مقدمة في البرمجة
Introduction To Programming

IPG101

الفصل الأول

مدخل إلى الحاسوب والبرمجة

Introduction to Computers and Programming

الكلمات المفتاحية

حاسوب، ترميز، نظام ثنائي، برمجة، لغة برمجة، برنامج، لغة منخفضة المستوى، لغة عالية المستوى، لغة المجمع، المترجم، المفسر.

ملخص الفصل

يتضمن هذا الفصل مدخلاً تأسيسياً لعلوم الحواسيب والبرمجة، حيث يقدم تعريفاً بالحاسوب وأسلوب عمله في تنفيذ المهام الموكلة إليه، ويستعرض أشكال البيانات التي يتعامل معها وأساليب ترميزها، ثم يقدم تعريفها بالبرمجة ولغات البرمجة وتصنفاتها المختلفة، ويوضح الفرق بين المترجم والمفسر، ويختتم بتقديم توصيف لبيئة العمل المثالية لتطوير البرامج.

أهداف الفصل

بنهاية هذا الفصل سيكون الطالب قادراً على:

- معرفة المفاهيم العامة المرتبطة بالحاسوب والبرمجة.
- ذكر أشكال البيانات التي يستطيع الحاسوب التعامل معها.
- فهم أسلوب الحواسيب في تنفيذ المهام الموكلة إليها.
- معرفة أسلوب تمثيل البيانات الرقمية والمحرفية في نظام الحاسوب.
- ذكر أنواع لغات البرمجة الشائعة عالمياً.
- التمييز بين المترجم والمفسر وأسلوب عملهما.
- وصف بيئة تطوير البرنامج المثالية.
- ذكر أنواع الأخطاء التي يمكن أن ترتكب أثناء تطوير البرنامج أو تنفيذه.

محتويات الفصل

1. مقدمة.
2. مفاهيم عامة.
3. كيف يعمل الحاسوب.
4. تمثيل المعطيات والبرامج في نظام الحاسوب.
5. لغات البرمجة.
6. مراحل تطور لغات البرمجة.
7. المترجم والمفسر.
8. بيئة تطوير البرنامج المثالية.
9. أنشطة وتمارين.

1- مقدمة

إن الطيف الواسع من المجالات التي ولجتها النظم الحاسوبية بدءاً من مجالات المالية والأعمال والصناعة والطب والعلوم والتسلية والترفيه وانتهاء بالمجال الحكومي وتكنولوجيا المعلومات. كما أن الكم الكبير من التطبيقات التي أصبحت تعتمد المنطق والتقنية الإلكترونية، ومنها تطبيقات الفوترة وإدارة المستودعات والمصارف والتأمين وحجز الطيران والتصميم وإدارة المشاريع ..... وغيرها من التطبيقات التي لاتنتهي.

إن كل ما سبق يجعل لزاماً علينا مقاربة المفاهيم والمبادئ والأسس التي تقوم عليها هذه التطبيقات بطريقة تمكن طلاب الاختصاصات التقنية والهندسية من أن يكونوا فاعلين في حياتهم العملية من خلال القدرة على بناء واستثمار وترقية مثل هذه التطبيقات انطلاقاً من خلفية علمية وهندسية صلبة.

نركز في هذا الفصل على توضيح العلاقة بين الحواسيب والبرمجة، حيث ننطلق من التعريف بالحاسوب والمفاهيم المرتبطة به وتمثيل البيانات في الحاسوب، ومروراً بمفهوم البرمجة ولغات البرمجة ومفهومي المترجم compiler والمفسر interpreter، وانتهاء بمنهجية حل المسائل استناداً إلى مفاهيم هندسة البرمجيات.

2- مفاهيم عامة

تتعدد العبارات التي يمكن استخدامها لدى تعريف الحاسوب وذلك بحسب التوظيف الذي نرغب به لهذا التعريف، إنطلاقاً من هذا، يمكن تعريف الحاسوب– بشكل عام– على أنه **جهاز يعمل وفقاً لمجموعة من البرامج المخزنة لاستقبال ومعالجة البيانات تلقائياً لكي تعطي معلومات مفيدة نتيجة لتلك المعالجة.**

بالنظر إلى هذا التعريف نجد أنه من المفيد التوقف قليلاً عند بعض المفاهيم.

البيانات Data

هي بشكل أساسي **حقائق لا معنى لها بحد ذاتها**. ولكنها تصبح ذات معنى بعد إدخالها إلى الحاسوب.

