] = 8 **;** Notes[3] = 14 **;** Notes[4] = 11 **;** **}**

لابد من الإشارة الى ان النقاط قد أُدخلت من طرف المبرمج مباشرة، فإدا أردنا ترك الادخال الى المستخدم فمن الاحسن استخدام حلقة pour لتسهيل المهمة كون هناك تكرار لعملية الادخال، ويصبح المثال السابق كالاتي:

Algorithme	Programme C

Algorithme exemple_17
Variables:
Notes: **tableau**[0..4] **d'entiers**
i:**entier**
Début
  pour i allant de 0 à 4 **faire**
  Lire(Notes[i])
  fin pour
fin
	include<stdio.h>
	main()
	{
	int i, Notes[5] **;**
	
	  for (i=0; i<=4; i++) **{**
	  scanf("%d \n", &Notes[i])**;**
	  }
	}

و هنا قمنا بقراءة للقيم الخمسة (إدخالها من طرف المستخدم) عن طريق استخدام لحلقة pour، حيث توضع كل قيمة في العنصر المناسب لها Notes[i] بدلالة العدّاد i الذي استخدمناه كفهرس ينتقل من العنصر ذو الفهرس 0 (Notes[0]) إلى العنصر ذو الفهرس 4 (Notes[4]). كما استخدمنا الرمز "\n" في برنامج C للرجوع الى السطر بعد كل ادخال لقيمة عنصر.

3.2. إخراج قيمة عنصر من الجدول

يتم إخراج قيمة عنصر من الجدول بالمناداة عليها باستخدام اسم الجدول مع فهرس العنصر المراد إخراج القيمة منه. فعلى سبيل المثال، لكي نضع قيمة العنصر الثاني من جدول النقاط (النقطة 12) في متغير من نفس النوع (entier) ليكن "p" مثلا، تكون الصيغة بالشكل التالي:

Algorithme	Programme C
P ← Notes[1]	P = Notes[1]**;**

كما يمكن استخراج قيمة عنصر باستدعائها مباشرة دون وضعها في متغير من نفس النوع كاستدعائها مباشرة لمقارنتها مع قيمة أخرى أو لإخراجها للمستخدم. فعلى سبيل المثال، لإخراج النقاط الخمسة السابقة (كل النقاط) للمستخدم بعد إدخالها من طرف المبرمج، نستخدم الحلقة pour لتسهيل المهمة كون هناك عمليه إخراج متكررة، ويكون الخوارزم كالاتي:

Algorithme	Programme C
Algorithme exemple_18 **Variables:** Notes: **tableau**[0..4] **d'entiers** i: **entier** **Début** Notes[0] ← 10 Notes[1] ← 12 Notes[2] ← 8 Notes[3] ← 14 Notes[4] ← 11 **Pour** i allant de 0 à 4 **faire** **Ecrire** (Notes[i]) **Fin pour** **fin**	**include<stdio.h>** **main()** **{** **int** i, Notes[5] **;** Notes[0] = 10 **;** Notes[1] = 12 **;** Notes[2] = 8 **;** Notes[3] = 14 **;** Notes[4] = 11 **;** **For** (i=0; i<=4; i++) **{** **Printf**("%d \n", Notes[i])**;** **}** **}**

3. الجداول متعددة الابعاد (Les tableaux multidimentionnels)

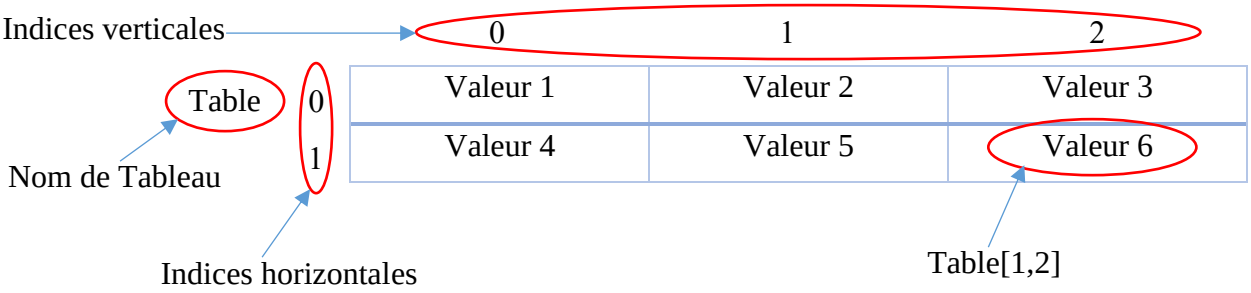
يُعتبر الجدول المدروس في الفقرة السابقة جدول أُحادي البعد (Tableau unidimentionnnel) لانه يتوفر على بعد واحد افقي. وتوجد هناك أيضا جداول متعددة الابعاد (Tableaux multidimentionnels) كالجداول ثنائية البعد (Tableaux bidimentionnels) التي تتميز بتخزين أكبر ويتوفر نوع الجداول ثنائية البعد على الخصائص التالية:

1- له اسم مختار يتميز بنفس ميزات اسم الجدول أحادي البعد.

2- له بعدين أحدهما أفقي (Horizontale) والأخر عمودي (Verticale) ولكل بعد فهرس خاص به.

3- يُعرّف العنصر فيه باسم الجدول مع الفهرسين الأفقي ثم العمودي.

يوضح الشكل المرافق الجدول متعدد الابعاد يحتوي على بعد أفقي يساوي 2 واخر عمودي يساوي 3:



على سبيل المثال لو اردنا تخزين نقاط TD، TP، والامتحان لمادتين لكانت الطريقة الأمثل هي استخدام جدول متعدد الابعاد يحتوي على بعد أفقي يساوي 2 واخر عمودي يساوي 3 حيث يخصص البعد الافقي للمادتين، و يخصص البعد العمودي للنقاط الثلاث. ويكون التمثيل الفعلي بالشكل التالي:

		0	1	2
NotesTable	0	14	12	10
	1	13	14	13

1.3. التبليغ عن الجدول المتعدد الابعاد (Déclaration de tableau multidimentionnel)

يتم التبليغ عن الجدول المتعدد الابعاد في جزء التبليغات (Partie de déclaration) حيث نختار**له** اسما ونوضح مؤشرات العناصر الاولى والأخيرة لكل بعد (الافقي ثم العمودي في الجداول الثنائية الابعاد) على اختلاف التوضيح بين الخوارزم وبرنامج C. ويكون هذا التبليغ كالتالي:

Algorithme

Variables:
Nom_de_tableau**: tableau**[indice_horizontal_initiale..indice_horizontal_finale , indice_verticale_initiale..indice_verticale_finale] **de** Type_d'éléments
…

Programme C

```
include<stdio.h>
main()
{
Type_d'éléments Nom_de_tableau[Nombre_d'éléments_horizontale]
[Nombre_d'éléments_verticale] ;
…
```

مثال: يكون تبليغ الجدول السابق لنقاط المادتين كالتالي:

Algorithme	Programme C
Algorithme exemple_19 **Variables:** NotesTable: **tableau**[0..1, 0..2] **d'entiers** **…**	**include<stdio.h>** **main()** **{** **int** NotesTable[2][3] **;** **…**

2.3. ادخال قيمة لعنصر في جدول متعدد الابعاد

يتم إدخال قيمة لعنصر عن طريق إسناد قيمة من نفس النوع (ثابتة أو عن طريق متغير من نفس النوع) لهذا العنصر بفهارسه (الافقي ثم العمودي في الجداول الثنائية الابعاد). كما تتم بنفس الطريقة تغيير قيمة عنصر معين بإدخال القيمة الجديدة التي ستمحي القيمة القديمة وتحل محلها. ويكون الشكل العام لإدخال قيمة ما في عنصر معين بالشكل التالي:

Algorithme
Nom_de_tableau[indice_horizontale_de_l'élément , indice_verticale_de_l'élément] ← valeur

Programme C
Nom_de_tableau[[indice_horizontale_de_l'élément][indice_verticale_de_l'élément] = valeur**;**

مثال: نقوم في هذا المثال بملئ الجدول السابق NotesTable من طرف المبرمج.

