كلية التربية النوعية
قسم تكنولوجيا التعليم والحاسب الآلي

محاضرات في لغات الحاسب الآلي

الفرقة الثالثة

إعداد

قسم تكنولوجيا التعليم والحاسب الآلي

كلية التربية النوعية – جامعة المنوفية

٢٠١٩ / ٢٠٢٠ م

نموذج رقم (١٢)

جامعة / أكاديمية : جامعة المنوفية.

كلية / معهد : كلية التربية النوعية.

قسم : .تكنولوجيا التعليم والحاسب الالى .

توصيف مقرر دارسي: لغات الحاسب الالى

١- بيانات المقرر:		
الرمز الكودى :ET(32) 8	**اسم المقرر : لغات الحاسب الالى**	**الفرقة / المستوى : مرحلة البكالوريوس - الفرقة الثالثة – الفصل الدراسي الثاني**
التخصص : تكنولوجيا التعليم والحاسب الالى	**عدد الوحدات الدراسية : نظرى عملى** ٢ ٢	

٢- هدف المقرر :	**الاهداف العامة للمقرر :** • يعرف الطالب لغة السى ++ وكيفية استخدامها. • تحديد انواع لغات الحاسب الالى . • التعرف على بعض التطبيقات فى لغة السي ++. **- بنهاية هذا المقرر يكون الطالب قادر علي :** • ثقافه استيعاب التنكولوجيات الحديثه فى لغات الحاسب الالى . • تحليل لغات الحاسب الالى ومكوناتها • تبسيط بعض التطبيقات فى لغة السي ++.
٣- المستهدف من تدريس المقرر :طلاب الفرقة الثالثه تكنولوجيا التعليم والحاسب الالى	
أ- المعلومات والمفاهيم	أ-١يعرف المتغيرات وانواع المتغيرات والاعلان عن المتغير. أ- ٢ يذكراساسيات لغات الحاسب الالى وانواعها والفرق بين هذه الانواع وكيفية استخدام كل لغة .

ب- المهارات الذهنية:	ب-١يحلل جمل التكرار والتحديد والجمل الشرطيه فى لغة c++ وباقي اللغات
ج- المهارات المهنية الخاصة بالمقرر :	ت–١ يتدرب على برنامج ال c++ من حيث نافذة المشروع والخصائص والنموذج والادوات. ت–٢ تصميم برنامج للرسم بلغة c++
د- المهارات العامة :	ث-١ يوعي باهمية لغات المستوي العالي المختلفة والفروق بينها ث-٢ يشارك في العمل كفريق في انتاج برنامج بلغة c++. ث-٣ يتواصل عن طريق بعض التطبيقات العامة بلغة c++.
٤- محتوى المقرر :	١- مقدمة عن لغات الحاسب الالى وانواعها والفرق بين هذه الانواع وكيفية استخدام كل لغة (١). ٢- مقدمة عن لغات الحاسب الالى وانواعها والفرق بين الانواع وكيفية استخدام كل لغه (٢). ٣- دراسة برنامج ال c++ من حيث نافذة المشروع والخصائص والنموذج والادوات وشرح كل مكون بالتفصيل وخطوات وضع البرامج (١). ٤- دراسة برنامج ال c++ من حيث نافذة المشروع والخصائص والنموذج والادوات وشرح كل مكون بالتفصيل وخطوات وضع البرامج (٢). ٥- شرح المتغيرات وانواع المتغيرات والاعلان عن المتغير ، جملة التكرار for وجملة التكرار Do وجملة التحديد select. والجمل الشرطيه الاخرى وامثله عليهم(١). ٦- شرح المتغيرات وانواع المتغيرات والاعلان عن المتغير ، جملة التكرار for وجملة التكرار Do وجملة التحديد select. والجمل الشرطيه الاخرى وامثله عليهم(٢). ٧- تطبيقات فى لغة ال c++ لتصميم برامج حسابيه .(١) ٨- تطبيقات فى لغة ال c++ لتصميم برامج حسابيه .(٢) ٩- انشاء قواعد البيانات وربطها بالبرامج المنتجه بواسطة لغة c++(١). ١٠- انشاء قواعد البيانات وربطها بالبرامج المنتجه بواسطة لغة c++(٢). ١١- انشاء برامج للرسم بلغة ال c++.(١) ١٢- انشاء برامج للرسم بلغة ال c++.(٢) ١٣- تطبيقات عامة بلغة ال c++(١) ١٤- تطبيقات عامة بلغة ال c++(٢)

٥- أساليب التعليم والتعلم	١- محاضرات ✓ عملى / معملى ✓ ٢- جلسات مناقشة ✓ / ورشة عمل ✓ ٣- أنشطة فى الفصل (السكشن) ✓ ٤- واجبات منزلية . ✓ ٥- تدريب ٦ - ندوة ٧- دراسة الحالة
٦- أساليب التعليم والتعلم للطلاب ذوى القدارة المحدودة :	**لا يوجد طلاب من ذوي الاحتياجات كسمة من سمات الكلية والمجال**
٧- تقويم الطلاب :	
أ- الأساليب المستخدمة	١. **امتحان نصف الفصل :** لقياس المقدرة علي التركيز وفهم الجوانب والخلفيات العلمية. ٢. **امتحان أخر الفصل :** لقياس مهارات التذكر والإبداع . ٣. **الشفهي :** لقياس مهارات التحليل والعرض والمناقشة . ٤. **التطبيقى :** لقياس مهارات الممارسة والتطبيق والمهارات الفنية والعملية . ٥. **أعمال الفصل:** لقياس مهارات حل المشكلة وتقديم البيانات والمناقشة وقياس المقدرة على العمل فى جماعة .
ب- التوقيت	**- التقييم ١ امتحان نصف الفصل : يتم فى الأسبوع السادس.** **- التقييم ٢ امتحان أخر الفصل : الأسبوع الخامس عشر.** **- التقييم ٣ الشفهى: الأسبوع الرابع عشر.** **- التقييم ٤ التطبيقى : الأسبوع الثالث عشر.** **- التقييم ٥ أعمال الفصل : الأسبوع الخامس والعاشر.**
ج- توزيع الدرجات	**- امتحان نصف الفصل : ١٠ درجات.** **- امتحان أخر الفصل : ٩٠ درجة.** **- الشفهى: ١٠ درجة.** **- التطبيقى : ٣٠ درجة.** **- أعمال الفصل : ١٠ درجة.** **- المجموع: ١٥٠ درجة.**
٨- قائمة الكتب الدراسية والمراجع :	
أ- مذكرات	مذكره مادة لغات الحاسب الالى اعداد قسم تكنولوجيا التعليم والحاسب الالى
ب- كتب ملزمة	كتاب مادة لغات الحاسب الالى قسم تكنولوجيا التعليم والحاسب الالى
ج- كتب مقترحة	

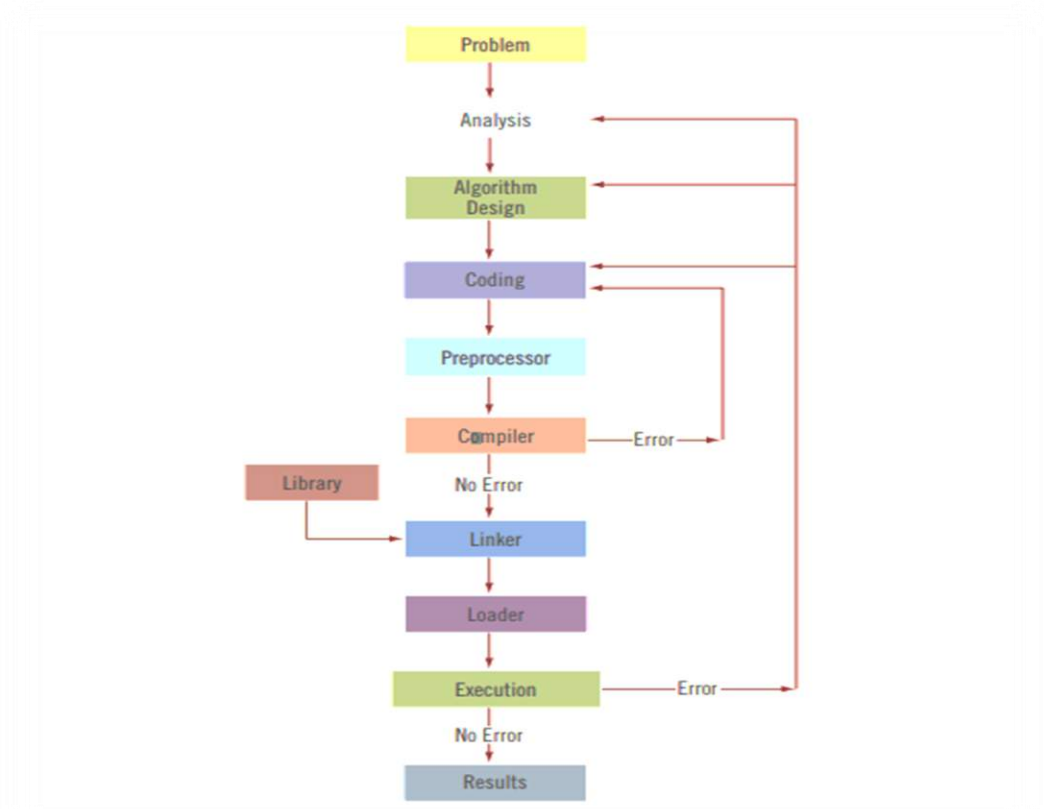
لغات الحاسب الآلي

خطوات وضع البرامج:

البرمجة فن فالكثير يعتقدون أن المبرمج محتاج أن يكون متخصصًا في الرياضيات، وذو عقلية مؤهلة للتعامل مع الأرقام والمعلومات، ومستعد لأن يظل ساعات طوال جالسًا إلى جهازه، يكتب البرامج ويصححها، إلا أنه في الواقع لو أن المبرمج أمد بالوسائل المناسبة، وأسلوبًا منهجيًا يسير عليه، فإن البرمجة تصبح متاحة للجميع. إنها مهمة يجب تنفيذها، ولكنها في نفس الوقت تحمل الكثير من التحفيز والشعور بالرضا عن النفس.

يمكن تعريف عملية البرمجة على أنها وضع تصور عام لتنفيذ مهمة تم توصيفها، ثم وضع الخطوات التنفيذية لهذا الحل على صورة تعليمات متتالية حينما توجه لمكونات الحاسوب المادية تؤدي للنتيجة المطلوبة. ويتمثل الجانب الإبداعي الخلاق من البرمجة في أول جزء من التعريف، ألا وهو وضع التصور العام للحلول للمشاكل التي يتم مواجهتها. ومع ذلك فغالبا ما يتم تجاهل هذه الخطوة، بالقفز فورًا إلى مرحلة صياغة البرنامج دون وضع تصميم واضح لما يكون عليه الحل، وغلبا ما يترتب على ذلك الكثير من الأخطاء المنطقية Logical errors التي تتطلب جهدا كبيرًا في اكتشافها وتصحيحها، أما المبرمج الأكثر خبرة فيضع في البداية تصميما للحل، ثم يختبره مكتبيًا وعند الاطمئنان على صحته يترجمه إلى أوامر برمجية باللغة التي اختارها.

وتتمثل خطوات وضع الحل لمشكلة ما في سبع نقاط نعرض إطارها العام فيما يلي:



١-حل المشكلة:

تتطلب هذه الخطوة قراءة متأنية للمشكلة إلى أن يتم استيعابها تماما ومعرفة المطلوب لكل وضوح، وحتى يمكن القيام بهذه الخطوة الأولية بنجاح، تقسم المسألة إلى:

*المدخلات.

*المخرجات.

*الخطوات المطلوبة للوصول إلى المخرجات.

ونحبذ في ذلك الاستعانة بجدول تحليلي للتعرف على المشكلة كما سنقدمه في الفصل الثالث، إذ يساعد ذلك على فصل هذه العناصر الثلاثة والتعرف عليها.

٢- وضع تصورًا عامًا للحل:

بمجرد تحديد المشكلة بوضوح، قد ترى تجزئتها إلى خطوات أو مهام أبسط، ونصبح بذلك تصورًا عاما للحل. هذا التصور المبدئي هو عادة مجرد خطوات عامة للحل يمكن أن تحتوي على في حالة البرامج الحاسوبية على:

*خطوات المعالجة الأساسية.

*المهام الفرعية الرئيسية (إن وجدت).

*الهياكل الرئيسية للتحكم في سير التنفيذ (كالدوارات loops أو القرارات Decisions).

* المتغيرات الأساسية وهياكل البيانات.

*المنطق العام للبرنامج.

٣- ضع التصور العام على هيئة خوارزم.

يمكن تطوير التصور العام الذي وضع في الخطوة السابقة إلى خوارزم، أي مجموعة الخطوات المحددة التي تصف بدقة المهام المنفذة وترتيب تنفيذها، ويستخدم هذا الكتاب أسلوب اللغة شبه برمجية Pseudocode، تتميز بكونها قريبة من اللغة المهيكلة (متفق عليه لوصف النظم خلال عملية تحليل وتصميم النظم، راجع كتاب " تحليل وتصميم نظم المعلومات" ترجمة المهندس علي يوسف علي وإصدار دار خوارزم – المترجم Structured language لتمثل الحل، وأيضا لتمثيل تكنيكات البرمجة المهيكلة Structured programming techniques كما نقدم أيضا أشكال ناسى- شنايدر لمن يفضلون التمثيل التصويري للخوارزميات. وسوف نناقش باستفاضة في الفصول من ٢إلى ٦ استخدام اللغة شبه البرمجية و"نظرية الهيكلة Structure Theorem" في وضع الخوارزمات.

٤-اختبر صلاحية الخوارزم

هذه الخطوة من أهم الخطوات في وضع البرامج ورغم ذلك فكثيرا ما يتم تجاهلها وأهمية هذه الخطوة هي اكتشاف الأخطاء المنطقية في الخوارزمات مبكرا وتستخدم بيانات اختبارية في عملية التشغيل التجريبي لتسير مع الخوارزم خطوة فخطوة، للتأكد من أن التعليمات الخاصة بالخطوات ستؤدي بالفعل لما هو مطلوب وهنا يسير المبرمج مع خطوات الخوارزم، بالضبط كما سيفعل الحاسب عند التنفيذ، مسجلا التغير في قيم المتغيرات الأساسية. ويقدم الفصل الثالث جداول وخوارزمات اختبار مقترحة وعديد من الأمثلة التوضيحية.

٥-ضع الخوارزم بلغة برمجية:

وليس لك أن تنشط لصياغة برنامج في لغته البرمجية إلا بعد الوفاء بكل متطلبات التصميم كما عرضناها في الخطوات الأربع السابقة.

٦- شغل برنامجك على الحاسوب:

هذه الخطوة الحاسمة، والمثيرة في نفس الوقت، وبافتراض أنك قد بذلت العناية الكافية في التصميم، فإن ساعات الإحباط لوجود الأخطاء سوف تنكمش لأقل مستواها: وقد يتطلب الأمر تكرار هذه الخطوة عدة مرات قبل الوصول للرضا الكامل عن البرنامج.