يمكن أن توجد البيانات التي يستطيع الحاسوب التعامل معها في عدة أشكال:

1. **البيانات النصية Text Data**: وهي أطول أشكال البيانات، وتتألف عادة من رموز قياسية خاصة، رقمية وأبجدية، ومن أمثلة هذا النوع الرسائل، الميزانيات، التقارير المطبوعة.
2. **البيانات الرسومية Graphics Data**: تتألف من الصور كاللوحات، الرسوم البيانية، الصور الضوئية. إن هذه البيانات تتطلب تمثيلاً أكثر تعقيداً داخل الجهاز من التي تتطلبها البيانات النصية. لأن البيانات الرسومية تكون أكثر تعقيداً وغالباً تستخدم ألواناً مركبة.
3. **البيانات الصوتية Audio Data**: أي نوع من الأصوات – بما في ذلك الموسيقا والكلام –. تستطيع الحواسيب الحديثة تخزين الأصوات بشكل قابل للقراءة من قبل الآلة Machine Readable Form كما تخزن أي نوع من البيانات.

4. **بيانات الفديو Video data**: تتألف من صور متحركة، كالأفلام، أو الصور المنقولة على الهواء مباشرة كالمؤتمرات، المباريات ..... إلخ.

المعلومات Information

عند إدخال البيانات إلى نظام الحاسوب من قبل المستثمرين فهم عادة لا يريدون تلقي نفس البيانات بدون تغيير، بل يريدون من النظام أن يعالج هذه البيانات ويعطي معلومات جديدة ومفيدة. **فالمعلومات إذاً تشير إلى البيانات التي تمت معالجتها إلى شكل له معنى.**

المعالجة Processing

وهي مصطلح يشير إلى **عملية قراءة وتخزين البيانات وفرزها وتصنيفها وإجراء العمليات المختلفة عليها**، كالعمليات الحسابية المعروفة وكذلك العمليات المنطقية، ومن ثم تحليل نتائج هذه العمليات واتخاذ القرارات المناسبة على ضوئها، بالإضافة إلى تلخيص النتائج وإعدادها للإخراج .

البرامج Programs

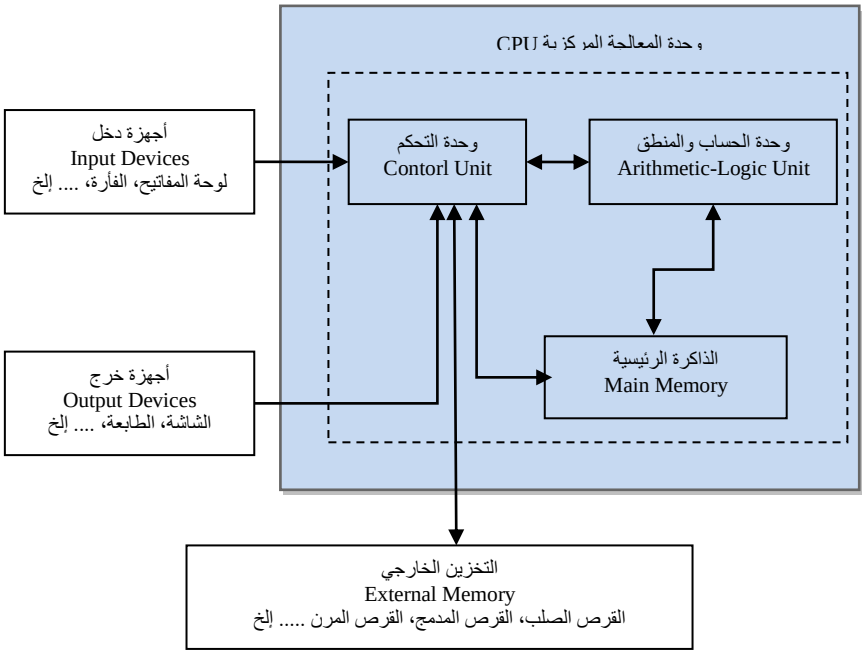
لكي يقوم الحاسوب بإجراء عملية المعالجة فإنه بحاجة إلى من يخبره بما سيفعله. إذاً **البرنامج هو مجموعة من الأوامر والتعليمات التي تخبر الحاسوب كيف سيعالج البيانات لإنتاج المعلومات التي يريدها المستخدم.**

إن كل ماسبق يدفعنا إلى تسجيل الملاحظات والاستنتاجات التالية:

- إن الحاسوب قادر على تنفيذ تعليمة واحدة في اللحظة ذاتها، في حين أن حل أي مسألة هو عبارة عن مجموعة من التعليمات، وبالتالي يجب أن يقدم حل المسألة إلى الحاسوب على شكل **برنامج program**.
- إن الحاسوب جهاز يملك قدرات خارقة على تنفيذ الأوامر بفعالية عالية، إلا أنه –بحد ذاته– غير قادر على المحاكمة العقلية، وبالتالي فإنه غالباً ما ينفذ الأوامر بنفس الترتيب الذي تقدم إليه به. ولذلك يمكن القول أن الوصول إلى حل سليم ونتائج صحيحة من قبل الحاسوب لمسألة ما يتوقف على تقديم الأوامر له وفق ترتيب صحيح وفعال ويأخذ بعين الاعتبار جميع الاحتمالات الممكنة، وبالتالي قبل كتابة برنامج لحل مسألة ما، لابد من وضع خوارزمية algorithm الحل لهذه المسألة.
- بما أن الحواسيب لا تستطيع تشغيل برامج مكتوبة بلغة إنكليزية عادية (كاللغة المتداولة). وبالتالي لابد من كتابة البرامج باستخدام لغات برمجة خاصة Programming Languages **ولغة البرمجة بالتعريف هي عبارة عن مجموعة من الرموز والأوامر والقواعد التي يستطيع الحاسوب قراءتها وترجمتها وتنفيذها.**
- بما أن الحاسوب هو عبارة عن مجموعة من العتاديات المادية hardware مكونة بشكل أساسي من دارات إلكترونية وبالتالي فإنه يفهم لغة واحدة ألا وهي الإشارات الكهربائية والنبضات الإلكترونية، أما البيانات –فكما سبق وأشرنا في هذه الفقرة– فهي غالباً من أشكال مختلفة، وبالتالي فلابد من وجود وسيلة تمكن من تحويل البيانات عند إدخالها إلى الشكل القابل للفهم من قبل الحاسوب، وتحويل المعلومات عند إخراجها إلى الشكل القابل للفهم من قبل مستخدم الحاسوب، هذه الوسيلة تدعى المترجم أو المفسر.

3- كيف يعمل الحاسوب

تملك أغلب الحواسيب في هذه الأيام بنية معمارية تدعى باسم بنية فون نيومان Von Neumann Architecture نسبة إلى العالم الرياضي الهنغاري الذي يحمل الاسم ذاته، هذه البنية موضحة بالشكل 1.



الشكل 1- بنية النظام الحاسوبي وفق Von Nuemann.

لفهم هذه البنية بشكل أكبر، لابد من فهم كيفية قيام الحاسوب بتنفيذ مهمة موكلة إليه.

أولاً : يجب أن نقوم بتحديد المهمة المراد منه إنجازها، توصيفها، معرفة تفاصيلها. بمعنى آخر يجب أن نزوده بالتعليمات التي تبين له كيفية الوصول إلى النتيجة المطلوبة، ثم يجب أن نقدم له المعطيات التي سيقوم باستخدامها. يتم هذا الأمر بواسطة تجهيزات خاصة تدعى **أجهزة الإدخال Input Devices**.

ثانياً : إن التعليمات والمعطيات تقدم له من خارج نظام الحاسوب – يشير الصندوق المظلل في الشكل (1) إلى الحد الفاصل بين نظام الحاسوب والوسط الخارجي – وهي تقدم بشكل غير مفهوم بالنسبة للنظام وبالتالي لا بد من توفر قسم خاص ضمن نظام الحاسوب يقوم بالتقاط هذه التعليمات والمعطيات وتحويلها إلى الشكل الملائم الذي يسمح للحاسوب بالتعامل معه. ومن ثم إرسالها إلى الجزء المعني بالتعامل معها، يدعى هذا القسم باسم **وحدة التحكم Control Unit** .

ثالثاً : تتم في المرحلة التالية ، معالجة المعلومات والمعطيات من قبل وحدة خاصة تدعى **وحدة الحساب والمنطق Arithmeitc-Logic Unit** والتي تشكل مع وحدة التحكم ما يسمى **وحدة المعالجة المركزية Central Processing Unit** . ولما كان تنفيذ التعليمات يتم بشكل متسلسل وبالتالي لابد من تخزين التعليمات والمعطيات في وحدة تخزين Storage Unit (تنقسم إلى **التخزين الداخلي** أو الذاكرة الرئيسية Main Memory **والتخزين الخارجي** External Memory) للحفاظ عليها لحين احتياجها أو لحين إظهارها بعد إنجاز المعالجة. مع التنويه إلى أن وحدة المعالجة المركزية لاتتعامل بشكل مباشر إلا مع الذاكرة الرئيسية وبالتالي فأي أمر أو تعليمة أو بيانات مخزنة في وسيط تخزين خارجي لابد من نقلها إلى الذاكرة الرئيسية عند الرغبة بالمعالجة.