Algorithme	Programme C
Algorithme exemple_20 **Variables:** NotesTable: **tableau**[0..1,0..2] **d'entiers** **Début** NotesTable[0,0] ← 14 NotesTable[0,1] ← 12 NotesTable[0,2] ← 10 NotesTable[1,0] ← 13 NotesTable[1,1] ← 14 NotesTable[1,2] ← 13 **fin**	**include<stdio.h>** **main()** **{** **int** NotesTable[2][3] **;** NotesTable[0][0] = 14 **;** NotesTable[0][1] = 12 **;** NotesTable[0][2] = 10 **;** NotesTable[1][0] = 13 **;** NotesTable[1][1] = 14 **;** NotesTable[1][2] = 13 **;** **}**

فإذا أردنا ملئ نفس الجدول عن طريق المستخدم واخترنا الحلقات لتسهيل البرمجة فلابد من استخدام تقنية les boucles imbriquées بدمج حلقتين ل pour، الاولى تتبع الفهرس الافقي والثانية تتبع الفهرس

العمودي. وبما ان هناك حلقتي pour فيستلوم الامر استخدام عدّادين مختلفين، ليكونا i و j فيصبح الحل كالتالي:

Algorithme	Programme C

Algorithme exemple_21
Variables:
NotesTable: **tableau**[0..1,0..2] **d'entiers**
i,j:**entier**
Début
  pour i allant de 0 à 1 **faire**
  pour j allant de 0 à 2 **faire**
  Lire(NotesTable[i,j])
  fin pour
  fin pour
fin
	include<stdio.h>
	main()
	{
	int i,j, NotesTable[2][3] **;**
	
	  for (i=0; i<=1; i++) **{**
	  for (i=0; i<=2; i++) **{**
	  scanf("%d \n", &NotesTable[i][j])**;**
	  }
	  }
	}

3.3. اخراج قيمة لعنصر من جدول متعدد الابعاد

يتم إخراج قيمة عنصر من الجدول بالمناداة عليها باستخدام اسم الجدول مع فهارس العنصر المراد إخراج القيمة منه (الافقي ثم العمودي في الجداول الثنائية الابعاد). ويتم ذلك باستدعائها إما لوضعها في متغير من نفس النوع أو مباشرة كمقارنتها مع قيمة أخرى أو كإخراجها للمستخدم. فعلى سبيل المثال، لإخراج نقاط الجدول NotesTable للمستخدم بعد إدخالها من طرف المبرمج، نستخدم لحلقتي pour بنفس طريقة الادخال ولكن هذه المرة للإخراج، ويكون الخوارزم مع برنامج C كالتالي:

Algorithme	Programme C

Algorithme exemple_22
Variables:
NotesTable: **tableau**[0..1,0..2] **d'entiers**
i,j:**entier**
Début
NotesTable[0,0] ← 14
NotesTable[0,1] ← 12
NotesTable[0,2] ← 10
NotesTable[1,0] ← 13
NotesTable[1,1] ← 14
NotesTable[1,2] ← 13
 	
  pour i allant de 0 à 1 **faire**
  pour j allant de 0 à 2 **faire**
  ecrire(NotesTable[i,j])
  fin pour
  fin pour
fin
	include<stdio.h>
	main()
	{
	int i,j, NotesTable[2][3] **;**
	
	NotesTable[0][0] = 14 **;**
	NotesTable[0][1] = 12 **;**
	NotesTable[0][2] = 10 **;**
	NotesTable[1][0] = 13 **;**
	NotesTable[1][1] = 14 **;**
	NotesTable[1][2] = 13 **;**
	
	  for (i=0; i<=1; i++) **{**
	  for (i=0; i<=2; i++) **{**
	  printf("%d \t", NotesTable[i][j])**;**
	  }
	  printf("\n")**;**
	  }
	}

والملاحظ في برنامج C اننا استخدمنا للرمز "\t" من أجل وضع فراغات بين القيم الثلاث في كل سطر، بينما الرمز "\n" الذي استخدمناه في تعليمة printf بعد كل خروج من الحلقة الداخلية (مع بقاء التنفيذ في الحلقة الأولى بالرجوع للحلقة الداخلية 10 12 14 ‘مر) فهو للرجوع الى السطر بعد إكمال القيم الثلاث للسطر الأول. وبالتالي القي 13 14 13 ن على الشكل التالي:

4. سلاسل الاحرف (Les chaines de caractères)

تعتبر سلسلة الأحرف حالة خاصة من الجداول أحادية البعد، إذ تعتبر جدول احادي البعد يحوي فقط الاحرف (Caractères) وتنتهي بحرف خاص يسمى الحرف المعدوم ويحمل الشكل '\0'، ويتم استخدامها لتخزين نص معين (كلمة ، جملة ... إلخ). بالإضافة إلى معالجة البيانات النصية، يتم استخدامه على نطاق واسع لأداء الإدخال أو الإخراج للنصوص.

1.4. اعلان سلسلة الاحرف

يتم الإعلان عن سلسلة الحروف بالشكل التالي:

Algorithme
Variables: Nom_de_chaine**: chaine**[Nombre_d'éléments] **Ou** Nom_de_chaine**: tableau**[0..Nombre_élément - 1] **de caractère** …

Programme C
include<stdio.h> **main()** **{** **char** Nom_de_chaine[Nombre_d'éléments] **;** **…**

لابد من الإشارة أن في الخوارزم تُستخدم صيغتين للإعلان الأولى باستخدام كلمة chaine والثانية باستخدام كلمة tableau ، وان عدد العناصر (Nombre_d'éléments) في برنامج C يأخذ بعين

الاعتبار حرف النهاية '\0'. فعلى سبيل المثال لو أعلنا عن سلسلة حروف تحوي 25 عنصر نعطيها اسم اللقب (Prenom) مثلا فهي تحوي فقط 24 حرفا ويبقى العنصر الأخير لحرف النهاية كالتالي:

Algorithme	Programme C
Algorithme exemple_23 **Variables:** Prenom: **chaine**[25] Ou Prenom: **tableau**[0..24] de **caractère** **…**	**include<stdio.h>** **main()** **{** **char** prenom[25] **;** **…**

2.4. ادخال القيم

يتم ادخال القيم في سلسلة الحروف اما حرفا بحرف او ادخال السلسلة بكاملها. ففي الادخال حرفا بحرف لابد من التعامل مع سلسلة الحروف كجدول احادي البعد له ميزة واحدة وهي وضع الحرف بين مزدوجة (' ') إذا كان اسنادا مباشرا كما ذكرنا ذلك سابقا في الفصل الثاني. فعلى سبيل المثال إذا أراد المبرمج ادخال كلمة "yes" بهذه الطريقة في سلسلة حروف اسمها test يكون الخوارزم مع برنامج C كالتالي:

Algorithme	Programme C
Algorithme exemple_24 **Variables:** test: **tableau**[0..3] de **caractère** **début** test[0] ← 'y' test[1] ← 'e' test[2] ← 's' **fin**	**include<stdio.h>** **main()** **{** **char** test[4] **;** test[0] = 'y' ; test[1] = 'e' ; test[2] = 's' ; **}**

فإذا أردنا تحويل الدخال من المبرمج الى المستخدم فلا بد من استخدام لحلقة pour وكذلك الرمز "%c" في برنامج C للدلالة على نوع البيانات caractère بدل الرمز "%d" الذي يدل على النوع entier. وبالتالي تكون النتيجة كالاتي:

Algorithme	Programme C

Algorithme exemple_25
Variables:
test: **tableau**[0..3] de **caractère**
i: **entier**
début
  pour i allant de 0 à 2 **faire**
  lire(test[i])
  fin pour
fin
	include<stdio.h>
	main()
	{
	char test[4] **;**
	int i**;**
	  for (i=0; i<=2; i++) **{**
	  scanf ("%c", &test[i])**;**
	  }
	}

كما ذكرنا سابقا أنه يمكن ادخال السلسلة بكاملها. وهنا لا بد من الاشارة ان هناك أكثر من صيغة للإسناد المباشر في برنامج C. فعلى سبيل المثال إذا أردنا ادخال كلمة "yes" بكاملها كإسناد مباشر، تكون الحالات التالية في برنامج C:

Algorithme	Programme C
Algorithme exemple_26 **Variables:** test: **chaine[4]** **début** test ← 'yes' **fin**	**include<stdio.h>** **main()** **{** **char** test[4] = "yes" **;** ou **char** test[4] = {'y', 'e', 's'} **;** ou **char** test[4] ="ye"+"s" ou **char** test[4] ="y"+"e"+"s" **}**

وهنا لابد من الاشارة الى نقطتين:

1- توضع سلسلة الحروف في برنامج C بين مزدوجتين (" ") بخلاف الحرف الذي يوضع بين مزدوجة واحدة كما ذكرنا سابقا.

2- تُستخدم عملية (+) للجمع بين سلاسل الحروف (concaténation) كما ذكرنا في الفصل الثاني.