٧- وثق البرنامج:

ليس من اللائق أن توضع خطوة توثيق البرنامج كخطوة ألأخيرة في تتابع وضع البرامج، حيث إنها في الواقع إجراء يسير مع كافة الخطوات السابقة.

والتوثيق نوعان، خارجي؛ يتمثل في الخرائط الهيكلية، وخوارزمات الحلول، ونتائج الاختبارات، وداخلي؛ وهو ما يضعه المبرمج في صياغة الكود نفسه.

تطوير البرنامج:

بعد أن ينطلق البرنامج في التطبيق العملي، تثور الحاجة عادة إلى إجراء تعديلات عليه مع مرور الزمن، وغالبا ما يكون ذلك من قبل مبرمجين لم يشتركوا في وضع الكود الأصلي. ومن ثم فإنه في حالة البرامج بذلت فيها العناية الواجبة في التصميم والتوثيق، تكون إجراءات تطوير البرنامج سهلة تلقائية.

البرمجة المهيكلة:

البرمجة المهيكلة تساعدك على كتابة برامج على أعلى مستوى من الكفاءة، خالية بقدر الإمكان من الأخطاء المنطقية، وقد ظهر مفهوم البرمجة المهيكلة لأول مرة في ورقة وضعها Jacopini , Bohm في إيطاليا عام ١٩٦٤، حيث قدما فكرة تصميم البرامج باستخدام نظرية الهيكلة Structure Theory، مؤسسة على ثلاثة هياكل حاكمة Control Structures، ومنذ ذلك الحين تطور المفهوم على يد كثيرين مثل

Edsger Dijkstra , Niklaus Wirth, Ed Yourdon, Michael Jackson إلى أن تبلور في الصورة الشائعة لمصطلح" البرمجة المهيكلة" هذا المصطلح لم يعد مقصورا على نظرية الهيكلة ذاتها، بل اتسع ليشمل معها التصميم من أعلى إلى أسفل، والتصميم البنائي modular design أو Modularization

أسلوب من أعلى إلى أسفل في وضع البرامج:

كان التقليد الشائع هو أن يبدأ المبرمج بمجرد أن يتعرف على المشكلة في صياغة البرنامج، بادئا من بداية المشكلة متدرجا إلى نهايتها. وغالبا ما يجد نفسه قد انشغل بتفاصيل الخطوات، بدلا من أن يتفرغ لوضع تصور لحل شامل للمشكلة. وفي أسلوب التصميم من أعلى إلى أسفل، يكون البدء بوضع تصور شامل للحل، والذي يتفرع تدريجيا إلى خطوات فرعية قد تتفرع بدورها إلى أن نصل إلى آخر مستوى من التفاصيل. معنى ذلك أن البدء في الصياغة الفعلية للبرنامج لا تكون إلا بعد إجراء هذه العملية التي يطلق عليها" التحليل الوظيفي Functional decomposition" ونتيجة لهذا الأسلوب المنهجي المنظم نحصل على تصميم على أعلى درجة من الدقة لم يكن متاحا من قبل.

التصميم البنائي:

تتضمن البرمجة المهيكلة أيضا فكرة التصميم البنائي، والذي يعني جمع المهام في وحدات طبقا لمنطق معين (مثلا، تهدف لمهمة معينة كحساب الضريبة أو طباعة وثيقة) والتصميم البنائي مرتبط ارتباطًا وثيقًا بأسلوب الوضع من أعلى إلى أسفل، حيث إن التدرج المشار إليه يتكشف عنه المهام الجزئية، والتي تأخذ شكل الوحدات البنائية modules ويساعد التصميم البنائي الجيد على فهم البرنامج وتتبعه.

نظرية الهيكلة:

تعتبر نظرية الهيكلة ثورة في أسلوب البرمجة، بدأت بإلغاء عبارة GOTO الشهيرة، ووضع إطار هيكلي للحل. وتنص النظرية على إمكانية وضع أي برنامج حاسوبي باستخدام ثلاثة هياكل حاكمة أولية، ألا وهي:

*الأوامر المتتابعة sequences

*القرارات (تسمى أيضا: العبارات الشرطية Conditional Statemets والاختيار selection)

*الدوارات loops

وسوف يغطي هذا في الفصل الثاني

مقدمة عن الخوارزم واللغة شبه البرمجية

تتطلب تكنيكات البرمجة المهيكلة أن يكون البرنامج قد تم تصميمه جيدا قبل البدء في كتابته، وعملية التصميم هذه هي التي يتولد عنها الخوارزم.

ما هو الخوارزم؟

يشبه الخوارزم (وصفات) الطهي، فهي تضع الخطوات لتنفيذ مهمة ما [طهي طبق ما – المخرجات، بتحديد المواد المطلوبة ومقاديرها – المدخلات، وعمليات الطهي– المعالجة]. هذه الوصفات يمكن أن توضع بمصطلحات البرمجة لبيان كيفية الوصول إلى مخرجات معينة من مدخلات معينة. ويكتب الخوارزم بلغة بسيطة واضحة. ويجب الالتزام بمبادئ معينة حتى يكون الخوارزم فعالا:

*أن يكون سلسا، دقيقا، لا يشوبه الغموض.

* يغطي الحل الذي يغطي كافة الاحتمالات.

* أن يصل لنهاية محددة.

فعلى سبيل المثال، لو أردت أن تعطي شخصا ما تعليمات لإضافة بعض الأرقام باستخدام الآلة الحاسبة، فقد تضع ذلك على شكل الخوارزم التالي:

*شغل الآلة

*امح أية بيانات موجودة.

*كرر التعليمات التالية:

–أدخل الجزء الصحيح

– اضرب على العلامة العشرية.

– أدخل الجزء الكسري.

– اضرب على علامة +

إلى أن تنتهي من كافة الأرقام

*اكتب الناتج النهائي.

– اطفئ الآلة.

ونلاحظ أنه في هذا الخوارزم تنفذ الخطوتان الأوليان مرة واحدة، قبل دخول العملية التكرارية لإدخال الأرقام، كما تنفذ الخطوتان الأخيرتان أيضا مرة واحدة.

هذا الخوارزم يفي بكل المتطلبات لخوارزم فعال: فهو يورد كافة الخطوات في ترتيبها الصحيح، وبأسلوب غير غامض أو مبهم، إلى أن نصل إلى النهاية المنشودة. كما تلاحظ أن الخطوات التكرارية قد أزيحت للداخل، لتأخذ وضعا مستقلا يظهر طبيعتها الخاصة. وتعتبر الإزاحة للداخل من الوسائل الهامة للغاية في توضيح البرنامج وتوثيقه.

ما هي اللغة شبه البرمجية ؟

كان استعمال مخططات التدفق flow charts من الوسائل الأكثر شيوعا في تصوير الخوارزمات، ولكنها كانت معقدة، وغامضة، وصعبة الفهم والتتبع، وتؤدي بالتالي في كثير من الأحيان إلى أكواد مليئة بالأخطاء. وفي المقابل، تسمح اللغة شبه البرمجية بقراءة خطوات الخوارزم باللغة الطبيعية المألوفة، إلا أنها وضعت على هيئة تجعلها تشابه اللغات البرمجية الراقية.

وكثيرا ما كان المبرمجون يضعون خوارزمات برامجهم بلغة أقرب للغة التي سوف يستخدمونها في وضع البرامج، وهذا الكتاب يحاول وضع لغة شبه البرمجية لكتابة الخوارزمات، لا تتعلق بلغة برمجية معينة، بل هي منبثقة من التقاليد العامة للغات المهيكلة، على الوجه التالي:

١. تكتب التعليمات بلغة بسيطة.
٢. تكتب كل عبارة على سطر مستقل.
٣. تستخدم الكلمات الحاكمة Keyword والإزاحة للداخل لكي تميز الهياكل الحاكمة.
٤. تكتب كل مجموعة متميزة من التعليمات(بلوك) من أعلى إلى أسفل، بمدخل واحد، ومخرج واحد.
٥. قد تضم مجموعات التعليمات في وحدة بنائية مميزة باسم مستقل.

وقد اتبع أسلوب صياغة الخوارزمات بأسلوب اللغة شبه البرمجية في هذا الكتاب لكونها تسمح للمبرمج أن يركز على الجانب المنطقي من برنامجه. على أن البديل التصويري المعروف باسم مخططات ناسى– شنايدر قد تمت معالجته.

الفصل الأول

مقدمة للبرمجة بلغة C++

Introduction to programming in C++

البرنامج هو مجموعة متتابعة من الأوامر أو التعليمات يتم تنفيذها بواسطة الحاسب (Computer) كل برنامج يكتب بلغة من لغات البرمجة. لغة C++ (وتنطق سي بلس بلس) تعتبر واحدة من أحدث وأقوى لغات البرمجة المتداولة فهي تمكن المبرمج (programmer) من كتابة البرامج الموجهة المركبة الأفضل كفاءة.

هذا الباب يقدم بعض السمات الرئيسية للغة C++. يجب أن تقوم بترجمة compile وتنفيذ كل مثال في هذا الباب.

١.١ برامج مبسطة:

مثالنا الأولى يوضح الأجزاء الرئيسية لبرنامج C++

مثال ١.١ برنامج ترحيب العالم

```
#include < iostream.h>
// This program prints "Hello, World."
cout <<"hello, World.\n";
return 0;
```

تعليمة التوجيه #include التي في السطر الأول ضرورية لأي برنامج له خرج، فهي تشير إلى ملف خارجي يسمى iostream.h يحتوي على معلومات عن الهدف cout. لاحظ أن أقواس الزاوية > و < ليست جزءا من اسم الملف ولكنها تستخدم لبيان أن هذا الملف من مكتبة لغة C++ القياسية.

السطر الثاني في البرنامج هو ملاحظة أو تعليق comment ويبدأ بالشرطتين المائلتين //.

الملاحظات أو التعليقات توضع في البرنامج لتزويد القراء بالشرح والتوضيح عن خطوات البرنامج، وهذه الملاحظات لا يهتم بها المترجم(compiler) ولا تؤثر في تنفيذ البرنامج.

السطر الثالث في البرنامج يحتوي على الدالة الرئيسية ( ) main. وهذه الدالة يجب أن تكون موجودة في كل برنامج C++ فهي تخبر المترجم من أين يبدأ تنفيذ البرنامج. القوسين ( ) التابعين لكلمة main يجب يكونا موجودين أيضا.

يحتوي **السطر الرابع والسابع** على القوسين } و { وفقط. يوجد بين هذين القوسين جسم الدالة ( )main وهما لازمين لكل برنامج C++.

السطر الخامس يحتوي على الجملة الآتية:

```
Cout<<"Hello,world.\n"
```

وهذه الجملة تعني إرسال الرسالة "Hello, World.\n" إلى الهدف Cout وتنطق "سي أوت" (see out) هذا الهدف هو مجرد خرج قياسي standard output stream عادة يكون شاشة الحاسب. الاسم cout يرمز إلى "خرج وحدة تحكم طرفية، هذا الخرج سيظهر كالآتي:

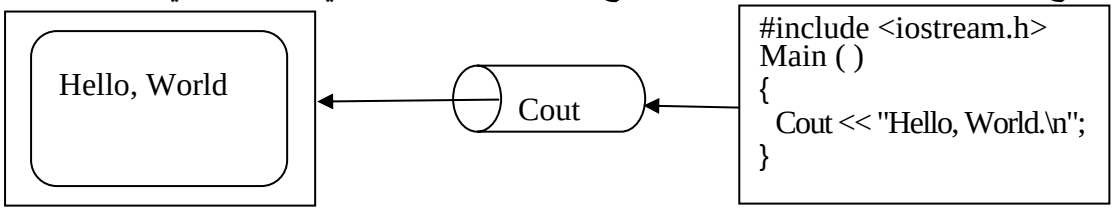
Hello, World

الرمز n\ هو رمز الانتقال إلى سطر جديد. لا حظ أنه رمز واحد يتكون من الشرطة المائلة "\" وحرف "n" وجود هذا الرمز في نهاية الجملة بين علامتي اقتباس يخبر الحاسب بأن يبدأ سطر جديد بعد طباعة الحرف السابق للرمز n\ وهذا يعني نهاية السطر الحالي.

السطر السادس يحتوي على الجملة return 0. وهذه الجملة تنهي تنفيذ البرنامج والعودة إلى نظام التشغيل للحاسب. الرقم 0 يستخدم كإشارة بأن البرنامج انتهى بنجاح. عبارة الخرج التي في السطر الخامس تحتوي على عديد من الرموز الشائعة في لغة سي ++. الرمز » يسمى معامل الخرج (output operator): معامل الإدماج (insertion operator). فهو يدمج أو يضع الرسالة المراد إخراجها في مجرى الخرج (output stream). وجود الرمز n\ في نهاية الرسالة يعني الانتقال إلى سطر جديد وجود هذا الرمز في أي ***الرسالة المراد إخراجها يسبب نهاية السطر الحالي في الخرج وبداية سطر جديد. لاحظ أن كل من الرموز (» و n\) يتكون من حرفين متجاورين ليس بينهما فراغ أو مسافة.

لاحظ وجود الفاصلة في نهاية كل من السطر الخامس والسادس، كل جملة أو أمر في C++ يجب أن تنتهي بالفاصلة المنقوطة ليس بالضرورة أن تكون كل جملة في سطر مستقل، يمكننا وضع عدد من الجمل على سطر واحد ويمكن وضع جملة واحدة على عدد من السطور. ليس المهم وضع الجملة على واحد أو أكثر ولكن المهم أن كل جملة يجب أن تنتهي بالفاصلة المنقوطة.

نستطيع أن نتخيل علاقة الهدف cout للبرنامج وشاشة الحاسب كما في الشكل التالي:



مجرى الخرج cout يعمل عمل قناة تنقل الخرج من البرنامج إلى شاشة الحاسب(أو الطابعة أو أي جهة خرج آخر) حرف بعد حرف.

البرنامج الذي في المثال 1.1 ليس هو أقصر برنامج، بعض أجزاءه فقط هي المطلوب لكل برنامج في الحقيقة برنامج C++ لا يحتاج أن يحتوي على أي جمل (statements). وبالطبع مثل هذا البرنامج يكون برنامج خالي أو فارغ "empty" لا يقوم بتنفيذ أي شيء وهذا البرنامج الخالي هو أقصر برنامج C++ والمثال التالي يوضح أقصر برنامج C++

مثال 2.1 أقصر برنامج C++

```
Main ( ) { }
```

هذا البرنامج الخالي أو الفارغ لا يقوم بعمل أي شيء هو ببساطة يبين الهيكل المطلوب لكل برنامج C++.

معظم برامج الترجمة(compilers) لا تحتاج للعبارة ;0 return. بعض برامج الترجمة يعطي تحذير إذا حذفت هذه العبارة من البرنامج. كل مثال في الباب الأول سوف يحتوي على العبارة return 0

نوصي أيضا أن يحتوي كل برنامج في بدايته على ملاحظة أو تعليق يوضح ما يفعله البرنامج.