رابعاً : إن ناتج المعالجة يكون بالشكل الذي يفهمه نظام الحاسوب . وبالتالي لإخراجه لابد من تحويله إلى الشكل الذي يستطيع مستثمر الحاسوب أن يفهمه ويتعامل معه وبالتالي لا بد له من المرور عبر وحدة التحكم ومن ثم إلى **جهاز الخرج Output Device** المرغوب.

إذن ، يمكن أن نلخص العمليات الأساسية التي يقوم بها الحاسوب بما يلي :

1. استقبال البيانات والأوامر .
2. تخزين البيانات والأوامر .
3. معالجة البيانات .
4. إخراج المعلومات .

بالنظر إلى مبدأ عمل الحاسوب من جهة وإلى مفهوم البرنامج نجد أن هناك تشابها كبيراً، **فالبرنامج هو عبارة عن سلسلة من التعليمات والأوامر التي تقوم بمعالجة بيانات مدخلة من أجل إعطاء نتائج واستجابات على الخرج.**

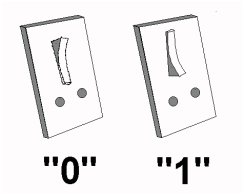
وبالتالي فالمكونات الأساسية لأي برنامج هي أربعة: الدخل، المعالجة، التخزين، الخرج. وهذا يقتضي أن أي لغة برمجة يجب أن تتضمن أربع أنواع من التعليمات:

1- تعليمات الإدخال.
2- تعليمات الإخراج.
3- تعليمات التخزين في الذاكرة.
4- تعليمات الحساب والمعالجة.

4- تمثيل المعطيات في نظام الحاسوب

بما أن الحاسوب هو عبارة عن مجموعة من الدارات الإلكترونية وهذه الدارات تمر بأحد وضعين ممكنين، فالدارة إما أن تكون مفتوحة Open، أو مغلقة Closed، أي إما أن تكون بحالة تشغيل On أو توقف Off .

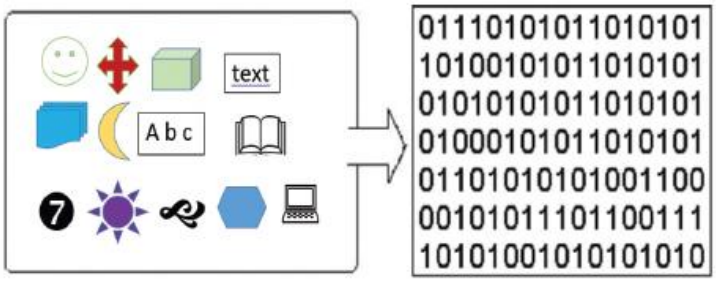
وبالتالي يمكن تمثيل وضعية هذه الدارات باستخدام نظام تمثيل أساسي هو النظام الثنائي Binary System بكلام آخر ، يشار إلى هاتين الوضعيتين إما بـ " 0 " وإما بـ " 1 " . يطلق على الوضعية – وفق مصطلحات الحاسوب – التسمية " بِت Bit " وهي مشتقة من العبارة Binary Digit.



الشكل 2- مفتاح دارة إلكترونية

تقوم الحواسيب في كل معالجاتها وحساباتها بتمثيل البرامج والمعطيات على شكل بتات وبالتالي يمكن اعتبار هذين الرمزين يشكلان اللغة الأم للحاسوب Computer native Toungue .

تنظم المعطيات في سلاسل ذات طول ثابت تسمى كلمات Words وهي تمثل الوحدة الأساسية في تخزين ومعالجة المعلومات في الحاسوب، ويحدد طول الكلمة بحسب المكونات المادية للحاسوب.



الشكل 3- تمثيل البيانات المدخلة ضمن نظام الحاسوب.

في بداية الحاسوب اعتمدت كلمة بطول 6 خانات ثنائية (بتات) وهي تعطي 64 رقماً مختلفاً فكانت كافية لتمثيل 26 حرفاً من الحروف الإنكليزية و 10 أرقام عشرية إضافة إلى علامات الترقيم والرموز الحسابية. حديثاً تستخدم كلمة من ثمانية خانات ثنائية لتشكيل ما يسمى بالبايت Byte.

بالطبع لا يتكلم الناس هذه اللغة الثنائية. بل يتواصلون بلغة طبيعية منطوقة ومسموعة ومرئية وبالتالي لكي يتفاعلوا مع الحاسوب يجب أن تترجم مخاطبتهم له إلى الشكل الثنائي والعكس صحيح .