إذا أردنا استخدام التعليمة lire فلا بد من استخدام لرمز "%s" الدال على نوع البيانات chaine.

Algorithme	Programme C
Algorithme exemple_27 **Variables:** test: **chaine[4]** **début** lire(test) **fin**	**include<stdio.h>** **main()** **{** **char** test[4] **;** **scanf**("%s", &test) **;** **}**

3.4. اخراج القيم

يتم إخراج القيم إما بطريقة مباشرة بوضع سلسلة الحروف بين مزدوجتين أو عبر الاستدعاء للسلسلة.

فعلى سبيل المثال يكون اخراج عبارة "yes" كالتالي:

Algorithme	Programme C
Algorithme exemple_28 **Variables:** test: **chaine[4]** **début** test ← 'yes' **ecrire**('Le mot est: ',test) **fin**	**include<stdio.h>** **main()** **{** **char** test[4] = "yes" **;** **printf**("Le mot est: %s", test); **}**

4.4. بعض وظائف معالجة سلاسل الاحرف

توجد وظائف (Fonctions) معالجة سلاسل الاحرف في ملفات المكتبات التالية: string.h, stdlib.h, ctype.h. ولكي نستخدم على الأقل واحدة منها في برنامج C لابد من ذكر أحد هذه الملفات بصيغة "include<string.h>" مثلا، مع ملف المكتبة الأساسية "include<stdio.h>" بحيث يكون كل ملف في سطر مستقل. فيما يلي بعض من هذه الوظائف:

-strcpy(chaine_destination, chaine_source): وتستخدم هذه الوظيفة لنقل نسخة من سلسلة الاحرف الموجودة في chaine_source ووضعها في chaine_destination.

- strlen(chaine): وتستخدم لإعطاء عدد الاحرف لسلسلة الاحرف المذكورة chaine.

-strcmp(chaine1, chaine2): وتستخدم للمقارنة بين سلسلتي الاحرف المذكورتين: chaine1 و chain2.

- gets(chaine): وتستخدم لقراءة سلسلة حروف ووضعها في سلسلة الاحرف المذكورة chaine وهي مكافئة للتعليمة scanf("%s",&chaine).

- getchar(): وتستخدم لقراءة حرف واحد، ولتخزينه لابد من اسناده الى متغير من نوع caractère.

- getch(): وتستخدم لقراءة حرف واحد، بدون ان يظهر، ولتخزينه لابد من اسناده الى متغير من نوع caractère.

الفصل السادس: الأنواع المُخصّصة

(Les types personnalisés)

1. مقدمة (Intrduction)
2. التعدادات (Enumérations)
3. السجلات (Enregistrements)

1. مقدمة (Intrduction)

رأينا في الفصل الثاني (الفقرة الرابعة) أنواع المعطيات المعرّفة مسبقا (Prédéfini)، حيث انها لا تمثل حصرا لأنواع البيانات الممكن استخدامها في البرمجة. بل توجد أيضا الأنواع المخصصة التي تستخدم تلك المعطيات المعرّفة مسبقا لتعريف أنواع خاصة للمعطيات وفق نماذج معينة تلعب دورا هاما في تسهيل البرمجة والتخزين، حيث تُستخدم هذه الانواع الخاصة بنفس الطريقة التي تستخدم بها الأنواع المعرّفة مسبقا. تتمثل النماذج المتبعة التعدادات والسجلات.

2. التعدادات (Enumérations)

يعتبر التعداد نموذج يُستخدم لتعريف نوع بيانات خاص له اسم مُختار وقائمة منتهية ومرتبة للقيم التي تمثل الأسماء التي يمكن ان يأخذها المتغير المُنتمي لهذا النوع الخاص. ويستخدم هذا النموذج لتعريف المعطيات التي لها أسماء محددة لا تخرج عنها. ويتم تعريف التعداد باستخدام الكلمة المفتاحية "énumération" (الموافقة لكلمة "enum" في برنامج C) في جزء الإعلانات وقبل المتغيرات (Variables)، والاعلان عن المتغيرات التابعة له في المتغيرات.

1.2. الشكل العام لتعريف التعداد

Algorithme
énumération Nom_énumétation = (Nom1, Nom2,…,Nom n)

Programme C
enum Nom_énumération{Nom1, Nom2, … ,Nomn} **;**

2.2. الشكل العام للإعلان عن التعداد

Algorithme	Programme C
Algorithme declare_enum **énumération** Nom_énumétation = (Nom1, Nom2,…,Nom n) **Variables:** Nom_variable**:** Nom_énumétation **…**	**include<stdio.h>** **enum** Nom_énumération{Nom1, Nom2, … ,Nomn} **;** **main()** **{** **enum** Nom_énumétation Nom_variable **;** **…**

مثال:

سنقوم في هذا المثال بتعريف نوع تعداد خاص يسمى "يوم" (Jour) يأخذ جميع أيام الأسبوع كقيم ثم نقوم بالاعلان عنه بطريقة الخوارزم وكذا برنامج C ، ثم نقوم بإخراج يوم العطلة (jour férié) الموافق ليوم الجمعة (Vendredi).

Algorithme	Programme C
Algorithme exemple_29 **énumération** Jour=(Samedi, Dimanche, Lundi, Mardi, Mercredi, Jeudi, Vendredi) **Variables** jour_ferie**: Jour** **Début** jour_ferie ← Vendredi **Ecrire**('Le jour férié est: ', jour_ferie) **fin**	**include<stdio.h>** **enum** Jour{Samedi, Dimanche, Lundi, Mardi, Mercredi, Jeudi, Vendredi} **;** **main()** **{** **enum** Jour jour_ferie **;** jour_ferie = Vendredi **;** **printf**("Le jour férié est: %d", jour_ferie) **;** **}**

الملاحظ من المثال أننا استخدمنا للرمز "%d" للإخراج في برنامج C، مع العلم ان هذا الرمز يُستخدم فقط للنوع entier من البيانات. وتفسير ذلك، أن قيم التعداد تأخذ ضمنيا ترتيبا حسب تواليها أثناء التعريف بنفس شكل فهرس الجدول احادي البعد، حيث يبدأ هذا الترتيب من الصفر للقيمة الأولى ثم يزداد واحد مع كل قيمة موالية حتى اخر قيمة. من هذا المنطلق، فإن الرمز "%d" يدل على ترتيب القيمة الموافقة ل (Vendredi)والذي هو 6 والذي أُسند مع قيمته للمتغير"jour_ferie ". ففي هذا المثال تأخذ القيم ترتيبها بالشكل التالي:

enum Jour{Samedi, Dimanche, Lundi, Mardi, Mercredi, Jeudi, Vendredi}
0 1 2 3 4 5 6

3.2. مقارنة قيم التعدادات

تتم عملية المقارنة بين قيم التعداد باستخدام ترتيبها. فإذا افترضنا ان لدينا متغيرjour1 من نوع Jour المعرف في المثال السابق يحمل لقيمة "Mardi" مثلا، فإن المقارنة تكون كالتالي:

Opération	Exemples
= Ou ≠	jour1 = Mardi ou jour1 ≠ Lundi
< ou >	jour1 < Jeudi ou jour1 > Samedi

jour1 ≤ Jeudi ou jour1 ≥ Mardi	≤ ou ≥

مثال:

سنقوم في هذا المثال بتعريف تعداد اسمه لون couleur يأخذ القيم (rouge, vert, orange) ثم نطلب من المستخدم ادخال رقم بين 0 و 2 ثم يُخرج له البرنامج الارشاد الموافق للرقم المُختار.