2.1 معامل الخرج

الرمز » يسمى معامل الإدماج insertion operator أو معامل الخرج output، فهو يدمج أو يضع خرج البرنامج في مجرى الخرج المسمى على يساره. عادة تستخدم مجرى الخرج cout الذي يشير إلى شاشة الحاسب. لذلك عند تنفيذ الجملة 66 » cout سوف يظهر الرقم 66 على الشاشة المعامل أحيانا يقوم بتنفيذ عملية واحدة أو أكثر. معامل الخرج » يقوم بإرسال قيمة الجملة التعبيرية الموجودة على يمينه إلى مجرى الخرج الموجود على يساره. وحيث أن تنفيذ هذا العمل يتم من اليمين إلى اليسار لذلك تم اختيار الرمز » ليمثل معامل الخرج. يجب أن تتذكر أن اتجاه السهم يشير إلى اليسار.

سبب تسمية cout بالمجرى (أو النهر) هو أن الخرج المرسل إليه يسير مثل المجرى أو النهر، لو أن مجموعة أشياء أرسلت إلى cout فإنها تسير في خط كما تسير نقط الماء في النهر واحدة بعد الأخرى، لذلك فإنها تظهر على الشاشة حسب ترتيبها.

مثال 3.1

هذه النسخة من برنامجنا Hello World لها نفس الخرج في المثال 1.1

```
# include <iostream.h >
//This program illustrates the sequential output of three strings.
Main(){
cout«"Hello,"«"wor"«ld.\n";
return0;}
```

output….Hello, World.....

في هذا المثال تم تجزيء الرسالة المراد إخراجها على الشاشة إلى ثلاثة أجزاء. حيث أن سطر الخرج ينفذ من اليسار إلى اليمين فإن كل جزء يرسل إلى مجرى الخرج حسب ترتيبه: الأول "Hello" ثم "wor" وأخيرا "ld.\n" حيث أنه لا يوجد أي رموز للانتقال إلى سطر جديد بين الثلاثة أجزاء، لذك فإنهم ظهروا جميعا في سطر واحد كما في المثال 1.1. يستخدم مجرى للخرج cout عادة مع معامل الإدماج » في الصورة العامة الآتية:

```
Cout « expression «expression«….«expression;
```

تركيب هذه الجملة يبين أن cout يمكن أن يكون متبوعا بزوج أو أكثر. كل زوج مكون من عامل الإدماج » متبوعا بجملة تعبيرية. في المثال 3.1 يوجد ثلاثة أزواج.

3.1 الحروف characters وسلاسل الحروف string literals

كلمة "Hello" تسمى سلسلة حروف string literal حيث أنها تتكون من مجموعة من الحروف بين علامتي اقتباس.

تعريف الحرف character هو أي حرف من حروف الهجاء أو رقم من الأرقام(0 إلى 9) أو علامة من العلامات مثل(+ و – و# و% و.....). معظم أجهزة الحاسب تستخدم نظام الشفرة الأمريكي القياسي لتبادل المعلومات(ASCII code). انظر الملحق A لمجموعة الشفرة الكاملة لهذا النظام. هذه المجموعة تحتوي على 52 حرفا كبيرا upper case وحرفا صغيرًا lower case من حروف الهجاء وكذلك 10 أرقام من (0 إلى 9) وكل علامات الترقيم(– و+ و* و...) الموجودة على لوحة مفاتيح الحاسب وكذلك بعض الحروف التي لا تطبع.

رمز الانتقال إلى سطر جديد "\n" هو واحد من الرموز التي لا تطبع. هذا الرمز مكون من الشرطة المائلة \ وحرف n. يوجد أيضا مجموعة أخرى من الرموز مكونة بهذه الطريقة مثل رمز ترك حفل خالي أفقي "\t" ورمز الإنذار الذي ينتج صفير عند الطباعة "\a" أيضا لطباعة الشرطة المائلة نفسها يستخدم الرمز "\\".

الحروف يمكن أن تستخدم في جملة البرنامج كجزء من سلسلة حروف أو كأهداف منفردة. عند الاستعمال المنفرد يجب أن تظهر في شكل ثوابت حرفية. الثابت الحرفي character constant هو حرف موجود بين علامتي اقتباس مفردة. ثوابت الحروف مثل الأهداف المفردة يمكن أن تخرج بنفس الطريقة التي تخرج بها سلسلة الحروف.

مثال 4.1 نسخة أخرى من برنامج الترحيب بالعالم:

هذه النسخة من برنامجنا Hello World لها نفس الخرج مثل النسخ السابقة من البرنامج.

```
#include < iostream.h>
//This program illustrates the output of strings and characters:
main ( )
{
cout<<"Hello,"<<'w'<<'o'<<'r'<<"1d"<<'.'<<'\n';
return0;
}
```

Output……..Hello, World……..

جملة الخرج في هذا البرنامج ترسل سبعة أهداف إلى cout: 2 سلسلة حروف هما "Hello" و"1d" و 5 ثوابت حرفية هي 'w' , 'o' ,'r', '.' , '\n'.

بالطبع الحروف المفردة يمكن أن تستخدم في شكل سلسلة حروف. جملة خرج البرنامج السابقة يمكن أن تستبدل بالآتي:

```
cout << "Hello,"<< "w"<<"o"<<"r"<<"1d"<< "."<<"\n";
```

هذه الجمل ترسل سبع سلاسل حروف إلى cout. لكن عند التعامل مع الحروف المفردة كأهداف مستقلة من الأفضل استعمال ثوابت الحروف. سلاسل الحروف تخزن بطريقة مختلفة وتحتاج إلى حيز أكبر.

سلسلة الحروف التي لا تحتوي على أي حروف تسمى سلسلة حروف فارغة empty string ويرمز لها بالرمز "". يمكننا طباعة رسالتنا باستخدام سلسلة الحروف الفارغة كالآتي:

```
cout << "Hello,Wo"<<"  "<<"  "<<"rl"<<"  "<< "  "<<"d.\n";
```

4.1 طول سلسلة الحروف

طول سلسلة الحروف هو عدد الحروف التي تحتوي عليها. على سبيل المثال طول سلسلة الحروف "ABCDE"هو 5 حروف.

لغة C++ تحتوي على دالة خاصة تسمى Strlen0 التي يمكنك استعمالها لمعرفة طول أي سلسلة حروف. وهذا موضح بالمالي التالي:

مثال 5.1

هذا البرنامج يطبع أطوال مجموعة من سلسلة الحروف

```
#include <iostream.h>
#include <string.h >
//This program tests the strlen()function
Main ( )
{
cout<< strlen ("Hello, World.\n")<<'\n';
cout<< strlen ("Hello, World. ")<<'\n';
cout<< strlen ("Hello, ")<<'\n';
cout<< strlen (" ")<<'\n';
cout<< strlen ("")<<'\n';
return 0;}
```

خرج البرنامج هو :-

```
14
13
7
1
0
```

الدالة ( ) strlen ببساطة تعد عدد الحروف الموجود في سلسلة الحروف المراد معرفة طولها. أول سطرين في خرج البرنامج هما العدد ١٤ والعدد ١٣ وذلك يوضح ان رمز الإنتقال لسطر جديد n/ تم اعتباره كحرف واحد.

إيضاحها أو شرحها، التعليق الثالث يشغل كل السطر الثامن. وهو يسبق كلمة ( ) main ويوضح باختصار ما يفعله البرنامج، التعليق الرابع موجود بداخل عبارة الخرج. وهذا غير مرغوب فيه. التعليق الخامس موجود في نهاية السطر الذي يحتوي على عبارة الخرج. التعليق السادس موجود في نهاية البرنامج. المثال التالي يبين برنامجنا "Hello World" مع كيفية كتابة التعليقات بلغة C++.

مثال 7.1 استعمال الشرطتين المائلتين فقط في كتابة الملاحظات.

هذه النسخة تبين كيف أن كل التعليقات الهامة تكتب بسهولة باستعمال الشرطتين المائلتين:

```
//---------------------------------------------------------------------------
// program to demonstrate comments
// Written by H. R. Hubbard
// June 11, 1996
// Version 1.6
//---------------------------------------------------------------------------
#include <iostream.h>          //This directive is needed to use cout
// Prints message ; "Hello, World.";
Main()
{
Cout<<"Hello, World";  //change?
return0;                    //Some compilers will complain if you omit this line
}
```

لاحظ أن هذه التعليقات محدودة بالشرطتين المائلتين إلى أخر السطر فقط ولا يمكن أن تمتد إلى أكثر من سطر إلا إذا بدأ كل سطر بالشرطتين المائلتين.

6.1 المتغيرات Variables والأهداف Objects والإعلان عنها

المتغير هو رمز يمثل مكان للتخزين في ذاكرة الحاسب. المعلومة التي تخزن في هذا المكان تسمى قيمة المتغير value. أكثر الطرق شيوعا لحصول المتغير على قيمة هي طريقة التخصيص assignment. وهذه الطريقة تأخذ الشكل التالي:

```
Variable = expression;
```

التعبير expression تقدر قيمته أولا ثم تخصص هذه القيمة للمتغير variable. علامة التساوي= تسمى معامل التخصيص assignment operator في لغة C++

مثال 8.1

هذا برنامج بسيط لغة سي ++ يستعمل متغير للأعداد الصحيحة يسمى n.

```
#include <iostream.h>
// A simple example to illustrate assignment:
Main()
{
int n;
n=66;
cout<< n<< endl;
return 0;
}
```

Output ……..66

السطر الأول بين القوسين { } يعلن عن أن n متغير من نوع الأعداد الصحيحة int. الجملة التي في السطر الثاني تخصص القيمة 66 للمتغير n.

لاحظ استعمال الثابت الرمزي endl. إرسال هذا الثابت الرمزي إلى cout يكافئ نهاية السطر '\n' ولكن في هذه الحالة يفرغ مخزن الخرج المؤقت output buffer.

في المثال السابق قيمة n هي 66. وهذه القيمة في الحقيقة خزنت في ذاكرة الحاسب في بتات bits متتالية كل منها يحتوي على الرقم 0 أو 1. الحاسب يفسر أو يترجم هذه البتات المتتالية كعدد صحيح لأن المتغير أعلن عنه بأنه متغير للأعداد الصحيحة.

الإعلان declaration عن متغير هو جملة تعطي معلومات عن هذا المتغير لمترجم C++. والإعلان يأخذ الشكل التالي:

```
Type variables
```

حيث أن type هي اسم لأحد أنواع المتغيرات في C++ فعلى سبيل المثال الإعلان التالي:

```
Int n ;
```

يخبر المترجم بأمرين: (1) اسم المتغير هو n و (2) هذا المتغير من نوع الأعداد الصحيحة int. كل

مثال 10.1

```
#include <iostream.h>
// A simple program to illustrate tokens:
main()
{
int n=66;
cout<< n<< endl;
return 0;
}
66
```

هذا البرنامج يبين 15 رمزا هي: main،)، (، {، int، n، =، 66، ;، cout، >>، endl، return، 0 و {. الرمز n هو متغير والرموز 66، 0 و endl ثوابت والرموز int و return كلمات مفتاحية والرموز = و >> هي معاملات والرموز) و(و} و ; و{ علامات ترقيم. يحتوي أول سطرين في البرنامج على توجيه أو إرشاد المعالج الأولى (preprocessor). والتعليقات وهي ليست جزء حقيقي من البرنامج.

الكلمات المفتاحية تسمى أيضا الكلمات المحجوزة أو المخصصة لأن هذه الكلمات محجوزة لأغراض معينة في لغة البرمجة ولا يمكن استعمالها كأسماء لمتغيرات أو أي أغراض أخرى.

المميز (identifier) هو سلسلة من الحروف الأبجدية والعددية تبدأ بأحد الحروف الأبجدية.

يوجد ٥٣ حرفا أبجديا: ٥٢ حرفا بالإضافة إلى الشرطة السفلية _. يوجد ٦٣ حرفا أبجديا وعدديا ٥٣حرفا أبجديا و١٠أرقام (0,1,……9). الكلمات () cout, n, int, main و endl تعتبر مميزات وكذلك the_day_after_tomorrow, Lastname, y4, x1m stack. لاحظ أن لغة C++ حساسة لحالة الحرف: فهي تميز الحروف الكبيرة (Uppercase) من الحروف الصغيرة (lowercase) لذلك فإن stack و stack يعتبران مميزان مختلفان.

المميزات تستخدم لتسمية الأشياء مثل المتغيرات والدوال. في البرنامج السابق، main اسم لدالة وint إسم لنوع بيانات و n و cout أسماء لمتغيرات و endl اسم ثابت. بعض المميزات مثل int تسمى كلمات مفتاحية لأنها جزء جوهري من لغة البرمجة نفسها. (الكلمات المفتاحية الـ 48 التي تعرف لغة C++ موضحة في الملحق B). مميزات أخرى مثل n معرفة في البرنامج نفسه.

8.1 إعطاء قيمة ابتدائية للمتغير في أمر الإعلان

المتغير يمكن أن يأخذ قيمة ابتدائية "initialized" وذلك بإعطائه قيمة عند الإعلان عنه.

مثال 11.1 إعطاء قيم ابتدائية للمتغيرات

هذا البرنامج البسيط يوضح طريقتين لإعطاء قيم ابتدائية للمتغير بداخل أمر الإعلان عنه.

```
#include <iostream.h>
// This shows how to initialize variable as they are declared:
Main()
{
Int George = 44;
Int Martha =33;
Int sum= George + martha;
Cout<< George<<" +" << Martha  <<" ="<< sum<< endl;
Return0;
}
14+33=77
```

المتغيران George و Martha أخذا القيم الابتدائية ٤٤ و ٣٣ بداخل أمر الإعلان عنهما. مع الإعلان عن المتغير sum فإن الجملة التعبيرية George + Martha تحدد بالمجموع ٤٤+٣٣ وقيمة الناتج ٧٧ خصصت للمتغير sum.

إعطاء قيم ابتدائية "initialization" هي تقريبا نفس طريقة التخصيص"assignment". كل من الطريقتين يستعمل علامة التساوي "=" متبوعا بجملة تعبيرية. الجملة التعبيرية تقدر أولا ثم تخصص قيمة الناتج للهدف الموجود على يسار معامل التخصيص.

بصفة عامة، من الأفضل إعطاء قيما للمتغيرات عند الإعلان عنها.

إعطاء قيم ابتدائية يمكن أن يستخدم أيضا عند الإعلان عن أكثر من متغير بعبارة إعلان واحدة كما يوضح المثال الآتي:

مثال 12.1 إعطاء قيم ابتدائية للمتغيرات:

```
#include <iostream.h>
// This shows how to initialize variable as they are declared:
Main()
{
Int  n1, n2=55 , n3, n4, n5=44, n6;
Cout<< n2<<" ," << n5  <<endl;
return0;
}
```

Output……..55,44

كل الستة متغيرات من n1 إلى n6 أعلن عنهم بالنوع int ولكن متغيرين فقط n2 و n5 خصص لهم قيما في جملة الإعلان.

بعض برامج الترجمة (compilers) (على سبيل المثال Borland C++) سوف يعطي تحذير لو أن أي من المتغيرات لم يخصص له قيمة ابتدائية.