إن عملية الترجمة هذه تدعى عملية **الترميز Coding وتعني تحويل المعلومات المراد معالجتها إلى كلمات مؤلفة من الخانات الثنائية.** ومع أن الترميز يجري على كلمة من 8 خانات ثنائية، فإن الحواسيب تستخدم كلمات أطوالها من مضاعفات العدد 8 مثل 8، 16، 32، 64 .

لنحاول أن نتخيل كيف يتصرف النظام الحاسوبي عند الضغط على المحرف 'A' على لوحة المفاتيح؟. يبين الشكل 4 الخطوات التي تحصل بدءاً من الضغط على المحرف "A" وانتهاء بعرضه على الشاشة.

الشكل 4– تمثيل المحرف "A" ضمن النظام الحاسوبي.

يختلف تمثيل المعطيات والبرامج وفقاً لطبيعة المعلومة المراد تمثيلها. فالبيانات النصية Text Data (كالأرقام و المحارف والرموز الخاصة) يتم تمثيلها باستخدام أنظمة ترميز Coding Systems معتمدة على المفهوم الثنائي ونظم العد، ومن أمثلتها ASCII، UNICODE. بينما البرامج تمثل باستخدام شيفرة (Code) تدعى لغة الآلة Machine Language. أما البيانات غير النصية Non-Text Data (كالصور والصوت والفديو) فتمثل بطرق أخرى مختلفة.

5- لغات البرمجة

تتوافر عالمياً أعداد كبيرة من لغات البرمجة التي تستخدم في حل المسائل بواسطة الحاسوب، حيث تندرج جميع هذه اللغات تحت أحد الأنواع التالية:

1- لغات الآلة Machine Languages.
2- لغات المجمع Assembler Languages.
3- اللغات عالية المستوى High Level Languages.

يقصد **بلغة الآلة Machine Language** بأنها اللغة المعرفة من قبل البنية الصلبة للحاسوب، وهي تختلف باختلاف البنيان المادي للآلة.
تتألف لغة الآلة من سلاسل من الأعداد (مجموعة من الأصفار والواحدات) التي تعطي الأوامر للحاسوب من أجل تنفيذ تعليماته الأولية كلاً على حدى.
إن كتابة البرامج بلغة الآلة يعتبر أمراً صعباً ومرهقاً وبالتالي فهي نادراً ما تستخدم بشكل مباشر.

ظهرت **لغات المجمع Assembler Languages** من أجل تجاوز الصعوبة التي تجلت في لغات الآلة، حيث اعتمدت على استخدام مصطلحات قريبة من اللغة الإنكليزية للتعبير عن العمليات الأولية للحاسوب، وقد استخدمت مترجمات للبرامج تدعى مجمعات Assemblers من أجل تحويل هذه البرامج من لغة المجمع إلى لغة الآلة.

على الرغم من تطور لغات المجمع مقارنة بلغات الآلة إلا أن إنجاز أبسط المسائل مازال يتطلب استخدام العديد من التعليمات. وبالتالي من أجل تسريع عملية البرمجة تم تطوير **لغات عالية المستوى High Level Languages**، وبشكل مماثل للغات المجمع فقد استخدمت برامج لتحويل البرامج المكتوبة بلغة عالية المستوى إلى لغة الآلة، وتم تسمية هذه البرامج باسم المترجمات Compilers أو المفسرات interpreters.
إن اللغات عالية المستوى أكثر تقبلاً من قبل المبرمجين من لغات الآلة أو المجمع لذلك فقد انصبت معظم جهود التطوير في السنوات الأخيرة على هذا النوع من اللغات.

	Sample code	Translator	From the programmer's perspective	From the computer's perspective
Machine language	+1300042774 +1400593419 +1200274027	None	Slow, tedious, error prone	Natural lanugage of a computer; the only language the computer can understand directly
Assembly language	LOAD BASEPAY ADD OVERPAY STORE GROSSPAY	Assembler	English-like abbreviations, easier to understand	Assemblers convert assembly language into machine language so the computer can understand
High-level language	grossPay = basePay + overTimePay	Compiler	Instructions resemble everyday English; single statements accomplish substantial tasks	Compilers convert high-level languages into machine language so the computer can understand

الشكل 5- مقارنة بين أنواع لغات البرمجة.

6- مرحل تطور لغات البرمجة

لقد مر تطوير لغات البرمجة عالية المستوى بأربع مراحل مفصلية:

المرحلة الأولى: مرحلة البرمجة التسلسلية Sequential Programming وفيها كان البرنامج عبارة عن سلسلة متعاقبة من التعليمات والأوامر التي يجب أن تنفذ جميعها بتتالي قسري ومتكرر، إلا أن تزايد تعقيد المسائل جعل هذا الأسلوب ينتج برامج طويلة ومعقدة وصعبة الفهم والتتبع.