Algorithme	Programme C
Algorithme exemple_30 **énumération** couleur=(rouge, vert, orange) **Variables** a**:couleur** x**: entier** **Début** **Ecrire**('Entrez un numéro entre 0 et 2') **Lire** (x) a ← (couleur) x **si** a > rouge **alors** **ecrire**('Passage autorisé') **si** a = orange **alors** **erire**(' Soyez prudent !') **fin si** **sinon** **ecrire**(Passage interdit') **fin si** **fin**	**include<stdio.h>** **enum** couleur{rouge, vert, orange} **;** **main()** **{** **enum** couleur a **;** **int x ;** **printf**("Entrez un numéro entre 0 et 2") **;** **scanf**("%d", &x)**;** a = (couleur) x **;** **if** (a > rouge) { **printf**("Passage autorisé")**;** **if (a = = orange) {** **printf**(" Soyez prudent !")**;** **}** **else {** **printf**("Passage interdit) **;** **}** **}**

يجدر الاشارة هنا الى اننا استخدمنا اسم التعداد "couleur" بين قوسين في اسناد القيمة الى المتغير "a" وذلك لأجل اسناد الرقم الموجود في المتغير"x" وكذلك القيمة الموافقة لذلك الرقم في التعداد "couleur". ثم قمنا بالمقارنة مع القيمة "rouge" حيث يعطينا البرنامج ارشاد Passage autorisé إذا كانت القيمة أكبر أي vert أو orange وتضيف الارشاد ! Soyez prudent اذا كانت القيمة "orange" ، بينما يُخرج البرنامج للإرشاد Passage interdit اذا كانت القيمة "rouge". يكون التنفيذ حسب الرقم المدخل كالتالي:

الارشاد	القيمة المُدخلة
Passage interdit	0
Passage autorisé	1
Passage autorisé Soyez prudent !	2

3. السجلات (Enregistrements)

تُستخدم السجلات للتعريف واستخدام نوع بيانات خاص ومركب من أنواع أخرى (معرفة مسبقا و/او خاصة) تُشكّل الحقول (Champs) وله اسم مُختار. ويتم تعريف المسجلة باستخدام الكلمة المفتاحية "enregistrement" (الموافقة لكلمة "struct" في برنامج C) في جزء الإعلانات (مثل التعدادات) وقبل المتغيرات (Variables)، والاعلان عن المتغيرات التابعة له في المتغيرات.

1.3. الشكل العام لتعريف السجل

Algorithme	Programme C
Type Nom_enregistrement = **enregistrement** Nom_champ 1: Type 1 Nom_champ 2: Type 2 … Nom_champ n: Type n **Fin_** Nom_enregistrement	**Typedef struct {** Nom_champ 1: Type 1 ; Nom_champ 2: Type 2 ; … Nom_champ n: Type n ; **}** Nom_enregistrement ;

2.3. الشكل العام للإعلان عن السجل

Algorithme	Programme C
Algorithme exemple_31 **Type** Nom_enregistrement = **enregistrement** Nom_champ_1: Type_1 Nom_champ_2: Type_2 … Nom_champ_n: Type_n **Fin_** Nom_enregistrement **Variables:** Nom_variable**:** Nom_ enregistrement **…**	**include<stdio.h>** **Typedef struct {** Type_1 Nom_champ_1 ; Type_2 Nom_champ_2 ; … Type_n Nom_champ_n ; **}** Nom_enregistrement ; **main() {** **typedef** Nom_ enregistrement Nom_variable **;** **…**

مثال:في هذا المثال سنعرف سجل للمادة الدراسية (module) ونعلن عن متغير له باسم "test" كما يلي:

Algorithme	Programme C
Algorithme exemple_32 **Type** module = **enregistrement** Nom_module: **chaine[25]** Specialite: **chaine[25]** Semestre: **entier** Coefficient: **entier** **Fin_** module **Variables:** test**:** module : **…**	**include<stdio.h>** **Typedef struct {** **char[25]** Nom_module **;** **char[25]** Specialite **;** **int** Semestre **;** **int** Coefficient **;** **}** module **;** **main() {** **typedef** module test **;** **…**

3.3. إسناد القيم لسجل

لإسناد قيمة لسجل لابد من تحديد الحقل الذي توضع فيه ثم يُستدعى هذا الحقل باستخدام اسم السجل تليه نقطة ثم اسم الحقل. فعلى سبيل المثال، لإسناد القيم للسجل "test" لابد من وضع سلسلة الحروف "algorithme1" في Nom_module وكذا "informatique" في Specialite و 1 في Semestre و 4 في Coefficient وبالتالي يكون الاسناد كالتالي:

Algorithme	Programme C
test.Nom_module ← 'algorithme1' test.Specialite ← 'informatique' test.Semestre ← 1 test.Coefficient ← 4	test.Nom_module = "algorithme1" ; test.Specialite = "informatique" ; test.Semestre = 1 ; test.Coefficient = 4 ;

كما يمكن الاسناد بين السجلات من نفس النوع مباشرة. فعلى سبيل المثال إذا كان لدينا متغير اخر من نفس نوع "module" نعطيه اسم "test2" فيكون اسناد "test" الى "test2" كالتالي:

Algorithme	Programme C
test2 ← test	test2 = test ;

وهذا يكافئ اسناد القيم حقلا بحقل كالاتي:

Algorithme	Programme C
test2.Nom_module ← test.Nom_module test2.Specialite ← test.Specialite test2.Semestre ← test.Semestre test2.Coefficient ← test.Coefficient	test2.Nom_module = test.Nom_module ; test2.Specialite = test.Specialite ; test2.Semestre = test.Semestre ; test2.Coefficient = test.Coefficient ;

4.3. قراءة القيم للسجل

لقراءة القيم (ادخال القيم) لابد أيضا من تحديد الحقول المناسبة لهذه القيم. فعلى سبيل المثال لو تركنا القيم السابقة للمستخدم لإدخالها لكانت النتيجة كالتالي:

Algorithme	Programme C
Algorithme exemple_33 **Type** module = **enregistrement** Nom_module: **chaine[25]** Specialite: **chaine[25]** Semestre: **entier** Coefficient: **entier** **Fin_** module **Variables:** test**:** module : **Début** **Lire**(test.Nom_module) **Lire**(test.Specialite) **Lire**(test.Semestre) **Lire**(test.Coefficient) **fin**	**include<stdio.h>** **Typedef struct {** **char[25]** Nom_module **;** **char[25]** Specialite **;** **int** Semestre **;** **int** Coefficient **;** **}** module **;** **main() {** **typedef** module test **;** **scanf**("%s", & test.Nom_module) **;** **scanf**("%s", & test.Specialite) **;** **scanf**("%d", & test.Semestre) **;** **scanf**("%d", & test.Coefficient) **;** **}**

5.3. إخراج القيم من السجل

لإخراج القيم من السجل (كتابة القيم) لابد من استدعاء الحقول المراد اخراج القيم منها. فعلى سبيل المثال، لو أردنا ادخال القيم السابقة من طرف المبرمج ثم إخراجها للمستخدم، ستكون النتيجة كالاتي:

Algorithme	Programme C

```
Algorithme exemple_34
Type module = enregistrement
    Nom_module: chaine[25]
    Specialite: chaine[25]
    Semestre: entier
    Coefficient: entier
Fin_ module

Variables:
 test: module :
Début
test.Nom_module ← 'algorithme1'
test.Specialite ← 'informatique'
test.Semestre ← 1
test.Coefficient ← 4

ecrire('Le nom est: ', test.Nom_module)
ecrire ('La spécialité est: ', test.Spetialite)
ecrire ('Le semestre est: ', test.Semestre)
ecrire ('Le coefficient est: ', test.Coefficient)
fin
```

```
include<stdio.h>
Typedef struct {
    char[25] Nom_module ;
    char[25] Specialite ;
    int Semestre ;
    int Coefficient ;
    } module ;

main() {
typedef module test ;

test.Nom_module = "algorithme1" ;
test.Specialite = "informatique" ;
test.Semestre = 1 ;
test.Coefficient = 4 ;

printf("Le nom est: %s", test 1.Nom_module) ;
printf ("La spécialité est: %s", test.Specialite);
printf ("Le semestre est: %d", test.Semestre) ;
printf ("Le coefficient est: %d", test.Coefficient) ;
}
```

ملاحظة:

يمكن تخزين جميع مواد الجامعة باستخدام جدول احادي البعد يأخذ قيم من نوع "module" وكذلك حلقة pour . فلو افترضنا انه يوجد 500 مادة تدريسية، يكون الادخال من طرف المستخدم في جدول نعطيه اسم "tous_modules" كالتالي:

Algorithme	Programme C
Algorithme exemple_35 **Type** module = **enregistrement** Nom_module: **chaine[25]** Specialite: **chaine[25]** Semestre: **entier** Coefficient: **entier** **Fin_** module **Variables:** tous_modules**: tableau**[0..499] de module i: **entier** **Début** **Pour** i allant de 0 à 499 **faire** **Lire**(tous_modules[i].Nom_module) **Lire**(tous_modules[i].Specialite) **Lire**(tous_modules[i].Semestre) **Lire**(tous_modules[i].Coefficient) **Fin pour** **fin**	**include<stdio.h>** **Typedef struct {** **char[25]** Nom_module **;** **char[25]** Specialite **;** **int** Semestre **;** **int** Coefficient **;** **}** module **;** **main() {** **typedef** module tous_modules[500] **;** **int** i **;** **for** (i=0; i<500; i++) **{** **scanf**("%s\n", & tous_modules[i].Nom_module)**;** **scanf**("%s\n", & tous_modules[i].Specialite) **;** **scanf**("%d\n", & tous_modules[i].Semestre) **;** **scanf**("%d\n", & tous_modules[i].Coefficient) **;** **}** **}**

الملحق

سلاسل تمارين الفصول مع حل بعض التمارين لكل سلسلة

سلسلة تمارين الفصل الأول (مقدمة)

التمرين 1:

أمامك عشرة أكياس بها 100 قطعة ذهبية. يوجد كيس من العملات المزيفة. العملة الحقيقية تزن 5 جرامات والمزيفة 4.5 جرام. لدينا ميزان رقمي، وبالتالي يعطي وزنًا دقيقًا بالجرام.