9.1 التخصصات المتسلسلة:

التخصيص نفسه هو عبارة تعبيرية لها قيمة أو مقدار. قيمة الجملة التعبيرية

X=22

هي ٢٢. ومثل أي قيمة أخرى، قيمة التخصيص يمكن أن تستخدم في تخصيص آخر:

Y=(x=22);

هذا هو التخصيص المتسلسل. بداية خصص القيمة ٢٢ لـ x ثم بعد ذلك خصص القيمة ٢٢ لـ y.

التخصص المركب عادة يكتب بدون أقواس

y=x=22;

بصفة عامة، قيمة التخصيص آخر قيمة خصصت.

مثال 13.1 التخصيصات المخفية:

هذا المثال يبين كيفية استعمال التخصيص بداخل الجملة التعبيرية.

```
#include <iostream.h>
// This shows that an assignment can be part of a larger expression:
Main()
{
Int m, n;
M=(n=66) +9;                //(n=66) is an assignment expression
Cout<< m<<" ," << n_  <<endl;
return0;
}
```

Output……75,66

التخصيص المركب يخصص أولا القيمة ٦٦ للمتغير n. وعند ذلك يحسب قيمة الجملة التعبيرية (n=66)+9 التي هي 75. عند ذلك يخصص هذه القيمة(75) للمتغير m.

التخصيصات المختصرة عادة يجب تجنبها. على سبيل المثال أول سطرين في البرنامج السابق يمكن أن يكتبا بطريقة أفضل كالآتي:

```
Int n = 66;
Int m = n+9;
```

هذا يوضح أيضا الطريقة المفضلة لإعطاء قيم ابتدائية للمتغيرات مع الإعلان عنها. توجد بعض الحالات للتخصيصات المختصرة تجعل البرنامج أسهل في القراءة، على سبيل المثال العبارة الوحيدة التالية أفضل من ٨عبارات مفصلة:

```
N1= n2 = n3 = n4 = n5 = n6 = n7 = n8 = 65535;
```

سوف نرى أمثلة أخرى للتخصيصات المختصرة في الفصل الثالث.

التخصيص المتسلسل لا يمكن استخدامه في إعطاء قيم ابتدائية عند الإعلان:

```
Int x = y =22;                    // ERROR
```

سبب الخطأ في الجملة السابقة هو أن إعطاء قيم ابتدائية للمتغيرات ليست تخصيصات. هما متشابهان ولكن المترجم يتعامل مع كل منها بطريقة مختلفة، الطريقة الصحيحة للجملة السابقة هي:

```
Int x =22, y = 22;          //OK
```

10.1 الفاصلة المنقوطة Semiclan

في لغة C++ الفاصلة المنقوطة نستخدم كفاصل بين العبارات أو نهاية الجملة. كل جملة(أو عبارة) يجب أن تنتهي بالفاصلة المنقوطة. هذا يختلف عن اللغات الأخرى مثل Pascal التي تستخدم الفاصلة المنقوطة كفاصل للأوامر، لاحظ أن السطور التي تبدأ بالرمز # مثل:

#include < iostream.h>

لا تنتهي بالفاصلة المنقوطة لأنها ليست جملة من البرنامج ولكنها توجيهات للمعالجة الأولية.

ذكرنا في المقطع السابق أن جملة(أو أوامر)C++ يمكن النظر إليها كصيغة تعبير جبرية والعكس صحيح. الجمل التعبيرية يمكن أن تستخدم كعبارات (أو أوامر) منفردة. على سبيل المثال الجمل التالية مقبولة في لغة C++.

X+y;
22;

هذه الجمل لا تؤدي أي عمل ولا فائدة منها. ولكنها جمل مقبولة في لغة C++. سوف ترى بعض الجمل المفيدة فيما بعد.

الفاصلة المنقوطة تعمل مثل المعامل في صيغة التعبير الجبرية. فهي تحول صيغة التعبير الجبرية إلى أمر. الفاصلة المنقوطة ليست معامل حقيقي لأنها لا تنتج قيمة. ولكن هذا التحويل يساعد في توضيح الفرق بين صيغة التعبير الجبرية(expression) والأمر (Statment).

11.1 شكل البرنامج:-

لغةC++ هي لغة بدون قيود، فليس لها متطلبات عن أماكن وضع عناصر البرنامج على السطر أو الصفحة. وبالتالي فإن المبرمج عنده حرية كاملة في تنسيق شكل البرنامج، ولكن المبرمجين من أهل الخبرة يعلمون أن مهمة الكتابة وتصحيح الأخطاء والحفاظ على نجاح البرنامج تكون أسهل كثيرا باستخدام طريقة متناسقة للبرمجة وسهلة القراءة، علاوة على ذلك فإن آخرين سوف يجدون أن برامجك تكون أسهل في القراءة لو أنك اتبعت المصطلحات القياسية لأسلوب البرنامج، إليك بعض القواعد البسيطة التي يتبعها معظم المبرمجين بلغة C++

- وضع كل التوجيهات include# في بداية ملف البرنامج.
- ضع كل جملة (أو أمر) على سطر جديد.
- اترك مسافة في بداية كل الجمل التي داخل نفس البلوك.
- اترك مسافة على جانبي كل معامل كالتالي jn=4z

هذه القواعد تقريبا متبعة ف يكل مكان في هذا الكتاب.

قاعدة أخرى من المهم إتباعها وهي اختيار أسماء المتغيرات بدقة. استعمل الأسماء القصيرة لتقليل فرصة الأخطاء المطبعية. ولكن أيضا اختار الأسماء التي تصف ما تمثله المتغيرات. هذا يسمى شفرة توثيق ذاتي self-documenting code. تقريبا، كل المبرمجين بلغة C++ يستخدمون الحروف الصغيرة في أسماء المتغيرات إلا إذا كان الاسم مكونا من كلمات متعددة فيكون الحرف الأول من كل كلمة حرف كبير. على سبيل المثال:

Char niddleInitial
Usigned maxUnsignedint

هذه الأسماء أسهل في القراءة من middleinitial و maxunsigndint كبديل لذلك فإن بعض المبرمجين يستعمل الشرطة الأفقية (underscore) كالتالي:

Char middle_initial;
Usigned max_unsigned_int;

12.1 أنواع الأعداد الصحيحة:-

الأعداد الصحيحة هي كل مجموعة الأرقام 0,1,-1,-2,3,-3 الخ: الأعداد الصحيحة الموجبة(unsigned integer) هي الأعداد الصحيحة الغير سالبة.
3,2,1,0 الخ. لغة C++ بها تسعة أنواع من الأعداد الصحيحة:

عدد صحيح قصير بدون إشارة	unsigned short int
عدد صحيح بدون إشارة	unsigned int
عدد صحيح طويل بدون إشارة	unsigned long int
عدد صحيح قصير	short int
عدد صحيح	int
عدد صحيح طويل	long int
حرف	char
حرف بإشارة	signed char
حرف بدون إشارة	unsigned char

الفرق بين الأنواع التسعة في مدى القيم التي تسمح باستعمالها، مدى القيم يعتمد على نظام الحاسب المستعمل على سبيل المثال في معظم أجهزة الحاسب الشخصية(PCs). مدى الأعداد الصحيحة int يتراوح بين ٣٢٧٦٧ و -٣٢٧٦٨ بينما في معظم محطات التشغيل UNIX فالمدى يتراوح بين ٢١٤٧٤٨٣٦٤٧ و -٢١٤٧٨٣٦٤٨، المقطع "int" يمكن حذفه من أسماء الأنواع unsignel short ،int، long int , short int، unsignel int و unsignel long int.

البرنامج الذي في المثال يطبع مدى كل نوع من الأعداد الحقيقية الموجودة على جهازك الحدود القصوى SCHAR_MIN و LONG_MAX الخ هي ثوابت مخزنة في الملف الرأسي <limit.h> لذلك أضيف توجيه المعالجة الألى الآتي

#include < limit.h>

الضروري لقراءة هذه الثوابت.

مثال 14.1 مدى أنواع الأعداد الصحيحة:

البرنامج التالي يطبع الحدود القصوى لمدى الأنواع المختلفة للأعداد الصحيحة:

```
#include <iostream>
#include <limits.h>
//prints the constants stored in limits.h:
Main()
{
Cout << "minimum car = "<<CHAR_MIN<<endl;
Cout << "maximum car = "<<CHAR_MAX<<endl;
Cout << "minimum chort = "<<SHRT_MIN<<endl;
Cout << " maximum chort = "<<SHRT_MAX<<endl;
Cout << "minimum int = "<<INT_MIN<<endl;
Cout << " maximum int = "<<INT_MAX<<endl;
Cout << "minimum long = "<<LONG_MIN<<endl;
Cout << " maximum long = "<<LONG_MAX<<endl;
Cout << "minimum signed cahr = "<<CHAR_MIN<<endl;
Cout << "maximum signed cahr = "<<CHAR_MAX<<endl;
Cout << "maximum unsigned cahr = "<<UCHAR_MAX<<endl;
Cout << "maximum unsigned short = "<<UCHAR_MAX<<endl;
Cout << "maximum unsigned = "<<UINT_MAX<<endl;
Cout << "maximum unsigned long = "<<ULONG_MAX<<endl;
Return 0;
}
Minimum char =-128
Maximum char =127
Minimum short= -32768
Maximum short =32767
Minimum int = -2147483648
Maximum int =2147483647
Minimum long = -2147483648
Maximum long =2147483647
minimum signed cahr=-128
maximum signed cahr=-127
maximum unsigned cahr=255
```

maximum unsigned chort=65535
maximum unsigned =4294967295
maximum unsigned long= 4294967295

هذا الخرج من محطة تشغيل UNIX، فهو يبين أن هذا النظام يحتوي على ست أنواع فقط للأعداد الحقيقية:

المدى من −١٢٨ إلى ١٢٧ (واحد بايت) char

المدى من 32767- إلى 32767(2 بايت) short

المدى من 2147483648- إلى 2147483647(4 بايت)int

المدى من 0 إلى 255 (1 بايت) unsigned cahr

المدى من 0 إلى 65535(2 بايت) unsigned chort

المدى من 0 إلى 4294967295 (4 بايت) unsigned

يمكنك أن تعلم أن الأعداد الصحيحة من نوع short على سبيل المثال تشغل 2 بايت (16 بت) على هذا الحاسب لأن المدى 32768- إلى 32767 يغطي 2^{16} = 65536 قيمة ممكنة.(تذكر أن البايت = 8 بت وهي الوحدة القياسية لتخزين الحروف)

عند تشغيل بورلاند C++ على جهاز حاسب شخصي فإن هذا البرنامج سينتج نفس المدى لكل الأنواع ما عدا int و unsigned ينتجان التالي:

المدى من 32768- إلى 32767 (2 بايت) int

المدى من 0 إلى 65535 (2 بايت) unsigned

13.1 المعاملات الحسابية البسيطة:

المعامل (operator) هو رمز يؤثر على تغيير جبري واحد أو أكثر منتجا قيمة يمكن تخصيها لمتغير. سابقا تقابلنا مع معامل الخرج >> ومعامل التخصيص=.

بعض المعاملات البسيطة هي المعاملات التي تؤدي العمليات الحسابية:+و−و/و%. وهذه المعاملات تعمل على أنواع الأعداد الحقيقية لتنتج عدد حقيقي آخر: m+n حاصل جمع m و n، أو m-n حاصل طرح n من m، أو m-n حاصل ضرب m في n أو m/n حاصل قيمة m على n، أو m% n يعطي باقي خارج قسمة m/n هذه الستة معاملات ملخصة في الجدول التالي وموضحة في المثال الذي يليه.

جدول 1.1 معاملات الأعداد الصحيحة الحسابية

مثال	الوصف	المعامل
m+n	جمع	+
m-n	طرح	−

مثال	الوصف	المعامل
-n	سالب	–
m*n	ضرب	*
m/n	قسمة	/
m%n	باقي القسمة	%

```
#include < iostream.h>
//Tests arithmetic operators:
Main()
{
Int m = 38, n =5;
Cout << m<< "+"<<n<<"="<< (m+n)<<endl;
Cout << m<< "-"<<n<<"="<< (m-n)<<endl;
Cout << m<< "−"<<n<<"="<< (−n)<<endl;
Cout << m<< "*"<<n<<"="<< (m*n)<<endl;
Cout << m<< "/"<<n<<"="<< (m/n)<<endl;
Cout << m<< "%"<<n<<"="<< (m%n)<<endl;
return 0;
}
```

```
38+5=43
38-5=33
-5=-5
38*5=190
38%5=3
```

لاحظ ان 38/5=7 و 38%5=3. هاتين العمليتين تعطيان معلوما كاملة عن القسم المادية لـ 38 على 5 حيث 38÷5= 7.6، الجزء الصحيح للنتيجة هو7 =35÷5 والجزء الكسري هو 0.6= 5 +3، خارج قسمة العدد الصحيح وهو 7 وباقي القسمة 3 يمكن تجميعهم مع المقسوم 38 والمقسوم عليه 5 في الصيغة الآتية:

7×5+3=38

معاملات خارج قسمة العدد الصحيح وباقي القسمة تكون أكثر تعقيدا إذا كانت الأعداد الصحيحة ليست موجبة. بالطبع فإن المقسوم عليه يجب أن لا يكون صفرا. لكن لو أن أي من m أو n سالب عندئذ m/n و m%n يمكن أن يعطي نتائج مختلفة من حاسب إلى آخر، المطلب الوحيد هو

q * n + r =m

حيث q = m/n و r =m % n

على سبيل المثال 14- مقسومة على 5 تعطي 2.8-

خارج القسمة الصحيح يمكن أن يقرب إلى 3- أو 2- لو أن الحاسب قرب خارج القسمة qإلى 3- عند ذلك فإن باقي قسمة العدد الصحيح يكون 1. لكن لو أن جهاز الحاسب قرب q إلى 2- عند ذلك فإن 3 سوف تكون 4-.

مثال 16.1 قسمة الأعداد الصحيحة السالبة

هذا البرنامج يستعمل لتحديد كيفية معالجة الحاسب لقسمة الأعداد الصحيحة السالبة:

```
#include < iostream.h >
//Tests quotient and remainder operators:
main()
{
int m = -14, n=5, q=m/n, r= m%n;
cout <<"m=" <<m << endl;
cout <<"n=" <<n << endl;
cout <<"q=" <<q << endl;
cout <<"r=" <<r << endl;
cout <<"q*n+r=" <<"("<<q <<")*("<<n <<")+" <<r<<"= "<<q*n+r<<" = "<< m
<<endl;
return0;
}
```

```
m= -14
n=5
q=-2
r=-4
q*n+r=(-2)*(5)+-4=-14=-14
```

هذا يعطي نفس النتائج من كل من محطة التشغيل UNIX باستخدام المعالج الدقيق"processor" من نوع Motorola68040 ومن الحاسب الشخصي باستخدام المعالج الدقيق من نوع Intel Pentium.