المرحلة الثانية: مرحلة البرمجة الإجرائية Procedural Programming وفيها يتم الاعتماد على تجزئة المشكلة التي يعالجها البرنامج إلى عدة مشاكل جزئية، ومن ثم كتابة برنامج فرعي لكل جزء من هذه الأجزاء.

تمتعت البرامج الإجرائية بحسنات جمة على البرامج التسلسلية أهمها قصر طول البرامج وسهولة قراءتها وصيانتها، إضافة إلى قابلية إعادة استخدام الشيفرة في أكثر من موضع من البرنامج.

المرحلة الثالثة: مرحلة البرمجة غرضية التوجه Object Oriented Programming وفيها تم تطوير الاعتماد مفهومي الاعتماد على البرامج الفرعية وقابلية إعادة استخدام الشيفرة بالإضافة إلى مفهوم إضافي وهو أمن البيانات من خلال الإعتماد على الأغراض والأصناف Classes and Objects مع ماتضمنه ذلك من مفاهيم إضافية كالتغليف Encapsulation والوراثة Inheritance وتعدد الأشكال Polymorphism ... وغيرها من المفاهيم.
أشهر اللغات التي اعتمدت هذا المفهوم هي C++، Java، C# ... وغيرها.

المرحلة الرابعة: مرحلة البرمجة المستقلة عن بيئة التشغيل Platform independent حيث اعتمدت البرمجة ما قبل هذه المرحلة على كتابة برامج مرتبطة ببيئة التشغيل أي أنها غير قابلة للتنفيذ إلا في نفس البيئة التي كتبت فيها وذلك لأن المترجمات المستخدمة تعتمد تحويل البرنامج إلى لغة الآلة، في حين أن البرمجة في هذه المرحلة اعتمدت استخدام مترجمات تقوم بتحويل البرنامج إلى ملف من النوع Bytecode ومن يتم اعتماد آلات افتراضية Virtual Machines في التنفيذ تقوم بتحويل هذا الملف الثنائي إلى لغة الآلة التي تتناسب مع بيئة التشغيل التي يتم التنفيذ عليها.

من أشهر الآلات الافتراضية، آلة جافا الإفتراضية Java Virtual Machine (JVM) الخاصة بلغة الجافا، وإطار العمل دوت نيت Net Framwork. الخاصة بلغات أخرى مثل C#، Visual Basic.net، ... وغيرها.

7- المترجم والمفسر

رأينا من الفقرات السابقة أن تنفيذ البرامج المكتوبة باستخدام لغة برمجية عالية المستوى يتطلب وجود برنامج يقوم باختبار هذه البرامج والتأكد من خلوها من الأخطاء ومن ثم تحويلها إلى لغة الآلة وتنفيذها، هذا البرنامج يدعى **المترجم Compiler أو المفسر Interpreter**.

على الرغم من أن المترجم والمفسر كلاهما يؤدي الدور ذاته إلا أن هناك اختلافاً في أسلوب أداء هذا الدور:

يقوم المترجم Compiler باختبار كامل البرنامج والتأكد من صحته وسلامته من الأخطاء، ومن ثم ترجمة هذا البرنامج إلى لغة الآلة فينتج ملف ثنائي يتضمن الأوامر والتعليمات الخاصة بالبرنامج على شكل أصفار وواحدات ومن ثم تتولى بيئة التشغيل تنفيذ البرنامج المطلوب انطلاقاً من هذا الملف الثنائي.
من حسنات هذا الأسلوب أن البرنامج يترجم مرة واحدة وينفذ عدة مرات إذ أنه لاداعي لترجمته في كل مرة نرغب بتنفيذه، كما أن تنفيذ البرنامج المترجم لا يتطلب وجود المترجم ضمن بيئة التشغيل التي ينفذ فيها. إلا أن من مساوئه هو أنه في حال كان البرنامج ضخماً فإن تنفيذه يكون بطيئاً كونه لا يبدأ إلا بعد الانتهاء من الترجمة والتنقيح.

يقوم المفسر Interpreter باختبار البرنامج سطراً سطراً وكل سطر يتم التأكد من سلامته وترجمته إلى لغة الآلة وتنفيذه ومن ثم الانتقال إلى الذي يليه.