كيف يمكن تحديد كيس العملات المزيفة في وزن واحد لا يتجاوز 100 قطعة ذهبية؟

التمرين 2:

اكتب الخوارزمية (الخطوات المتسلسلة) التي تسمح لك بتناول وجبة في مطعم.

التمرين 3:

إذا كان لدينا ثلاثة دلاء. الدلو الأول "B1" ممتلئ بسعة 10 لترات، الدلو الثاني "B2" فارغ بسعة 7 لترات، والدلو الثالث "B3" فارغ بسعة 3 لترات. اكتب الخوارزمية التي تسمح لنا بالحصول على 5 لترات.

التمرين 4:

لدى أحد المزارعين خروف وذئب وكومة من العشب. اكتب خوارزمية تسمح للمزارع بنقل الخروف والذئب والعشب إلى الجانب الآخر من النهر باستخدام قارب لا يمكنه حمل سوى شيء واحد (خروف أو ذئب أو عشب) في المرة الواحدة، دون ترك الذئب وحده مع الخروف أو الخروف وحده مع العشب.

حل بعض تمارين سلسلة الفصل الأول

التمرين 1:

لتحديد كيس العملات المزيفة في وزن واحد: عليك أن تضع على الميزان عملة واحدة من الكيس الأول، واثنتين من الكيس الثاني، وثلاث عملات من الكيس الثالث... وهكذا، مما يعطي 55 عملة. يجب أن تعطي هذه العملات وزنًا: 55 × 5 = 275 جرامًا. بمجرد وجود وزن لا يتوافق مع 5 جرام، سيكون كيس العملات المزيفة.

على سبيل المثال: إذا كان لدينا:

0.5 جرام مفقود بينما الكيس الأول للعملات المزيفة.

1 جرام مفقود بينما الكيس الثاني للعملات المزيفة.

1.5 جرام مفقود بينما الكيس الثالث للعملات المزيفة

التمرين 3:

0. الحالة الأولية (B1=10، B2=0، B3=0)

1. صب 7 لتر من B1 في B2 (B1=3،B2=7،B3=0)

2. صب 3 لتر من B2 في B3 (B1=3،B2=4،B3=3)

3. صب B3 في B1 (B1=6،B2=4،B3=0)

4. صب 3 لتر من B2 في B3 (B1=6،B2=1،B3=3)

5. صب B3 في B1 (B1=9،B2=1،B3=0)

6. صب B2 في B3 (B1=9،B2=0،B3=1)

7. صب 7 لتر من B1 في B2 (B1=2،B2=7،B3=1)

8. نسكب 2 لتر من B2 في B3 (B1=2،B2=5،B3=3)

سلسلة تمارين الفصل الثاني (الخوارزم المتسلسل البسيط)

التمرين 1:

حدد الخطأ إذا كان موجودًا لكل معرف:

5TD, _3, bonne chance, TP, mathématiques, Δ, D-A, end, TP

التمرين 2:

أعط قيم المتغيرات بعد تنفيذ كل تعليمة من هذه الخوارزمية.

Algorithme Exo3

Variables A, B: entiers

début

A←7

B←A-4

A←A-1

B←A+5

Fin

التمرين 3:

نفد الخوارزمية التالية

Algorithme assignment
Variables a, b, c, x, y, z : entiers
d, e, f, g : Booléen
h, i : caractère
début

a ←2
h ←'c'
b ←3 * a
c ←10
i ←'r'
d ←(b - c) = a
e ←Not d
c ←b – c - a
f ←(c ≠ 12) and (e)
y ←c
x ←b
g ←h > i
fin

التمرين 4:

اكتب خوارزمية تقرأ اسم وسنة ميلاد شخص ما، بالإضافة إلى السنة الحالية. ثم تعرض عمر ذلك الشخص.

مثال:

Name: Said
Year of birth: 2005
Current year: 2023
Hello Said, you are 18 years old.

التمرين 5:

اكتب خوارزمية وبرنامجها بلغة C لحساب متوسط وحدة التحليل.

التمرين 6:

اكتب خوارزمية وبرنامجها بلغة C يقرأ الوقت بالثواني ثم يعرضه بالساعات والدقائق والثواني.

التمرين 7:

اكتب خوارزمية وبرنامجها بلغة C يستقبل الزاوية بالدرجات، ثم يعرض هذه الزاوية بالقرادس (gr) والراديان(rad) .

ملاحظة:

$$rad = deg° \times \pi/180$$

$$gr=\pi/200\ rad$$

حل بعض تمارين سلسلة الفصل الثاني

التمرين 2:

var A, B: integer **Begin**	A	B
A←7	7	/
B←A-4	7	3
A←A-1	6	3
B←A+5	6	11

التمرين 4:

Algorithme CalculateAge
Variables: name: chaine[25]
birthYear, currentYear, age : entier
début

ecrire("Name: ")
lire (name)

ecrire ("Year of birth: ")

lire (birthYear)

ecrire ("Current year: ")

lire (currentYear)

age ← currentYear - birthYear
ecrire ("Hello ", name, ", you are ", age, " years old.")
fin

التمرين 7:

```
Algorithme angle
constant pi=3.14
Variables: deg, rad, gr: réels
debut
 ecrire ("enter an angle in degrees
")
 lire (deg)
 rad ← deg *pi/180
 gr ← rad *pi/200
 ecrire (deg, "°=", rad, " rad = ",
gr," gr")
fin
```

```
#include <stdio.h>
int main() {
 const float pi=3.14;
 float deg, rad, gr;
 printf("enter an angle in degrees
");
 scanf("%f",&deg);
 rad = deg *pi/180;
 gr = rad *pi/200;
 printf("%.2f°=%.2f rad =%.2f gr",
deg, rad, gr);
}
```

سلسلة تمارين الفصل الثالث (الهياكل الشرطية)

التمرين 1:

اكتب خوارزمية مع برنامجها C تسمح بإدخال 3 أعداد صحيحة. ثم تعرض هذه الخوارزمية الأعداد الزوجية فقط.

التمرين 2:

اكتب خوارزمية وبرنامج C الخاص بها تسمح بقراءة عددين صحيحين A وB وتتحقق مما إذا كان A قابلاً للقسمة على B أم لا.

التمرين 3:

اكتب خوارزمية وبرنامجها C تحسب الزكاة. تستقبل هذه الخوارزمية ثروة الشخص مع ثمن جرام واحد من الذهب. ثم تعرض مقدار الزكاة. مع العلم أن نسبة الزكاة هي 2.5% وحد النصاب هو 85 جرامًا من الذهب.

التمرين 4:

اكتب خوارزمية تقرأ السنة A وتخبرنا ما إذا كانت هذه السنة سنة كبيسة (فبراير به 29 يومًا) أم لا.

ملاحظة:

• إذا لم تكن A قابلة للقسمة على 4، فإن السنة ليست سنة كبيسة.

• إذا كانت A قابلة للقسمة على 4، فإن السنة تكون سنة كبيسة ما لم تكن A قابلة للقسمة على 100 وليس على 400.

التمرين 5:

اكتب خوارزمية وبرنامجها C لآلة حاسبة صغيرة تتيح للمستخدم إجراء إحدى العمليات التالية (جمع رقمين، طرح رقمين، قسمة رقمين، ضرب رقمين، الجذر التربيعي لرقم، وقوة رقم).

التمرين 6:

يبيع متجر 3 منتجات، p1 وp2 وp3، بأسعار 24 DA و32 DA و43 DA على التوالي. يُمنح خصم بنسبة 1% إذا تجاوز المبلغ قبل الضريبة 220 DA، ويُمنح خصم بنسبة 2% إذا تجاوز 560 DA.
اكتب خوارزمية وبرنامج C الخاص بها تقرأ الكمية المشتراة لكل منتج، ثم تعرض:

• السعر الإجمالي لكل منتج، السعر الإجمالي قبل الضريبة، السعر الإجمالي قبل الضريبة بعد الخصم، مبلغ ضريبة القيمة المضافة (مع العلم أن معدل ضريبة القيمة المضافة هو 19%)، وأخيرا المبلغ الإجمالي الذي يجب دفعه.