14.1 الملازمة وأسبقية تنفيذ التعليمات:

لغة C++ غنية بالعديد من المعاملات، (الملحق C يحتوي على 55 منهم)، حيث أن التعبير الجبري يمكن أن يحتوي على أكثر من معامل لذلك من المهم أن نعلم ترتيب أو أسبقية تنفيذ هذه المعاملات، نحن معتادين على ترتيب المعاملات الحسابية العادية:

معاملات الضرب * والقسمة / والباقي % لها أسبقية عن معاملات الجمع+ والطرح– أي أنها تنفذ أولا. على سبيل المثال:

42-5*3

تنفذ كالآتي:–

42-(3*5) =42 – 15 =27

أكثر من ذلك، كل المعاملات الحسابية لها أسبقية في التنفيذ عن معاملات التخصيص والخرج. على سبيل المثال الأمر

n=42-3*5

سوف يخصص القيمة 27 للمتغير n، أولا المعامل * يستدعي ليحسب 3*5 بعد ذلك المعامل – يستدعي ليحسب 42-15 بعد ذلك المعامل = يستدعي ليخصص القيمة ٢٧ للمتغير n

هذا هو جزء من الجول C1 في الملحق C.

جدول 2.1 بعض معاملات لغة C++

المعامل	الوصف	أسبقية التنفيذ	الملازمة	الحالة	مثال
–	سالب	15	يمين	أحادي	-n
*	الضرب	13	شمال	ثنائي	-m*n
/	القسمة	13	شمال	ثنائي	m/n
%	باقي القسمة	13	شمال	ثنائي	m%n
+	الجمع	12	شمال	ثنائي	m+n
–	الطرح	12	شمال	ثنائي	m-n
<<	إزاحة للشمال	11	شمال	ثنائي	cout<<n
=	التخصيص البسيط	2	يمين	ثنائي	m=n

هذا الجدول به ٨معاملات تستعمل مع متغيرات الأعداد الصحيحة. هذه المعاملات لها ٥مستويات مختلفة للأسبقية، على سبيل المثال معامل النفي الأحادي له مستوى أسبقية ١٥ ومعامل الضرب الثنائي * له مستوى أسبقية ١٣ عملية النفي تنفذ قبل عملية الضرب. لذلك فإن التعبير الجبري m*-n ينفذ كالتالي m*(-n). معاملات التخصيص لها أقل أسبقية عن كل المعاملات الأخرى لذلك فهي عادة تتم (أو تنفذ) في الآخر.

العمود المسمى " الملازمة" يخبرنا ماذا يحدث عندما يكون عدة معاملات مختلفة لها نفس مستوى الأسبقية تظهر في نفس التعبير الجبري على سبيل المثال + و – لهما مستوى الأسبقية ١٢ وملازمة إلى اليسار لذلك فإن المعاملات تنفذ من اليسار إلى اليمين. على سبيل المثال في التعبير الجبري:

8-5+4

أولا: 5 تطرح من 8 وبعد ذلك 4 تضاف إلى الناتج:

(8-5)+4=3+4=7

العمود المسمى الحالة" Arity" مدون به إذا كان المعامل أحادي أو ثنائي، أحادي"unary" يعني أن المعامل ينفذ على طرف واحد. على سبيل المثال معامل الزيادة البعدية ++ هو أحادي ++n يعمل على متغير واحد n، الثنائي(binary) يعني أن المعامل ينفذ على طرفين، على سبيل المثال الجمع+ هو ثنائي:حيثm+n ينفذ على متغيرين m و n.

15.1 معاملات الزيادة والنقصان:

لغة C++ بها خصائص كثيرة موروثة من لغة C من أهمها معامل الزيادة ++ ومعامل النقصان --.

هذه المعاملات تحول المتغير إلى صيغة جبرية مختصرة لأشكال معينة من أوامر التخصيص.

مثال 17.1 معاملات الزيادة والنقصان

لغة سي بلس بلس بها خصائص كثيرة موروثة من لغة السي من أهمها معامل الزيادة ++ ومعامل النقصان (--) هذه المعاملات تحول المتغير إلى صيغة جبرية مختصرة لأشكال معينة من أوامر التخصيص.

هذا المثال يوضح كيفية عمل معاملات بالزيادة والنقصان

```
#include < iostream.h >
//Tests quotient and remainder operators:
main()
{
int m =44 ,  n = 66;
cout << "m="<<m " <<, n="<<n<< endl;
++m;
--n;
cout << "m="<<m "<<, n="<<n<< endl;
m++;
n--;
cout << "m="<<m<< ", n="<<n<< endl;
return0;
}
m=44, n=66
m=45 , n=65
m=46 , n=64
```

كل معامل الزيادة القبلي ومعامل الزيادة البعدي له نفس التأثير هنا: كل منهم يجمع واحد على قيمة m أيضا معامل النقصان القبلي n– ومعامل النقصان البعدي —nله نفس التأثير: كل منهم يطرح واحد من قيمة n.

عند الاستخدام في جملة تعبيرية منفردة فإن m ++ و ++m كل منهما يكافئ التخصيص التالي:

m=m+1;

كل منهم يزيد قيمة m بمقدار 1. أيضا الجملة التعبيرية nn- - و - -n كل منهما يكافئ التخصيص التالي:

n=n-1;

كل منهما ينقص قيمة n بمقدار واحد، (معامل الزيادة ++ يستخدم في الاسم "c++" لأنه يزيد "increment" لغة C الأصلية ولغة C++ يوجد بها كل شيء موجود في لغة C وزيادة).

على أي حال عند استعمال التعبير الجبري الفرعي(تعبير جبري داخل التعبير الجبري)فإن عملية الزيادة القبلية m++ تختلف عن عملية الزيادة البعدية ++m الزيادة القبلية تزيد التغير لأولا قبل استعماله في التعبير الجبري الأكبر بينما الزيادة البعدية تزيد قيمة المتغير بعد استعمال القيمة السابقة له بداخل التعبير الجبري الأكبر، حيث أن عملية الزيادة تكافئ عملية تخصيص مستقلة. فإنه هي الحقيقة يوجد أمرين يتم تنفيذهما عند استعمال عملية الزيادة في تعبير جبري فرعي: تخصيص الزيادة والأمر الشامل الأكبر، الفرق بين الزيادة القبلية والزيادة البعدية هو ببساطة الفرق بين تنفيذ التخصيص قبل أو بعد الجملة الشاملة.

مثال 18.1 معاملات الزيادة القبلية والزيادة البعدية

هذا المثال يوضح الفرق بين الزيادة القبلية والزيادة البعدية

```
#include < iostream.h >
//Tests quotient and remainder operators:
main()
{
int m=66 , n;
n=++m;
cout<< "m="<<m<<",n="<<n<<endl;
n=m++;
cout<< "m="<<m<<",n="<<n<<endl;
cout<< "m="<<m++<< endl;
cout<< "m="<<m<< endl;
cout<< "m="<<++m<< endl;
```

```
return 0;
}
m=67 , n=67
m = 68 , n = 67
m = 68
m = 69
m = 70
```

في التخصيص الأول زيادة قبلية لـ m لذلك قيمتها زادت إلى ٦٧ ثم خصصت هذه القيمة للمتغير n. في التخصيص الثاني زيادة بعدية لـ m لذلك القيمة 67 خصصت للمتغير n ثم بعد ذلك m زادت إلى 69. في آخر جملة خرج زيادة قبلية لـ m زادت إلى ٧٠ أولا وبعد ذلك وضعت هذه القيمة في مجرى الخرج.

استعمال معاملات الزيادة والنقصان في التعبيرات الجبرية الفرعية ممكن أن يخدع ويجب استعماله بحذر، على سبيل المثال فإن ترتيب تنفيذ التعبيرات الجبرية التي تحتوي على هذه المعاملات ليست معرفة في اللغة وبالتالي لا يمكن التنبؤ بنتيجتها.

مثال 19.1 عدم التنبؤ بترتيب تنفيذ التعبيرات

```
#include < iostream.h >
main()
{
int n = 5 , x;
x= ++n* -n;
cout<< "n="<<n<<",x="<<x<<endl;
cout<< ++n<<" "<<+nn<<" "<<++n<<endl;
}
n = 5 , x= 25
8-7-6
```

في التخصيص للمتغير x، أولا n قد زادت إلى 6 وبعد ذلك نقصت إلى 5 قبل تنفيذ معامل الضرب وبذلك كان حساب قيمة x هو 5*5، في آخر سطر، نفذت التعبيرات الجبرية الفرعية من اليمين إلى اليسار، الملازمة اليسارية لمعامل الخرج >> ليست لها أهمية لأنه لا يوجد معاملات أخرى لها نفس مستوى الأسبقية.

16.1 تعبيرات التخصيص المركبة:

معاملات الزيادة والنقصان تختصر أنواع معينة من التخصيصات، لغة c++ تسمح أيضا بجمع معامل التخصيص مع المعاملات الأخرى، الشكل العام لهذه التخصيصات المركبة هو:

```
variable op = expression
```

على سبيل المثال التخصيص المركب

n+ = 8;

له نفس التأثير مثل التخصيص البسيط

n= n+8;

مثال 20.1 معاملات التخصيص:

هذا المثال يوضح كيفية استعمال معاملات التخصيص المركبة

```
#include < iostream.h >
//Tests combined operators:
main()
{
int n = 44;
n+= 9;
cout << n << endl;
n - = 5;
cout << n << endl;
n * = 2;
cout << n << endl;
return 0;
}
53
48
96
```

الأمر n+ = 9 تضيف 9 إلى n والجملة n- =5 تطرح 5 من n، والجملة n*=2 تضرب n في 2.

17.1 تجاوز الحد الأعلى والحد الأدنى للأعداد الصحيحة:

الأعداد الصحيحة في الحاسب محدودة بخلاف الأعداد الصحيحة في الرياضيات البحتة. كما رأينا سابقا كل نوع من الأعداد الصحيحة له قيمة عظمى وقيمة صغرى، لو أن قيمة المتغير زادت عن حدود نوع هذا المتغير فإننا نحصل على ما يسمى فائض حسابي(overflow)

مثال 21.1 اختبار الفائض الحسابي:

هذا البرنامج يوضح ماذا يحدث عندما تحدث إفاضة في المتغير من النوع القصير.

```
#include < iostream.h >
#include < limits.h >
//Tests for overflow for type short:
main()
{
short n = SHRT_MAX -1;
```

```
cout << n++<<endl;
cout << n++<<endl;
cout << n++<<endl;
cout << n++<<endl;
return 0;
}
32766
32767
-32768
-32767
```

إن القيم تلف أو تدور حول النهايتين العظمى والصغرى 32767 و 32768- بمعنى آخر فإن قيمة الناتج عند إضافة 1 إلى 32767 هو 32768-، وهذا خطأ واضح.

معظم أجهزة الحاسب يتعامل مع الفائض الحسابي بهذه الطريقة، القيم تدور حول النهايات، لذلك فإن الرقم الذي يأتي بعد النهاية العظمى هو النهاية الصغرى، وهذا أسوأ نوع للخطأ الذي يمكن أن يحدث في الحاسب لأنه عادة لا يوجد أي دليل خارجي لحدوث أي شيء خطأ لحسن الحظ لغة C++ تحتوي على إمكانيات تساعد المبرمج للتغلب على هذه المشكلة كما سوف نرى فيما بعد.

الفائض الحسابي هو أحد أنواع الأخطاء التي تحدث عند تنفيذ البرنامج (run_time error)، مثال آخر أكثر شيوعا هو القسمة على الصفر، وهذا النوع ليس مشكلة كبيرة لأنك تستطيع أن تعرفه عندما يحدث حيث أن البرنامج ينهار"crashes"، أما الفائض الحسابي فهو مثل النزيف الداخلي، فأنت ربما لا تدرك أنك في خطر مميت.

81.1 النوع الحرفي:

في لغة C++ النوع الحرفي char هو أحد أنواع الأعداد الصحيحة، هذا يعني أن أي متغير من نوع char يمكن أن يستخدم في التعبير الجبري للأعداد الصحيحة تماما مثل أي نوع للأعداد الصحيحة الأخرى على سبيل المثال معاملات الأعداد الصحيحة تطبق على المتغيرات من النوع الحرفي char.

```
char c = 45;
char d = 2*c-7;
c+ =d%3;
```

الاسم "char" هو مختصر لكلمة "character" الاسم char يستخدم لأن المتغيرات من هذا النوع عندما تكون داخل أو خرج للبرنامج فإنها تفسر أو تفهم على أنها حروف. عندما يمثل الحرف داخل البرنامج فإن النظام يقوم بتخزينه كشفرة أسكي ASCII code (انظر المحلق A) كقيمة للعدد الصحيح

من النوع الحرفي char. عندما يكون المتغير الحرفي يمثل خرج للبرنامج فإن النظام(system) يرسل الحرف المقابل إلى مجرى الخرج، وهذا موضح في المثال القادم.

لغة C++ تحتوي على ثلاثة أنواع للأعداد الصحيحة كل منها له 8 بت(8-bits): signed , char و unsigned char، ولكن نوعين فقط من هذه الأنواع الثلاثة مختلفين. النوع char يمكن أن يكون أي من unsigned char أو signed char تبعا لجهاز الحاسب. استخدم النوع char للحروف العادية، استعمل النوع unsigned char في سلسلة الحروف القصيرة، النوع signed char غالبا غير واضح الاستعمال ولكنه ممكن أن يكون اختيار جيد لو أنك احتجت أن تخزن كمية كبيرة من الأعداد الصحيحة القصيرة جدا التي تكون خرج للبرنامج بواسطة معامل الخر القياسي".

مثال 22.1 الخرج الحرفي:

هذا المثال يبين كيفية خروج المتغيرات الحرفية من البرنامج

```
#include < iostream.h >
//Tests output of type cahr:
main()
{
char c = 64;
cout << C++<<" ";          //prints '@' and increments c to 65
cout << C++<<" ";          //prints 'A' and increments c to 66
cout << C++<<" ";          //prints 'B' and increments c to 67
cout << C++<<" ";          //prints 'C' and increments c to 68
cout =96;
cout << C++<<" ";          //prints ',' and increments c to 97
cout << C++<<" ";          //prints 'a' and increments c to 98
cout << C++<<" ";          //prints 'b' and increments c to 99
cout << C++<<" ";          //prints 'c' and increments c to 100
return 0;
}
@ A B C
,  a b c
```

أول جملة للخرج تطبع المتغير الحرفي C في مجرى الخرج، حيث أن هذا المتغير الحرفي له قيمة عددية 64 فإن خرج هذا المتغير يكون الحرف "@".(شفرة الاسكي لهذا الرمز هي 64)، بعد ذلك فإن قيمة المتغير C تزداد مباشرة إلى 65 التي تسبب أن يكون الحرف"A" هو الخرج التالي.(شفرة الاسكي للحرف

A هي 65) البرنامج يستمر بنفس الطريقة إلى نهايته. (لاحظ أن أجهزة الحاسب التي تستعمل الشفرة EBCDIC سوف يختلف خرجها عن الخرج المذكور هنا).