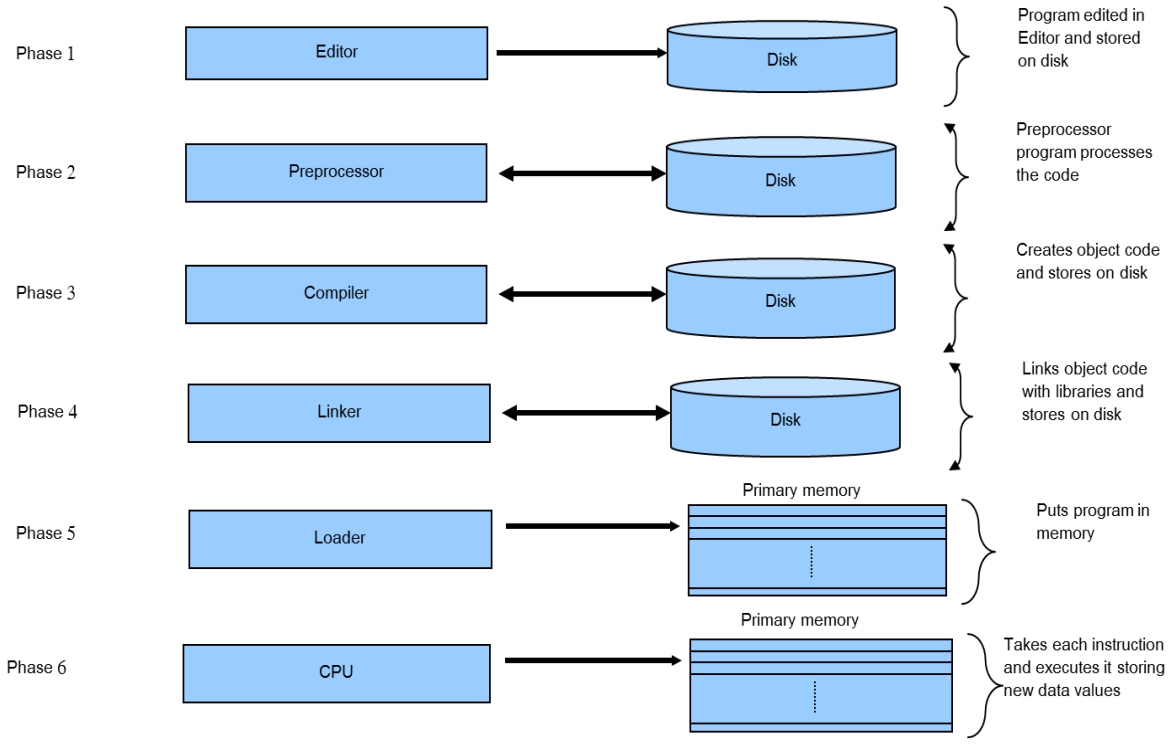
من حسنات هذا الأسلوب هو السرعة في بدء تنفيذ البرامج الضخمة، إلا أن مساوئه تظهر عند الحاجة لتنفيذ البرنامج أكثر من مرة، إذ أن عملية الترجمة والاختبار تتم في كل مرة، كما أن تنفيذ البرنامج يتطلب وجودة المفسر ضمن بيئة التشغيل التي ينفذ فيها.

اعتمدت لغات البرمجة المستقلة عن بيئة التشغيل مثل Java، C# ... وغيرها أسلوباً مزيجاً بين الأسلوبين فالمترجم يختبر وينقح البرنامج ويحوله إلى الترميز المستقل عن الآلة Bytecode، والآلة الافتراضية تلعب دور المفسر لملف الـ Bytecode.

8- بيئة تطوير البرنامج المثالية

يمر تطوير البرنامج بـ 6 مراحل أساسية:

- **التحرير Edit:**
  - يقوم المبرمج بكتابة البرنامج ويمكن أن يتم ذلك باستخدام أي محرر نصوص (تحوي بيئات التطوير المتكاملة IDE's مثل Visual Studio محرراً خاصاً بها مدمجاً في البيئة)، ومن ثم يتم تخزينه على أقراص التخزين.
- **ما قبل المعالجة Preprocessing:**
  - ينفذ أوامر خاصة تسمى توجيهات ماقبل المعالجة تحدد هذه التوجيهات جملة من المعالجات والإجراءات الواجب تنفيذها على نص البرنامج قبل ترجمته كتضمين المكتبات، تعريف ثوابت، تعريف أنماط البيانات، .. إلخ.
- **الترجمة Compile:**
  - يقوم المترجم بتوليد شيفرة الكائنات أو الشيفرة الثنائية لأوامر البرنامج (object code أو Byte code بحسب اللغة).
- **الربط Link:**
  - الربط بين أكواد الكائن والتعليمات البرمجية للوظائف لإنتاج نسخة قابلة للتنفيذ.
- **التحميل Load:**
  - تخزين أكواد البرنامج وملفه التنفيذي في الذاكرة الرئيسية.
- **التنفيذ Execute:**
  - تحويل شيفرة الكائنات أو الشيفرة الثنائية إلى لغة الآلة وتنفيذها.