حل بعض تمارين سلسلة الفصل الثالث

التمرين 1:

```
Algorithme pair
Variables: n1, n2 , n3 : entiers
début
 ecrire("enter 3 nbrs ")
 lire (n1, n2 , n3)
 si n1 mod 2 =0 alors
  ecrire(n1)
 fin si
 si n2 mod 2 =0 alors
  ecrire(n2)
 fin si
 si n3 mod 2 =0 alors
  ecrire(n3)
 fin si
fin
```

```
#include <stdio.h>
int main() {
 int n1, n2 , n3;
 printf("enter 3 nbrs ");
 scanf("%d%d%d",&n1,&n2,&n3);
 if (n1 % 2 ==0)
  printf("%d\n",n1);
 if (n2 % 2 ==0)
  printf("%d\n",n2);
 if (n3 % 2 ==0)
  printf("%d\n",n3);
}
```

التمرين 3:

```
Algorithme zakat
Constant Nissaabe=85
Variables: money, zkt , Nsb_arg,
gold_price: réels
début
 ecrire("enter your amount ")
 lire (money)
 ecrire("enter the price of gold
")
 lire (gold_price)
 Nsb_arg← gold_price* Nissaabe
 si money >= Nsb_arg alors
  zkt ←money*2.5/100
 sinon
  zkt ← 0
 fin si
 ecrire("le montant du zakat est
", zkt)
fin
```

```
#include <stdio.h>
int main() {
 Const int Nissaabe=85;
 float money,zkt ,
Nsb_arg,gold_price;
 printf("enter your amount ");
 scanf("%f",&money);
 printf("enter the price of gold ");
 scanf("%f",&gold_price);
 Nsb_arg= gold_price* Nissaabe ;
 if (money >= Nsb_arg)
  zkt = money*2.5/100;
 else
  zkt = 0;
 printf("the amount of zakat is
%.2f\n", zkt);
}
```

التمرين 5:

```
Algorithme mini_calculator
Variables: x, y: réels
    Choice : caractère
début
 ecrire("mini-calculator ")
 ecrire ("+ : addition ")
 ecrire ("- : subtraction ")
 ecrire ("* : multiplication ")
 ecrire ("/ : division ")
 ecrire ("r : square root ")
 ecrire ("^ : power ")
 ecrire ("enter your choice ")
 lire (Choice)
 si Choice = 'r' alors
   ecrire ("enter a nbr ")
   lire (x)
   ecrire ("√",x, "=",√ x)
 sinon
  ecrire ("enter two nbrs ")
  lire (x, y)
  selon le cas (Choice)
  Choice ='+': ecrire
(x,"+",y,"=",x+y)
  Choice = '-': ecrire (x," -
",y,"=",x-y)
  Choice = '*': ecrire (x,"
*",y,"=",x*y)
  Choice = '/': ecrire
(x,"/",y,"=",x/y)
  Choice = '^': ecrire (x,"
^",y,"=",x^y)
  fin selon
 fin si
fin.
```

```
#include <stdio.h>
#include <math.h>
int main() {
 float x, y;
 char Choice ;
 printf("mini-calculator \n");
 printf("+ : addition \n");
 printf("- : subtraction \n");
 printf("* : multiplication \n");
 printf("/ : division \n");
 printf("r : square root \n");
 printf("^ : power \n");
 printf("enter your choice \n");
 scanf("%c", &Choice);
 if (Choice == 'r'){
   printf("enter a nbr \n");
   scanf("%f", &x);
   printf("√%f=%f", x,  sqrt( x));
 }else{
  printf("enter two nbrs ")
  scanf("%f",&x, y)
  switch (Choice) {
  case '+': printf("%f+%f=%f",x, y,
x+y); break;
  case '-': printf("%f-%f=%f",x, y, x-
y); break;
  case '*': printf("%f*%f=%f",x, y,
x*y); break;
  case '/': printf("%f/%f=%f",x, y,
x/y); break;
  case '^': printf("%f^%f=%f",x, y,
pow(x,y));
  } // end of switch
 } // end of else
 Return 0 ;
}
```

سلسلة تمارين الفصل الرابع (الحلقات)

التمرين 1:

اكتب خوارزمية وبرنامجها C الخاص بها لحساب عاملي عدد.

ملاحظة: 0!=1 et n!=1×2×…× n 1=!0

التمرين 2:

اكتب خوارزمية وبرنامجها C لتحديد ما إذا كان الرقم أوليًا أم لا.

- باستخدام حلقة poor.
- باستخدام حلقة tant que.
- قم بتعميم هذه الخوارزمية لعرض جميع الأعداد الأولية الأقل من أو تساوي N (N≤).

التمرين 3:

اكتب خوارزمية وبرنامجها C الخاص بها لحساب القاسم المشترك الأعظم. مع العلم أن:

$$PGCD(a,b) = \begin{cases} PGCD\big(b,(a\ \%\ b)\big), & b \neq 0 \\ a, & b = 0 \end{cases}$$

التمرين 4:

اكتب خوارزمية وبرنامجها C لحساب الحد n من متتالية فيبوناتشي المحددة بواسطة:

$$\mathrm{u(n)} = \begin{cases} 0 & si\ n = 0 \\ 1 & si\ n = 1 \\ u(n-2) + u(n-1), si\ n > 1 \end{cases}$$

التمرين 5:

إذا كنت تعلم أن الجذر التربيعي للرقم "a" يتم حسابه بالعلاقة التكرارية التالية:

$$x_{n+1} = \frac{x_n + \frac{a}{x_n}}{2}$$

$$x_0 = 1$$

اكتب خوارزمية وبرنامجها C تحسب الجذر التربيعي للرقم "a" بخطأ تقريب $\varepsilon=10^{-6}$. بعبارة أخرى $(x_n)^2 - a \leq \varepsilon$.

حل بعض تمارين سلسلة الفصل الرابع

التمرين 2:

```
Algorithme Premier1
Variables : x, i: entiers
    estPrmeier : Booléen
début
 ecrire("entrer un nbr: ")
 lire(x)
 estPrmeier ← true
 pour i allant de 2 jusqu'a x div 2
faire
  si x mod i = 0 alors
   estPrmeier ← false
  fin si
 fin pour
 si estPrmeier alors
  ecrire(x," est Premier ")
 sinon
  ecrire(x," n'est pas Premier ")
 fin si
fin
```

```
#include <stdio.h>
int main() {
 int x, i, isPrime;
 printf("enter a nbr: ");
 scanf("%d", &x);
 isPrime =1;
 for (i=2 ;i<=(x / 2);i++)
  if (x%i==0)
    isPrime=0;
 if (isPrime==1)
  printf("%d est Prime" , x) ;
 else
  printf("%d n'est pas Prime",
x) ;
}
```

```
Algorithme Premier2
Variables: x, i: entiers
    estPrmeier: Booléen
début
 ecrire("entrer un nbr: ")
 lire(x)
 i←2
 estPrmeier ←true
 tant que i<=(x div 2) et estPrmeier
faire
  si x mod i = 0 alors
   estPrmeier ←false
  fin si
  i←i+1
 fin tant que
 si estPrmeier alors
  ecrire(x," est Premier ")
 sinon
  ecrire(x," n'est pas Premier ")
 fin si
fin
```

```
#include <stdio.h>
int main() {
 int x, i, isPrime;
 printf("enter a nbr: ");
 scanf("%d", &x);
 i=2 ;
 isPrime =1;
 while ((i<=(x / 2) &&
(isPrime==1)) {
  if (x%i==0)
    isPrime=0;
  i++ ;
 }
 if (isPrime==1)
  printf("%d est Prime" , x) ;
 else
  printf("%d n'est pas Prime",
x) ;
}
```

```
Algorithme Premier3
Variables x, i, N: entiers
    estPrmeier :booléen
début
 ecrire("entrer un nbr: ")
 lire(N)
 pour  x allant de 2 jusqu'a N faire
  i←2
  estPrmeier ←true
  tant que i<=(x div 2) et
estPrmeier faire
   si x mod i = 0 alors
    estPrmeier ←false
   fin si
   i←i+1
  fin tant que
```

```
#include <stdio.h>
int main() {
 int x, i, isPrime, N;
 printf("entrer un nbr: ");
 scanf("%d", &N);
 for (x=2 ;x<=N;x++){
  i=2 ;
  isPrime =1;
  while ((i<=x / 2) &&
(isPrime==1)) {
   if (x%i==0)
     isPrime=0;
   i++ ;
  }
  if (isPrime==1)
   printf("%d\n" , x) ;
```

```
  si estPrmeier alors
   ecrire(x)
  fin si
 fin pour
fin
```

```
 }
}
```