شفرة الاسكي كاملة مبينة في الملحق A

مثال 23.1 الحصول على شفرة الاسكي

```
#include < iostream.h >
//Tests output of type cahr:
main()
{
char c = 'A' ;
cout << C++<<" "<<      int(c) << endl ;     // prints 'A' and 65
cout << C++<<" "<<      int(c) << endl ;     // prints 'B' and 65
cout << C++<<" "<<      int(c) << endl ;     // prints 'C' and 65
return 0 ;
}
A 65
B 66
C 67
```

كما تم في تنفيذ هذا البرنامج فإن المتغير الحرفي c أخذ القيم 65, 66 , 68,67. حيث المتغيرات الحرفية تطبع كحروف لذلك فإن أول شيء طبع على كل سطر هو الحرف الذي له شفرة أسكي مخزنة في المتغير الحرفي c. لذلك تم طباعة الحروف A , B , C ، نحن نستخدم int (c) لطباعة القيمة العددية للمتغير الحرفي c.

التعبير int (c) يسمى تحويل النوع "cast'، فهو يحول المتغير c من متغير حرفي إلى متغير صحيح، وهذا يسمح لنا بطباعة شفرة الأسكى للحرف.

أسئلة للمراجعة

1.1 صف طريقتين لكتابة التعليقات comments في برنامج .C++

2.1 ما هو الخطأ في التعليق التالي:

```
cout << " Hello,/*change?*/World.\n" ;
```

3.1 ما الذي يفعله الإعلان declaration؟

4.1 ما الغرض من توجيهات أو إرشادات المعالجة الأولية؟

```
#include < iostream.h >
```

5.1 هل هذا برنامج C++ مقبول أو صحيح ؟ وضح

```
main() {22;}
```

6.1 من أين أتى اسم لغة "C++"؟

7.1 ما هو الخطأ في هذه الإعلانات:

```
in first = 22 , last = 99 , new = 44 old = 66 ;
```

8.1 ما هو الخطأ في هذه الإعلانات

```
int x =y = 22 ;
```

9.1 ما هو الخطأ في هذا البرنامج

```
main()
(n =22 ;
cout << n<< endl ;
}
```

10.1 أوجد قيمة كل من التعبيرات الجبرية الآتية إذا كان صحيحا أو بين لماذا هو غير صحيح:

a.37/(5%2)
b.37/5/2
c.37(5/2)
d.37%(5%2)
e.37%5%2
f. 37-5-2
g. (37-5)2

11.1 أوجد قيمة كل من التعبيرات الجبرية الآتية بفرض أن قيمة m=24 و n=7 في كل حالة:

a. m-8-n
b. m=n=3
c. m%n
d. m%n++
e. m%++n
f. ++m-n- -
g. m+ = n- = 2

12.1 حدد في كل مما يأتي المميزات الصحيحة، إذا كان المميز غير صحيح وضح لماذا؟

a. r2d2
b. H20
c. secondCousinonceRemoved
d. 2nBirthday
e. the_united_states_of_America
f._TIME_
g. _12345
h. x(3)
i. cost_in_s

مسائل محلولة

13.1 ما هو خرج البرنامج التالي:

```
#include < iostream.h >
main()
{
//cout << "Hello, World.\n";
}
```

هذا البرنامج ليس له خرج، الشرطتين المائلتين يحولوا جملة الخرج إلى ملاحظة.

14.1 ما الخطأ في البرنامج التالي:

```
#include < iostream.h >

//This program prints "Hello, World":
main()
{
cout<<"Hello, World.\n";
return 0;
}
```

الفاصلة المنقوطة ناقصة من نهاية جملة الخرج.

10.1 اكتب أربع جمل مختلفة لـ C++ كل منهم يطرح واحد من متغير العدد الصحيح n.

```
n=n-1;
n- =1;
- - n;
n- -;
```

16.1 اكتب جملة لـ C++ تطرح مجموعة x و y من z وبعد ذلك تزيد قيمة y بمقدار واحد.

```
z- =(x+y++)
```

17.1 اكتب جملة واحدة لـ C++ تنقص قيمة المتغير n، وبعد ذلك تضيفه إلى المتغير total.

```
total + = - -n;
```

18.1 في كل مما يأتي افترض أن m=5 و n=2 قبل تنفيذ الأمر، بين ما قيمة كل من m و n بعد تنفيذ بالأوامر التالية:

```
a. m* =n ++;
b. m+ = - -n;
```

a. n سوف تكون 3و m سوف تكون10

b. n سوف تكون 1 و m سوف تكون 6

19.1 بين وصحح الخطأ في كل مما يلي:

a. cout >> count;
b. m = ++n+ =2;

a. مجرى خرج الهدف cout يحتاج إلى معامل الخرج>>
الأمر يجب أن يكون cout>> cout

b. 20.1 التعبير n++ لا يمكن أن يكون على يسار التخصيص

تتبع الأوامر التالية وبين قيمة كل متغير بعد تنفيذ كل أمر:

```
int x , y, z ;
x=y=z=6;
x*=y+ =z- = 4;
```

أولاالرقم 6 خصص للمتغيرات z و y و x. بعد ذلك المتغير z طرح منه 4 وأصبح ٢.
عندئذ المتغير y زاد بـ 2 وبذلك أصبحت قيمته 8، عندئذ المتغير x ضرب في 8 وأصبحت قيمته 48.

21.1 على معظم محطات التشغيل UNIX يكون مدى نوع الأعداد الصحيحة int من -2,147,843,647 إلى 2,147,483,648. كم عدد البايتات (bytes) التي يشغلها المتغير من هذا النوع في ذاكرة الحاسب.

المدى من 2,147,483,648 إلى 2,147,483,647 يغطي 4,294,967, قيمة. هذا الرقم هو 232. لذلك كل عدد صحيح يتطلب 32 بت(bits) التي هي عبارة عن 4 بايت من ذاكرة الحاسب.

22.1 ما هو الفرق بين الأمرين التاليين:

```
char ch = 'A' ;
char ch = 65 ;
```

كل من الأمرين له نفس التأثير: يعلن عن ch أنه متغير حرفي ويخصص له القيمة 65. حيث أن هذه القيمة هي شفرة الأسكى للحرف 'A' ويمكن للثابت الحرفي أن يستخدم أيضا في تخصيص الرقم65 للمتغير ch.

23.1 ما هي الأوامر التي يجب أن تنفذ لإيجاد الحرف الذي له شفرة أسكي100؟

```
char ch = 100;
cout << c;
```

24.1 كيف تستطيع أن تحدد إذا كان النوع char هو نفس النوع Signed char أو unsigned char على جهاز الحاسب؟

نفذ برنامج مثل الذي في المثال 14.1 وقارن الثوابت char_max و schar_max و unchar_max.

مسائل محلولة في البرمجة

25.1 اكتب برنامج يطبع أول جملة من عنوان Gettysburg.

من الضروري أن كل الذي سوف نحتاج إليه هنا هو استعمال مجموعة من جمل الخرج ترسل أجزاء جملة العنوان إلى الهدف cout.

```
#include < iostream.h>
//prints the first sentence of the Gettysburg Address:
main ()
{
cout<<"\tFourscore and seven years ago fathers\n";
cout<< " brought forth upon this continent a new nation, \n";
cout<< " conceived in liberty , and dedicated to the \n";
cout<< " proposition that all men are created equal.\n";
return();
}
```

```
Fourscore and seven years ago our fathers
brought forth upon this continent a new nation,
conceived in liberty , and dedicated to the proposition that all men are created equal.
```

نستطيع أيضا أن نفعل هذا بتسلسل أجزاء جملة العنوان مع جملة خرج واحدة إلى cout كالتالي:

```
cout<<"\tFourscore and seven years ago fathers\n";
cout<< " brought forth upon this continent a new nation, \n";
cout<< " conceived in liberty , and dedicated to the \n";
cout<< " proposition that all men are created equal.\n";
```

لاحظ أن هذا أمر واحد مع فاصلة منقوطة واحدة.

إذا كنت ترغب في أن تكون سطور خرج البرنامج أطول(أو أقصر) من الأجزاء المفردة المرسلة إلى مجرى الخرج فإنه ببساطة يمكن أن تضبط مكان حرف السطر'\n';.

```
#include < iostream.h>
//prints the first sentence of the Gettysburg Address:
main ()
{
cout<<"\tFourscore and seven years ago fathers ";
cout<< " brought forth upon this continent a new nation ";
```

```
cout<< " conceived in liberty , and dedicated \n to the ";
cout<< " proposition that all men are created equal.\n";
return();
}
```

Fourscore and seven years ago our fathers brought forth upon this continent a new nation, conceived in liberty, and dedicated to the proposition that all men are created equal.

لا تنس أن تضع مسافة خالية بعد آخر كلمة في كل سطر لا ينتهي بحرف نهاية السطر.

26.1 اكتب برنامج يطبع مجموع وفرق وضرب وخارج قسمة وباقي قسمة متغيرين من الأعداد الصحيحة خصص القيم 60 و 7 للمتغيرين.

بعد الإعلان عن المتغيرين m و n بأنهم من الأعداد الصحيحة، وتخصيص القيم 60 و 7 للمتغيرين.

بعد الإعلان عن المتغيرين m و n بأنهم من الأعداد الصحيحة وتخصيص القيم60 و 7 لهم نستخدم جملة خرج واحدة لطباعة قيم المتغيرات وبعد ذلك جملة خرج واحدة لكل عملية من العمليات الخمسة.

```
#include < iostream.h>
//prints sum, difference, product , and quotient of given integers:
main ()
{
int m = 60 , n = 7 ;
cout << "The integers are " << m<<" and " <<n<< endl;
cout<< " Their sum is              " <<(m+n)<< endl;
cout<< " Their difference is          " <<(m-n)<< endl;
cout<< " Their products is            " <<(m* n)<< endl;
cout<< " Their quotient is            " <<(m/ n)<< endl;
cout<< " Their remainder  is          " <<(m % n)<< endl;
return 0;
}
```

```
The integers are 60 and 7
Their sum is                67
Their difference is 53
Their product is 420
Their quotient is 8
Their remainder is 4
```

لاحظ أن خارج القسمة 8 وباقي القسمة 4 يناسب العلاقة المطلوبة لخارج قسمة وباقي قسمة العدد الصحيح (4)+(7)(8)=60

27.1 اكتب برنامج يطبع بلوك الحرف "B" مكونة من 6*7 من النجوم كالتالي:

نستخدم جملة خرج واحدة لكل صف في بلوك الحرف

```
#includle  < iostream.h>
//prints the block letter "B" in a 7×6 grid;
main()
{
cout << "*****"<< endl;
cout << "*       *"<< endl;
cout << "*       *"<< endl;
cout << "*****"<< endl;
cout << "*       *"<< endl;
cout << "*       *"<< endl;
cout << "*****"<< endl;

return 0;
}
```

بدلا من endl لكل خرج يمكن أن ننهي كل سلسلة حرف بين علامتي اقتباس بحرف نهاية'\n' كالتالي:

```
cout << "*****\n"
```

مسائل إضافية

28.1 تتبع الأوامر التالية وبين قيمة كل متغير بعد تنفيذ كل منهم

```
int x,y, z;
x=y =z=5 ;
x*= y+=  z- = 1;
```

29.1 في معظم الأنظمة مدى النوع unsigned char هو0 إلى 255كم عدد البايتات التي سوف يشغلها المتغير من هذا النوع في ذاكرة الحاسب؟

مسائل برمجة إضافية

30.1 اكتب ونفذ برنامج يطبع اسمك وعنوانك.

31.1 اكتب ونفذ على الحاسب يطبع الجملة الأولى من عنوان Gettysburg بحيث لا يزيد عن 40 حرف في السطر.

32.1 نفذ البرنامج في المثال 11.1 على جهاز الشخصي. استخدم الخرج لتحديد الأنواع المختلفة للأعداد الصحيحة وكم عدد البايتات المطلوبة لكل منهم.

33.1 غير البرنامج المبين في المثال 16.1 لترى كيفية تعامل جهازك الشخصي مع قسمة العدد الصحيح 20 على 7-. حاول أن تتنبأ بخارج القسمة وباقي القسمة بعد ذلك نفذ برنامجك على الحاسب لترى إذا كنت على صواب.

34.1 اكتب ونفذ برنامج على الحاسب يطبع الحرف الأول من اسمك الأخير كبلوك لحرف في شبكة مكونة من 7 ×7 نجوم(stars).

35.1 اكتب ونفذ برنامج على الحاسب يطبع الأربع سطور الأولى من قصيدة شكسبير 18:

shall I compare there to a summer's day?
Thou art more iovely and more temperate.
rough winds do shake the darling buds of May,
And summer's lease hath all too short a date.

36.1 لكي تعلم ماذا يفعل جهازك الشخصي في المتغيرات التي لم يخصص لها قيم في البداية اكتب ونفذ برنامج على الحاسب يحتوي على السطرين الآتيين:

```
int n ;
cout << n<< endl;
```

38.1 اكتب ونفذ برنامج على الحاسب يسبب فائض حسابي لمتغير من نوعint.

39.1 اكتب ونفذ برنامج على الحاسب مثل المثال 22.1 يطبع شفرة الأسكي للأرقام من 1 إلى 10 وآخر 5 حروف صغيرة. استخدم الملحق A لتختبر خرج البرنامج.

40.1 اكتب ونفذ برنامج على الحاسب مثل المثال 22.1 يطبع شفرة الأسكي لعشرة حروف متحركة كبيرة وصغيرة ، استخدم الملحقA لتختبر خرج البرنامج.

إجابات أسئلة المراجعة

1.1أحد الطرق هي استخدام تعليق لغة C القياسية:

```
/*like this */
```

الطريقة الأخرى هي استخدام تعليق لغة C++:

```
// like this
```

الطريقة الأولى تبدأ بشرطة مائلة ونجمة وتنتهي بنجمة وشرطة مائلة، والطريقة الثانية تبدأ بشرطتين مائلتين وتنتهي بنهاية السطر.

2.1 أي شيء بين علامتي الاقتباس سوف يطبع بما في ذلك التعليق المقصود.

3.1 الإعلان يخبر المترجم باسم ونوع المتغير الذي أعلن عنه، أيضا يمكن تخصيص قيم للمتغير بداخل جملة الإعلان.

4.1 إنه يحتوي على الملف الرئيسي iostream.h في البرنامج، وهذا يحتوي على الإعلانات المطلوبة للدخل والخرج مثل معامل الخرج >>.

5.1 هذا برنامج C++ مقبول (أو صحيح)، فهو يحتوي على جملة واحدة:32 وهذه هي جملة جبرية لأن أي ثابت مثل 22 هو تعبير جبري مقبول. البرنامج لا يقوم بعمل أي شيء.

6.1 الاسم يشير إلى لغة C ومعامل الزيادة ++. الاسم يوحي بأنه لغة سي++ متقدمة عن لغة سي.

7.1 الشيء الوحيد الخطأ في هذه الإعلانات هي كلمة new لأنها من الكلمات المفتاحية(keyword) الكلمات المفتاحية هي كلمات محجوزة ولا يمكن استخدامها كأسماء لمتغيرات.