الشكل 6– مراحل تطوير البرنامج.

من أهم أنواع الأخطاء التي يمكن أن تحصل في هذه المرحلة:

1– **أخطاء في الصيغة Syntax Errors**: يتم كشفها من قبل المترجم.

2– **أخطاء وقت التنفيذ Runtime Errors**: تتسبب في إحباط البرنامج وإيقاف تنفيذه.

3– **الأخطاء المنطقية Logic Errors**: تسبب في إعطاء نتائج غير صحيحة.

9- أنشطة وتمارين

النشاط الأول

أوضح معنى المفاهيم التالية:

الحاسوب – البيانات – المعلومات – معالجة البيانات – البرنامج – لغة البرمجة – ترميز المعلومات – لغة الآلة –

النشاط الثاني

أجب بصح أو خطأ على العبارات التالية:

1. تستطيع الحواسيب تنفيذ أي نوع من البرامج مهما كانت اللغة التي تكتب بها سواء لغة برمجية أو لغة إنكليزية متداولة.
2. لا تتعامل العتاديات المادية للحاسوب إلا مع المعلومات والأوامر المقدمة بشكل ثنائي.
3. تتولى وحدة التحكم في النظام الحاسوبي التقاط المعطيات المقدمة من أجهزة الدخل وتحويلها إلى الشكل الذي يفهمه الحاسوب ويستطيع التعامل معه.
4. لا تتعامل وحدة المعالجة المركزية بشكل مباشر إلا مع الأوامر والبيانات المخزنة في الذاكرة الرئيسية.
5. لجميع أنواع الحواسيب مهما اختلفت بنيتها الداخلية لغة الآلة نفسها.
6. يقوم المفسر Interpreter باختبار كامل البرنامج وترجمته إلى لغة الآلة قبل تنفيذه.
7. يقوم المترجم compiler باختبار كامل البرنامج وترجمته إلى لغة الآلة قبل تنفيذه.
8. لدى استخدام المفسر تتم ترجمة البرنامج لمرة واحدة وتنفيذه عدة مرات دون الحاجة لترجمته من جديد.

النشاط الثالث

إختر الإجابة الصحيحة فيما يأتي:

1- لغة البرمجة C# هي من:

a. اللغات عالية المستوى.
b. لغات المجمع.
c. لغات الآلة.
d. جميع الأجوبة السابقة خاطئة.

2- الأمر التالي z = x+y هو من الأوامر المكتوبة بـ :

a. لغة برمجية عالية المستوى.
b. لغة المجمع.
c. لغة الآلة.
d. جميع الأجوبة السابقة صحيحة.

3- البرنامج في لغة البرمجة الإجرائية يكون مكوناً من:

a. سلسلة متتالية من التعليمات والأوامر التي تنفذ جميعها بشكل متتالي وقسري.
b. مجموعة من الأوامر التي تستخدم أغراضاً من أصناف معرفة من قبل المستخدم.
c. مجموعة من البرامج الجزئية يتم استدعاؤها حسب الحاجة.
d. جميع الأجوبة السابقة خاطئة.

4- قمت بإصدار أمر تنفيذ لبرنامج فظهرت رسالة خطأ تفيد بأن متحولاً ما غير موجود، يصنف هذا الخطأ على أنه:

a. خطأ صيغة syntax error.
b. خطأ تنفيذ runtime error.
c. خطأ منطقي logic error.
d. جميع الأجوبة خاطئة.

5- قمت بإصدار أمر تنفيذ لبرنامج فقام بالتنفيذ من أجل بعض المدخلات وفشل في التنفيذ من أجل مدخلات أخرى وأظهرت رسالة خطأ، يصنف هذا الخطأ على أنه:

a. خطأ صيغة syntax error.
b. خطأ تنفيذ runtime error.
c. خطأ منطقي logic error.
d. جميع الأجوبة خاطئة.

6- قمت بإصدار أمر تنفيذ لبرنامج فعمل البرنامج من دون إظهار أي رسالة خطأ ولكنه أعطى على خرجة قيماً غير متوقعة، يصنف هذا الخطأ على أنه:

a. خطأ صيغة syntax error.
b. خطأ تنفيذ runtime error.
c. خطأ منطقي logic error.
d. جميع الأجوبة خاطئة.