التمرين 4:

```
Algorithme Fibonacci
Variables : n, i, Un_2, Un_1, Un: entiers
début
 ecrire ("entrer n ")
 lire(n)
 Un_1 ← 1
 Un ← 0
 pour i allant de 1 jusqu'a n faire
  Un_2 ← Un_1
  Un_1 ← Un
  Un ← Un_2 + Un_1
 Fin pour
 ecrire("U(",n,")=", Un)
fin
```

```
#include <stdio.h>
int main() {
 int n, i, Un_2, Un_1, Un;
 printf("enter n ");
 scanf("%d", &n);
 Un_1= 1;
 Un = 0;
 for(i = 1 ; i<=n; i++) {
   Un_2 = Un_1;
   Un_1 = Un;
   Un = Un_2 + Un_1;
 }
 printf("U(%d)= %d" , n , Un) ;
}
```

التمرين 5:

```
Algorithme racine_carree
Constant eps=0.000001
Variables: a, x: réels
début
 ecrire ("entrer un nbr: ")
 lire(a)
 x ← 1
 répéter
  x ← (x+a/x)/2
 jusqu'a x*x-a≤eps
 ecrire("√",a ,"=", x)
fin
```

```
#include <stdio.h>
int main() {
  const float eps=0.000001;
  float a, x;
  printf("enter a nbr: ");
  scanf("%f", &a);
  x = 1;
  do
    x=(x+a/x)/2 ;
  while(x*x-a>eps);
  printf("v%f = %f" , a, x) ;
}
```

سلسلة تمارين الفصل الخامس (الجداول وسلاسل الأحرف)

التمرين 1:

اكتب خوارزمية و برنامج C الخاص بها تسمح بملء مجموعة من N عدد حقيقي في جدول ثم عرضها بترتيب عكسي.

التمرين 2:

اكتب خوارزمية وبرنامجها C لإيجاد العنصر الأقصى وموقعه في جدول مكون من N عنصر حقيقي.

التمرين 3:

اكتب خوارزمية وبرنامجها C لحساب عدد مرات ظهور عنصر معين في جدول عددي مكون من N عنصر.

التمرين 4:

اكتب خوارزمية وبرنامجها C لحساب حاصل الضرب القياسي لجدولين من أبعاد N. إذا كان u هو جدول الدرجات وكان v هو جدول المعاملات، فقم بتعديل البرنامج لحساب المعدل.

ملاحظة: حاصل الضرب القياسي لجدولين يساوي مجموع حاصل ضرب مكوناتهما المتقابلة.

التمرين 5:

اكتب خوارزمية وبرنامجها C يقوم بجمع مصفوفتين بحجم m/n يحويان اعدادا حقيقية.

التمرين 6:

اكتب خوارزمية وبرنامجها C يتحقق مما إذا كانت المصفوفة متماثلة أم لا.

ملاحظة: تكون المصفوفة M من الرتبة n متماثلة اذا كانت مصفوفة مربعة (عدد الصفوف = عدد الأعمدة=n) تلبي الشرط التالي: M[i, j] = M[j, i] لكل من i وj.

التمرين 7:

اكتب خوارزمية وبرنامجها C يعرض ما إذا كانت سلسلة الاحرف S عبارة عن سلسلة متناظرة (متماثلة) أم لا، وذلك بعد إعطاء سلسلة الأحرف S.

ملاحظة: تكون سلسلة الاحرف متناظرة إذا كانت تقرأ بنفس الطريقة من الخلف ومن الأمام. على سبيل المثال، "radar" عبارة عن سلسلة أحرف متناظرة.

التمرين 8:

اكتب خوارزمية وبرنامجها C يزيل جميع تكرارات حرف معين في سلسلة أحرف معينة ويقوم بنقل باقي الأحرف إلى اليسار.

حل بعض تمارين سلسلة الفصل الخامس

التمرين 2:

```
Algorithme elementMax
Variable i, N, p :entiers
    t:taleau [0..99] de réels
    MAX : réel
début
ecrire("entrer le nombre de elements
<=100")
lire(N)
pour iallant de 0 jusqu'a N-1 faire
  ecrire(i,"=>")
  lire(t[i])
fin pour
MAX←t[0]
p←0
pour i allant de 0 jusqu'a N-1 faire
  si t[i]>MAX alors
    MAX← t[i]
    p← i
  fin si
fin pour
ecrire("le max est ", MAX,
"sa position est ", p)
fin
```

```
#include <stdio.h>
int main(){
int i, N , p;
float t[100] , MAX;
printf("entrer le nombre de
elem<=100");
scanf("%d", &N) ;
for ( i=0 ;i<N ;i++){
  printf("%d=>",i) ;
  scanf("%f", &t[i]) ;
}
MAX=t[0] ;
p=0 ;
for ( i=1 ;i<N ;i++)
  if(t[i]> MAX){
    MAX=t[i] ;
    p=i ;
  }
printf("le max est %.2f sa position
%d", MAX, p) ;
return 0 ;
}
```

التمرين 4:

```
Algorithme produit_scalaire
Variables i, N :entiers
    u,v: taleaux [0..99] de réels
    p: réel
début
ecrire("entrer le nombre de elements
<=100")
lire(N)
ecrire("entrer le premier vector ")
pour i allant de 0 jusqu'a N-1 faire
  ecrire(i,"=>")
  lire(u[i])
fin pour
ecrire("entrer le deuxieme vector ")
pour i allant de 0 jusqu'a N-1 faire
  ecrire(i,"=>")
  lire(v[i])
fin pour
p←0
pour i allant de 0 jusqu'a N-1 faire
  p←p+ u[i]* v[i]
fin pour
ecrire("le produit scalaire est ",
p)
fin
```

```
#include <stdio.h>
int main(){
int i, N;
float u[100],v[100], p;
printf("entrer le nombre de elements
<=100");
scanf("%d", &N) ;
printf("entrer le premier vector ");
for ( i=0 ;i<N ;i++){
  printf("%d=>",i) ;
  scanf("%f", &u[i]) ;
}
printf("entrer le deuxieme vector
");
for ( i=0 ;i<N ;i++){
  printf("%d=>",i) ;
  scanf("%f", &v[i]) ;
}
p=0 ;
for ( i=0 ;i<N ;i++)
  p=p+ u[i]* v[i] ;
printf("le produit scalaire est %f",
p) ;
return 0 ;
}
```

```
Algorithm moyenne
Var i, N, sc :entiers
    u: taleau [0..99] de réels
    v: taleau [0..99] de entiers
    p,avg: réels
début
ecrire("entrer le nombre de elements
<=100")
lire(N)
ecrire("entrer le vector des notes")
pour i allant de 0 jusqu'a N-1 faire
  ecrire(i,"=>")
  lire(u[i])
fin pour
ecrire("entrer le vector des
coefficients ")
pour i allant de 0 jusqu'a N-1 faire
  ecrire(i,"=>")
  lire(v[i])
fin pour
p←0
sc←0
pour i allant de 0 jusqu'a N-1 faire
  p←p+ u[i]* v[i]
  sc←sc + v[i]
fin pour
avg ← p/ sc
ecrire("la moyenne est: ", avg)
fin
```

```
#include <stdio.h>
int main(){
int i, N , sc,v[100];
float u[100], p, moy;
printf("entrer le nombre de elements
<=100");
scanf("%d", &N) ;
printf("entrer le vector des notes
");
for ( i=0 ;i<N ;i++){
  printf("%d=>",i) ;
  scanf("%f", &u[i]) ;
}
printf("entrer le vector des
coefficients ");
for ( i=0 ;i<N ;i++){
  printf("%d=>",i) ;
  scanf("%f", &v[i]) ;
}
p=0 ;
sc=0 ;
for ( i=0 ;i<N ;i++){
  p=p+ u[i]* v[i] ;
  sc=sc + v[i] ;
}
avg =p/ sc ;
printf("la moyenne est: %f", avg) ;
return 0 ;
}
```