8.1 الطريقة الوحيدة لاستخدام علامة التساوي في الإعلان هي تخصيص قيمة لمتغير الجبري x=y=22 غير مقبول في جملة واحدة تحتوي على الإعلان والتخصيص معا لأن المتغير لا يكون على يمين أو علامة تساوي. التعبير الصحيح يكون الآتي:

int x = 22, y = 22;

9.1 يوجد خطأين، المتغير n لم يعلن عنه و cout استخدم بدون وجود عبارة التوجيه للملف < iostream.h> في البرنامج.

10.1 أ- (2*5) 37 تقدر بـ 37/10=3

ب- 37/5/2 تقدر بـ (37/5)/2=7/2=3

جـ- هذا غير صحيح لعدم وجود معامل بين 37 و(5/2).

العملية المرادة كانت (5/2)*37 التي تقدر بـ 37/2 =18

د- 37%5%2 تقدر بـ 37%1=0

هـ- 37%5%2 تقدر بـ (37%5)%2=2%2=0

و-37-5-2 تقدر بـ (37-5)-2=32-2=30

ز- هذا غير صحيح لعدم وجود معامل بين5-37) و 2.

العملية المقصودة هي 2*(37-5) التي تقدر بـ 32*2=64

11.1 أ- m-8-n تقدر بـ (24-8)-7=16-7=9

ب- m=n=3 تقدر بـ3

جـ- m%n تقدر بـ 24%7=3

د-++m%n تقدر بـ 24%(7++)=24%7=3

هـ- m%++n تقدر بـ 24%(++7)= 24%8=0

و-m-n-++ تقدر بـ (++24)-(7- -)=25 -7=8

ز- m+=n-=2 تقدر بـ 24+=(7- =2) – 24= =5 =29

12.1 أ- r2d2 هي مميز مقبول (أو صحيح).

ب- H2O هي مميز مقبول.

جـ- SecondCourrsinonceRemoved هي مميز مقبول.

د- 2ndBirthday مميز غير صحيح لأن أول حرف رقم وهذا غير مقبول.

هـ- مميز صحيح.

و-مميز صحيح.

ز- مميز صحيح.

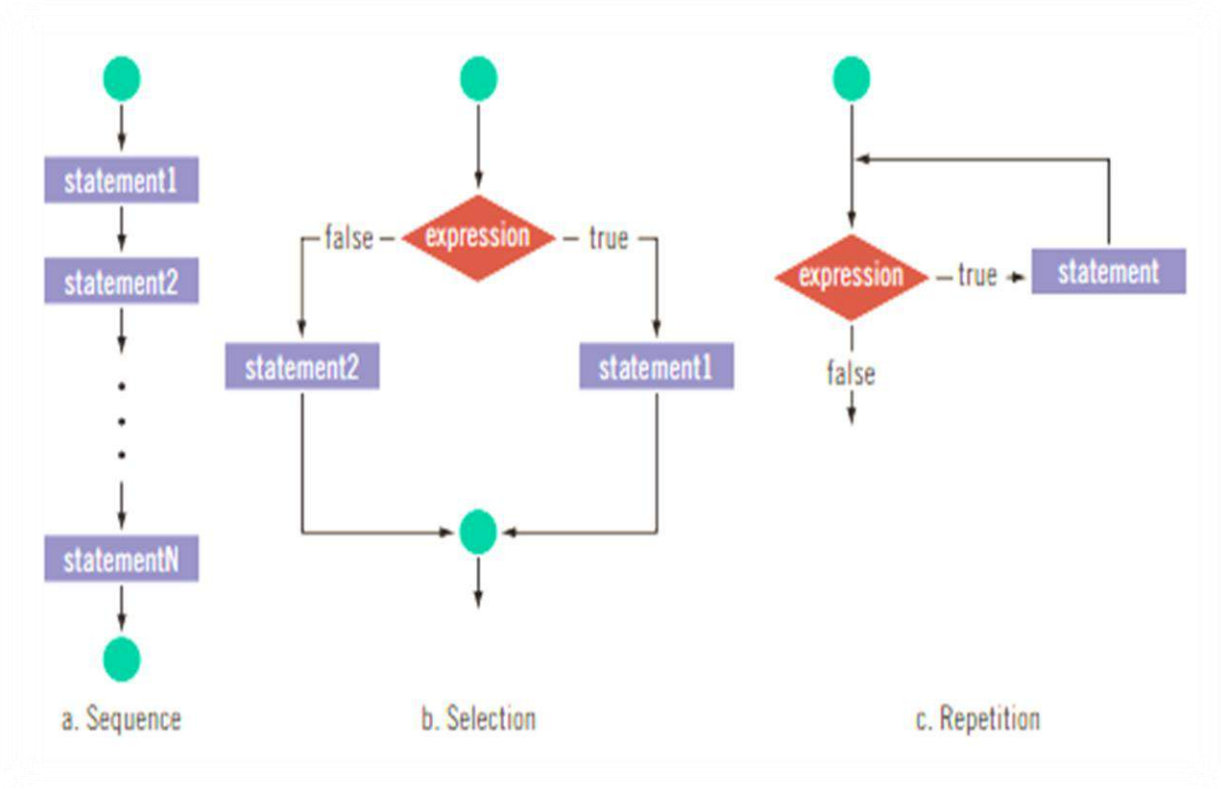
ح- x(3) مميز غير صحيح لأنه يحتوي على الأقواس(,) وهذا غير مقبول.

ط-$_cost_in مميز غير صحيح لأنه يحتوي على علامة الدولار$.

الفصل الثاني
الأوامر الشرطية
وأنواع الأعداد الصحيحة
conditional Statements
and Integer Types

كل البرامج التي ذكرت في الفصل الأول تنفذ على الحاسب بطريقة متتالية(متسلسلة):كل أمر ينفذ مرة واحدة حسب ترتيبه في البرنامج. الأوامر الشرطية تعطي سهولة أكثر في إنشاء البرامج حيث أن بعض الأوامر يتوقف تنفيذها على تحقيق شرط(أو شروط) معين يتغير أثناء تشغيل البرنامج.

هذا الفصل يصف طريقة استعمال أمر الشرط if وأمر الشرط if.....else وأمر الاختبار switch أيضا بين طريقة إدخال البيانات إلى البرنامج.



1.2 الدخل:

في لغة C++ الدخل هو المناظر(أو المماثل) للخرج. البيانات الخارجة من تنفيذ البرنامج تمر في مجرى الخرج cout والبيانات الداخلة من لوحة المفاتيح إلى البرنامج أثناء التشغيل تمر في مجرى الدخلcin(تنطق "see-in") الاسم مخصص لوحدة إدخال.

مثال 1.2 إدخال الأعداد الصحيحة:

هذا برنامج بسيط يقرأ الأعداد الصحيحة الداخلة من لوحة المفاتيح:

```
main()
{
int age ;
cout <<"How old are you:";
cin>> age;
cout << "In 10 years , you will be"<<age +10<<".\n";
}
How old are you:19
In 10years , you will be 29.
```

العدد المبين بالخط السميك في منطقة خرج البرنامج المظللة هو الدخل الذي تم بواسطة مستخدم البرنامج عن طريق لوحة المفاتيح.

الرمز << هو معامل الاستخلاص ويسمى أيضا معامل الإدخال. هذا المعامل يستخدم عادة مع مجرى الدخل cin والذي هو عبارة عن لوحة المفاتيح. لذلك عند تنفيذ الجملة التالية:

```
cin>> age;
```

فإن البرنامج يتوقف منتظرا أن يدخل مستخدم البرنامج عددا صحيحا من لوحة المفاتيح. وعند إدخال هذا العدد فإنه يخصص للمتغير age ثم يستمر البرنامج في التنفيذ. لاحظ أن أمر توجيه المعالجة الأولية:

```
#includle  < iostream.h>
```

غير موجود في المثال 2.1 ولكنه مطلوب في أي برنامج يستخدم في أي من cin أو cout. حيث أن كل برنامج في هذا الكتاب تقريبا يستخدم أي من cin أو cout فإننا سوف نفترض أنك تستطيع إضافة هذا التوجيه في بداية البرنامج. حذف هذا التوجيه من هذه الأمثلة ما هو إلا توفير حيز للطباعة. سوف تحذف أيضا كلمة return في نهاية البرنامج الرئيسي() main في كل الأمثلة التي سوف تأتي مستقبلا. مجرى الدخل cin هو مناظر لمجرى الخرج cout. كل منهم في لغة C++ يمثل مجرى عبارة عن قناة تمر خلالها المعلومات. المعلومات التي تمر إلى داخل البرنامج أثناء تشغيله تمر خلال cin والمعلومات التي تخرج من البرنامج تمر خلال cout هذا يمكن تخيله كالتالي:

مثال 2.2 إدخال الحروف:

```
main()
{
char frist , last;
cout<<" Enter your initials:\n;
cout<< "\tFirst name initial:";
cin>>first ;
cout<<"\tLast name initial:";
```

```
cin>>last;
cout << "Hello, "<< first<<"."<<last<<".!\n";
}
Enter your initials:
First name initial: J
Last name initial:H
Hello, J.H.!
```

هذا المثال يوضح طريقة قياسية لشكل الإدخال، أول سطر للخرج يذكر مستخدم البرنامج بالصورة العامة للدخل الذي يحتاج إليه، حيث يتبع ذلك مجموعة متتالية من طلبات الإدخال المحددة تسمى تذكيرات مستخدم البرنامج (user prompts) كل تذكير لمستخدم البرنامج يبدأ برمز ترك حقل خالي \t وبحذف رمز الانتقال إلى سطر جديد \n يجعل المؤشر cursor يظل على نفس السطر لإدخال رد المستخدم.

مثال 3.2 الإدخال المتعدد في نفس المجرى

يمكن قراءة أكثر من متغير بجملة إدخال واحدة:

```
main()
{
char first, last;
cout << "Enter your frist and last initials:";
cin>>first>> last;
cout<< "Hello,"<<first<<"."<<last<<".!\n";
}
Enter your first and last initials: JH
Hello, J. H!
```

هذا المثال يبين أن مجرى الدخل cin يقرأ المتغيرات من اليسار إلى اليمين، أي أن المتغير الذي في أقصى اليسار يقرأ أولا.

حيث أن النوع الحرفي char هو نوع للأعداد الصحيحة لذلك فإن cin سوف يهمل الأماكن الخالية white spaces التي تلي المسافات الخالية blanks والحقول tabs ورمز الانتقال إلى سطر جديد عند قراءة الدخل، لذلك فإن إدخال المتغيرات في هذا المثال يمكن أن يكون كالتالي:

```
Enter your first and last initials: J H
Hello , J. H. !
```

لاحظ ان هذا يمنع معاملة المسافات الخالية مثل معاملة الحروف باستخدام معامل الإدخال<<.

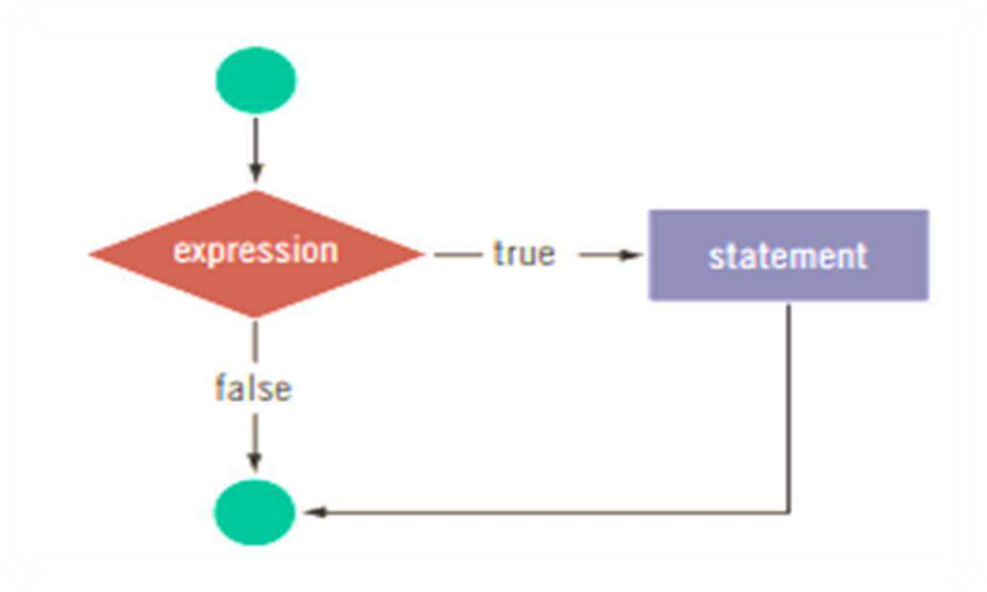
في الفصول القادمة سوف نرى طرق أكثر تخصص لإدخال الحروف.

2.2 عبارة if الشرطية

عبارة if الشرطية تسمح بتنفيذ بعض جمل البرنامج إذا تحقق شرط معين. الشكل العام لهذه العبارة الشرطية هو:

if(condition) statement;

حيث أن condition هو تعبير جبري نتيجته عدد صحيح و statement هي أي جملة قابلة للتنفيذ. الجملة سوف تنفذ فقط لو أن الشرط condition كانت قيمته لا تساوي صفرا.(عند حساب قيمة أي تعبير جبري كشرط، فإنه إذا كانت قيمة هذا التعبير لا تساوي صفر فإنها تترجم أن الشرط حقيقي true وإذا كانت قيمة التعبير تساوي صفر فإنها تترجم إلى أن الشرط غير حقيقي false. لاحظ وجود الأقواس حول الجملة الشرطية.



مثال 4.2 اختبار قابلية القسمة

```
main()
{
int n, d;
cout << "Enter two integers:";
con>> n>>d;
if(n%d==0)cout<<n<<" is divisible by" <<d<< endl;
}
```

```
Enter two integers: 24 6
24 is divisible by 6
```

هذا البرنامج يقرأ عددين من الأعداد الصحيحة ويختبر قيمة باقي خارج قسمة العددين n%d. في تنفيذ هذا البرنامج قيمة باقي خارج القسمة 24%6 هي صفر والتي تعني أن العدد 24 قابل للقسمة على العدد 6.