التمرين 6:

```
Algorithme Symetrique
Variables: i, j, N :entiers
    M: taleau [0..99,0..99] de réels
    estSymetrique : booleen
début
ecrire("entrer la taille de la
matrice <=100")
lire(N)
ecrire("entrer la matrice ")
pour i allant de 0 jusqu'a N-1 faire
pour j allant de 0 jusqu'a N-1 faire
  ecrire("M["i,"،",j,"]=>")
  lire(M[i,j])
fin pour
fin pour
estSymetrique ← true
pour i allant de 0 jusqu'a N-2 faire
pour j allant de i+1 jusqu'a N-1 faire
  si M[I,j]≠ M[j,i] alors
    estSymetrique ←false
  fin si
fin pour
fin pour
si estSymetrique alors
 ecrire("la matrice est symetrique ")
sinon
```

```
#include <stdio.h>
int main(){
int i, j, N, isSymetrique ;
float M[100][100];
printf("entrer la taille de la
matrice <=100");
scanf("%d", &N) ;
printf("entrer la matrice \n");
for (i=0 ;i< N ;i++)
 for (j=0 ;j<N ;j++){
  printf("M[%d،%d]=>",i, j);
  scanf("%f", &M[i][j]);
 }
isSymetrique=1 ;

for (i=0 ;i< N ;i++)
 for (j=0 ;j<N ;j++)
  if(M[i][j] !=          M[j][i])
isSymetrique=0;

if(isSymetrique)
  printf("la matrice est symetrique
");
else
  printf("la  matrice  n'est  pas
symetrique ");
return 0 ;
```

ecrire("la matrice n'est pas symetrique ") **fin si** **fin**	}

التمرين 8:

Algorithme supprime_ occurrences
Var i, j :entiers
S: chaine[100]
x : caractère
début
ecrire("entrer une chaine de caractère ")
lire(S)
ecrire("entrer le caractere a supprimer ")
lire(x)
i←0
j←0
tant que S[i]≠'\0' **faire**
if S[i]≠ x **alors**
S[j]← S[i]
j←j+1
fin if
i←i+1
fin tat que
S[j]←'\0'
ecrire("la nouvelle chaine est ")
ecrire(S)
end
	#include <stdio.h>
	#include <string.h>
	int main(){
	int i, j;
	char S[100],x;
	printf("entrer une chaine de caractère ");
	gets(S);
	printf("entrer le caractere a supprimer ");
	x=getch();
	i=0;
	j=0;
	while(S[i]!='\0') {
	if (S[i] !=x) {
	S[j]= S[i];
	j++;
	}
	i++;
	}
	S[j]='\0';
	printf("la nouvelle chaine est \n%s",S);
	return 0 ;
	}

سلسلة تمارين الفصل السادس (الأنواع المخصصة)

التمرين 1:

• حدد بنية لتخزين عدد مركب (جزء حقيقي، جزء تخيلي).

• اكتب برنامجًا يقوم بما يلي:

• يقرأ عددًا حقيقيًا (يمكن أن يكون سالبًا) ويحسب ويعرض جذره التربيعي.

• يقرأ عددًا مركبًا، ثم يحسب ويعرض معامله (الحجم).

• يقرأ عددين مركبين، ثم يحسب ويعرض مجموعهما وحاصل ضربهما.

التمرين 2:

• قم بتحديد بنية لتخزين التاريخ (اليوم والشهر والسنة).

• قم بتحديد بنية لتخزين معلومات للاتصال (الاسم ورقم هاتف وتاريخ الميلاد).

• اكتب برنامجًا يملأ مصفوفة من "N" لمعلومات الاتصال. ثم يعرض فقط معلومات الافراد التي لديها رقم هاتف Mobilis (رقم الهاتف يبدأ بـ 06).

التمرين 3:

• حدد بنية لتخزين الوقت (ساعة، دقيقة).

• حدد بنية لتخزين برنامج تلفزيوني لقناة (اسم البرنامج، ووقت البرنامج، والمدة بالدقائق).

• اكتب برنامجًا يملأ مجموعة من "N" برنامج تلفزيوني. ثم يطلب من المستخدم الوقت لعرض البرنامج الحالي الذي يتم بثه في ذلك الوقت.

التمرين 4:

• قم بتحديد بنية لتخزين بيانات الملابس في المتجر (السعر، النوع "رجالي أو نسائي"، والكمية).

• اكتب برنامجًا يقوم بما يلي:

• يملأ جدول ببيانات المبيعات ليوم واحد.

• يحسب ويعرض إجمالي المبيعات لكل نوع من الملابس.

• يحفظ جدول المبيعات في ملف ثنائي (fichier binaire).

حل بعض تمارين سلسلة الفصل السادس

التمرين 1:

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

typedef struct {
  float re,im;
} complex;

int main() {
  float x;
  complex a,b,s,p;
  printf("enter a real number \n");
  scanf("%f",&x);
  if (x>=0) {
    a.re=sqrt(x);
    a.im=0;
  } else {
    a.re=0;
    a.im=sqrt(-x);
  }
  printf("the result is %.2f - %.2fi\n", a.re,a.im );
  printf("enter a complex nbr \n");
  scanf("%f%f",&a.re,&a.im);
  printf("the module is %.2f\n", sqrt(a.re*a.re+a.im*a.im) );
  printf("enter first complex nbr \n");
  scanf("%f%f",&a.re,&a.im);
  printf("enter second nbr complex \n");
  scanf("%f%f",&b.re,&b.im);
  s.re=a.re+b.re;
  s.im=a.im+b.im;
```

```
  p.re=a.re*b.re - a.im*b.im;
  p.im=a.re*b.im + a.im*b.re;
  printf("the sum is %.2f - %.2fi\n", s.re, s.im );
  printf("the multiplication is %.2f - %.2fi\n", p.re, p.im );
  return 0;
}
```

التمرين 4:

```
#include <stdio.h>
typedef struct {
  int price, quantity;
  char type;
} clothing;
int main() {
  FILE *f;
  clothing sales [100];
  int i,n,vh,vf;
  printf("enter number of sales \n");
  scanf("%d",&n);
  for(i=0; i<n; i++) {
    printf("enter the sale N: %d\n",i+1);
    printf("price:");
    scanf("%d",& sales [i]. price);
    printf("type:");
    getchar();
    sales [i].type=getchar();
    printf("quantity:");
    scanf("%d",& sales [i]. quantity);
  }
  vh=0;
  vf=0;
  for(i=0; i<n; i++)
    if (sales [i].type=='h')
      vh=vh+ sales [i]. price * sales [i]. quantity;
```

```
    else
      vf=vf+ sales [i]. price * sales [i]. quantity;
  printf("total men's clothing sales: %d DA\n",vh);
  printf("total women's clothing sales: %d DA\n",vf);
  f=fopen("d:/myFile.dat","w");
  fwrite(sales,sizeof sales,1,f);
  fclose(f);
  return 0;
}
```

المراجع

- Thomas H. Cormen, Algorithmes Notions de base Collection : Sciences Sup, Dunod, 2013.
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Algorithmique - 3ème édition - Cours avec 957 exercices et 158 problèmes Broché, Dunod, 2010.
- Rémy Malgouyres, Rita Zrour et Fabien Feschet. Initiation à l'algorithmique et à la programmation C : cours avec 129 exercices corrigés. 2ième Edition. Dunod, Paris, 2011. ISBN : 978-2-10-055703-5.
- Damien Berthet et Vincent Labatut. Algorithmique & programmation en langage C - vol.1 : Supports de cours. Licence. Algorithmique et Programmation, Istanbul, Turquie. 2014, pp.232.
- Damien Berthet et Vincent Labatut. Algorithmique & programmation en langage C - vol.2 : Sujets de travaux pratiques. Licence. Algorithmique et Programmation, Istanbul, Turquie. 2014, pp.258.
- Damien Berthet et Vincent Labatut. Algorithmique & programmation en langage C - vol.3 : Corrigés de travaux pratiques. Licence. Algorithmique et Programmation, Istanbul, Turquie. 2014, pp.217.
- Claude Delannoy. Apprendre à programmer en Turbo C. Chihab- EYROLLES, 1994.
- Restier François. Langue, linguistique, communication. Hachette, 1989.
- C.Ernst, H.Ettaleb, A.Fonkoua. Algorithmique. Support de cours, Ecole Nationale Superieur des Mines, France, 2005.
- Yves Serra. Evolution des technologies numériques. Bibnum [En ligne], Calcul et informatique. URL :http://journals.openedition.org/bibnum/551;
DOI : https://doi.org/10.4000/bibnum.551