المشكلة في البرنامج السابق أنه لا يقوم بعمل أي شيء إذا كان العدد n غير قابل للقسمة على العدد d:

Enter two integers: 24 5

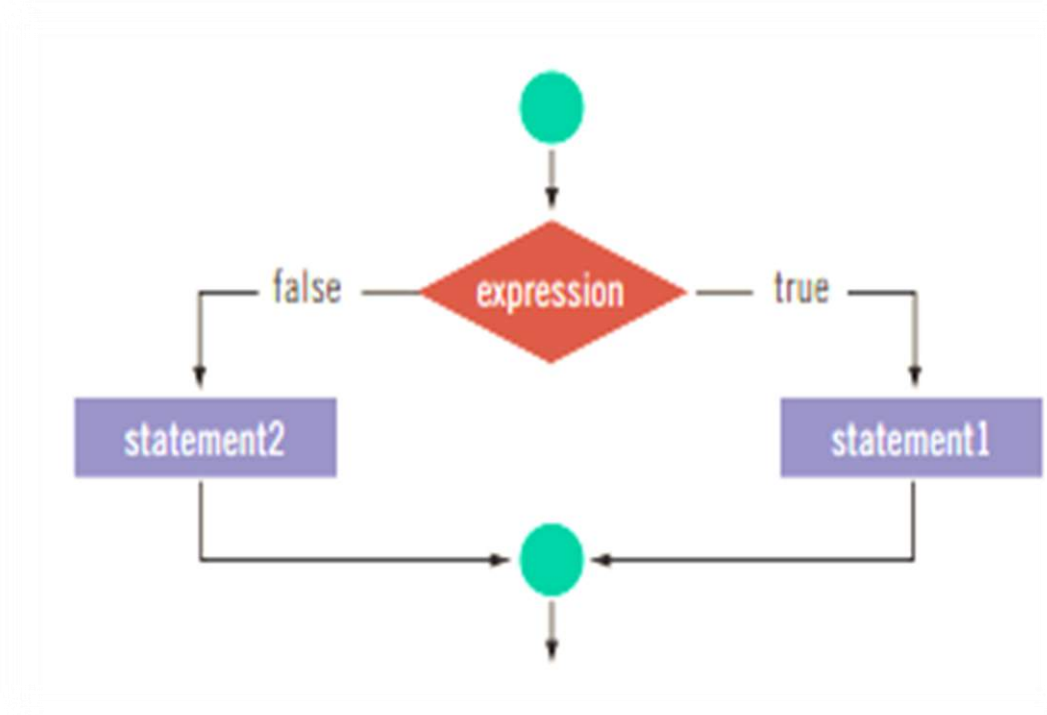
لكي يقوم البرنامج بتنفيذ جملة أخرى عندما تكون الجملة الشرطية تساوي صفر فإننا نحتاج لاستعمال الجملة الشرطية if……else.

3.2 الجملة الشرطية if …..else

الجملة الشرطية if….else تقوم بتنفيذ أحد أمرين تبعا لقيمة الشرط المحدد. وهذه الجملة الشرطية تأخذ الشكل الآتي:

```
If(condition) statementl;
Else statement2;
```

حيث أن الشرط condition هو تعبير جبري نتيجته عدد صحيح والأمر الأول statement1 والأمر الثاني statement2 هما أي جمل أوامر مطلوب تنفيذها، فالأول statement1 يتم تنفيذه إذا كانت القيمة العددية للشرط ليست صفرا والأمر الثاني statement2 يتم تنفيذه إذا كانت القيمة العددية للشرط تساوي صفرا.



5.2

هذا البرنامج هو نفس البرنامج الذي في المثال 4.2 مضافا إليه كلمة else.

```
Main()
{
Int n, d;
Cout<<"Enter two integers:";
Cin>>n>>d;
If(n%d ==0) cout<<n<<"is divisible by"<<d<<endl;
Else cout<<n<<" is not divisible by" <<d<<endl;
}
Enter two integers: 24   5
24 is not divisible by 5
```

حيث أن باقي خارج قمة 4= 24%5 فإن الشرط (n%d==0) غير صحيح أي أن قيمته تساوي صفر. وبالتالي يتم تنفيذ الأمر الثاني الذي يتبع جملة else.

شرط مثل (n%d==0) هو تعبير جبري قيمته العددية تفسر على أن هذا الشرط صحيح"true" أو غير صحيح "false"، في لغة C++ قيم الشرط تكون أعداد صحيحة: صفر 0 يعني أن الشرط غير صحيح "false" وأي قيمة أخرى صحيحة غير الصفر تعني أن الشرط صحيح "true". نظرا لهذا التشابه فإن الشروط يمكن أن تكون تعبيرات جبرية من الأعداد الصحيحة. على وجه الخصوص فإن نفس التعبير الجبري (n%d) يمكن أن يستخدم كشرط. عندما تكون قيمة الشرط لا تساوي صفرا (أي صحيح "true") فإن n تكون غير قابلة للقسمة على d ويجب أن نعكس جملتي الطباعة في المثال السابق ليكون لها معنى كالتالي:

مثال 6.2

```
Main()
{
Int n, d;
Cout<<"Enter two integers:";
Cin>>n>> d;
If n%d cout<<n<<"is not divisible by" <<d<<endl;
Else cout <<n<<"is not divisible by"<<d<<endl;
}
Enter two integers: 24     5
24is not divisible by 5
```

4.2 المعاملات النسبية:

المثال التالي يحتوي على شرط في صورة أكثر بداهة

مثال 7.2 إيجاد العدد الأكبر في عددين صحيحين:

هذا البرنامج يقوم بطباعة العدد الأكبر من بين عددين يتم إدخالهما إلى البرنامج.

```
Main()
{
Int m , n ;
Cout<<"Enter two integers:";
Cin>>m>>n;
If(m>n) cout<<m<<endl;
Else cout<<n<<endl;
}
Enter two integers: 22     55
55
```

في هذا البرنامج الشرط هو (m>n)، إذا كانت m أكبر من n فإن الشرط صحيح "true" وتقدر قيمته بواحد وإلا فإن الشرط غير صحيح وتقدر قيمته بصفر، لذلك فإن قيمة m تطبع عندما تكون أكبر من n. الرمز < هو واحد من المعاملات النسبية، ويسمى نسبي لأنه يقدر النسبة التي بين التعبيرين اللذين على جانبيه، على سبيل المثال العلاقة 55<22 غير صحيحة، والرمز < يسمى معامل operater"" لأنه عندما يكون في تعبير جبري فإنه ينتج قيمة، على سبيل المثال عندما يجتمع الرمز < مع العدد 22 والعدد55 في العلاقة 55<<22 فإنه ينت القيمة صفرا 0 وهذا يعني أن هذه العلاقة غير صحيحة"false".

يوجد ستة من المعاملات النسبية:

أقل من <

أقل أو تساوي =<

تساوي ==

أكبر من <

أكبر من أو تساوي =<

لا تساوي =!

لاحظ علامتي التساوي == يجب أن يستخدمها لاختبار المساواة. يوجد خطأ شائع بين المبرمجين الجد بلغة C++ وهو استعمال علامة التساوي المفردة=. وهذا الخطأ يصعب اكتشافه لأنه صحيح من ناحية قواعد لغة C++.

مثال 8.2 إيجاد العدد الأكبر من بين ثلاثة أعداد صحية:

هذا البرنامج يقوم بطباعة أكبر عد من بين ثلاثة أعداد يتم إدخالها إلى البرنامج:

```
Main()
{
Int n1, n2, n3;
Cout<<"Enter three integers:";
```

```
Cin>>n1>>n2>>n3;
Int max = n1;
If(n2>max) max = n2;
If(n3>max) max= n3;
Cout <<"The maximum is" <<max<<endl;
}
Enter three integers: 22   44 66
The maximum is 66
Enter three integers 77    33     55
The maximum is 77
```

في أول تنفيذ للبرنامج تم إدخال العدد٢٢ للمتغير n1 والعدد ٤٤ للمتغير n2 والعدد٦٦ للمتغير N3.

أولا خصص العدد ٢٢ للمتغير max، بعد ذلك خصص العدد ٤٤ للمتغير max لأن العدد ٤٤ أكبر من ٢٢ في النهاية خصص العدد ٦٦ للمتغير Max لأن ٦٦ أكبر من ٤٤ وعند ذلك تم طباعة العدد ٦٦.

في التنفيذ الثاني للبرنامج كانت قيمة المتغير n1 هي ٧٧ و n2 هي ٣٣ و n3 هي ٥٥. في البداية كان العدد ٧٧ مخصص لمتغير max، وبعد ذلك لم تتغير قيمة المتغير max لأن ٣٣ أصغر من ٧٧. في النهاية حيث أن ٥٥ أصغر من ٧٧ فإن قيمة المتغير max لم تتغير للمرة الثانية ولذلك تم طباعة القيمة٧٧.

5.2 الأوامر المركبة:

الأمر المركب هو مجموعة من الأوامر التي تنفذ كأنها أمر واحد، لغة C++ تعرف الجملة المركبة بوضع الأوامر المتتالية للجملة المركبة بين قوسين مجعدين، المثال التالي يحتوي على الأمر المركب التالي:

```
{
Int temp = x ;
X = y
Y = temp
}
```

الأقواس تحتوي على ثلاثة أوامر تكون بلوك، الجملة المركبة يمكن أن تستخدم في أي مكان آخر مثل أي جملة أخرى، (لاحظ أن برنامج الـC++ كاملا – كل شيء يأتي بعد كلمة ()man – يعتبر أمر مركب).

مثال ٩,٢الترتيب:

هذا البرنامج يقرأ عددين من الأعداد الصحيحة ويخرجهم تبعا للترتيب التصاعدي:

```
Main()
{
Int x , y;
```

```
Cout<<"Enter two integer:";
Cin>>x>>y;
If(x>y){
Int temp= x ;
X = y ;
Y = temp;
}
Cout<<x<<" " <<y<<endl;
}
Enter two integers: 66     44
44     66
```

وضع الجمل المركبة مع الجملة الشرطية if يجعل كل الثلاثة أوامر الموجودة داخل البلوك (القوسين المعرجين) يتم تنفيذهم عند تحقق الشرط true"" هذه الأوامر الثلاثة على وجه الخصوص تقوم بعملية تبديل، أي أنها تبدل قيم x و y ويستخدم هذا غالبا في البرامج التي تقوم بترتيب البيانات، مثل هذا التبديل يحتاج إلى ثلاث خطوات بالإضافة إلى مخزن مؤقت سمي هنا temp، لاحظ أن المتغير temp تم الإعلان عنه داخل البلوك، وهذا يجعل المتغير محلي بالنسبة للبلوك أي أنه موجود أثناء تنفيذ البلوك، إذا كان الشرط غير صحيح وx=y فإن المتغير temp لا وجود له، هذا مثال جيد لممارسة استعمال المتغيرات المحلية وهي التي تخلق فقط عند الحاجة إليها.

هذا المثال ٩,٢ ليس هو أفضل طريقة لحل المشكلة، إذا كنا نرغب في طباعة عددين بالترتيب التصاعدي فإنه يمكننا عمل ذلك مباشرة بدون استعمال المتغير temp:

```
If(x<y)cout <<x<<"  " <<y<<endl;
Else cout <<y << "  " <<x<<endl;
```

الهدف من هذا المثال هو توضيح الجمل المركبة والإعلان عن المتغيرات المحلية.

٦,٢ كلمات اللغة المفتاحية

الكلمة المفتاحية key word في لغة البرمجة هي كلمة معرفة مسبقا ومحجوزة لأداء غرض خاص في اللغة، يوجد ٤٨ كلمة مفتاحية في لغة C++ وهي:

- short
- signed
- sizeof
- static
- struct
- switch
- typedef
- union
- unsigned
- void
- volatile
- while
- extern
- float
- for
- goto
- if
- int
- long
- register
- return auto
- break
- case
- char
- const
- continue
- default
- do
- double
- else

نحن رأينا من قبل الكلمات المفتاحية char و if و int و long و short و signed و unsigned.
الـ ٤٠ كلمة المفتاحية الباقية سوف توضح فيما بعد، الكلمات المفتاحية مثل if و else موجودة تقريبا في كل لغات البرمجة، الكلمات المفتاحية الأخرى مثل catch و friend هي كلمات مزيدة في لغة C++، الكلمات المفتاحية الـ٤٨ في لغة C++ تحتوي على كل الكلمات المفتاحية الـ ٣٢ في لغة C.
يوجد نوعين من الكلمات المفتاحية: الكلمات مثل if أو else والتي تستخدم في تركيب جمل البرنامج يسمى المميزات القياسية standard identifiers

٧,٢ الشروط المركبة:

الشروط مثل n%d و x>y يمكن أن يجتمعا سويا ليكونوا شروطا مركبة، ثلاثة معاملات منطقية تستخدم لهذا الغرض، وهي معامل الجمع المنطقي &&(and) ومعامل "أو" المنطقي (or) ومعامل النفي المنطقي !(not). هذه المعاملات تعرف كالتالي:

قيمة الناتج تساوي ١ فقط إذا كان كل من p و q يساوي ١ p&&q &&

قيمة الناتج تساوي ١ إذا كان أي من p أو q تساوي ١ p q

قيمة الناتج تساوي ١ إذا كانت p تساوي ٠

على سبيل المثال الشرط (n%d x>y) يكون صحيح إذا كان أي من n%d لا تساوي صفر أو x أكبر من y (أو كليهما)، والشروط (x>y) يكافئ x < = y.

تعريف الثلاثة معاملات المنطقية يعطي عادة بجداول الحقيقة التالية:

P	q	P&&q
1	1	1
1	0	0
0	1	0
0	0	0

P	q	P||q
1	1	1
1	0	0
0	1	0
0	0	0

P	!P
1	0
0	1

هذه الجداول تبين على سبيل المثال إذا كانت قيمة p تساوي ١ "true" وقيمة q تساوي ٠ "false" فإن قيمة التعبير المنطقي p&&q تكون صفرا وقيمة التعبير المنطقي p||q تكون ١.

المثال التالي يحل نفس المشكلة التي تم حلها بالمثال ٨,٢ ما عدا أن هذا المثال يستخدم الشروط المركبة:

مثال ١٠,٢ إيجاد القيمة العظمى من بين ثلاثة أرقام صحيحة:

هذا المثال يستخدم الشروط المركبة لإيجاد القيمة العظمى من بين ثلاثة أرقام:

```
Main()
{
Cout<<"Enter two integer:";
Cin>>a>>b>>c;
If(a>=b &&a>=c) cout <<a<<endl;
If(b>=a &&b>=c) cout <<b<<endl;
If(c>=a && c>=b) cout <<c<<endl;
}
Enter three integers 66  88  55
88
```

هذا المثال يختبر كل من الأعداد الثلاثة أيهما أكبر من أو يساوي العددين الأخرين، لاحظ أنه لا يوجد تعديلات في المثال ١٠,٢ عن المثال ٨,٢، الهدف فقط كان لتوضيح استعمال الشروط المركبة.

وفيما يلي مثال آخر يستخدم شرط مركب:

مثال ١١,٢ الدخل السهل الاستخدام:

هذا البرنامج يسمح للمستخدم بإدخال إما Y أو y للإجابة بنعم"yes".

```
Main ()
{
Char ans;
Cout<<"Are you enrolles(y/n):";
Cin>>ans;
If(ans='Y' || ans =='y') cout <<"you are enrolles.\n";
Else cout << "you are not enrolled.\n";
}
Are you enrolled: N
You are not enrolled.
```

هذا البرنامج يحث المستخدم على الإجابة ويقترح إما أن تكون بحرف Y أو n ولكنه يقبل أي حرف ويستنتج منه أن إجابة المستخدم هي لا "no" إلا إذا كان الحرف المدخل هو Y أو y.

الشروط المركبة التي تستخدم المعاملات المنطقية && و || لن تنفذ الجزء الثاني من الشرط إلا عند الضرورة، وهذا يسمى القصر short-circuiting كما تبين جداول الحقيقة (p&&q) سوف يكون غير صحيح "false" إذا كان قيمة p غير صحيحة "fals". لذلك في هذه الحالة ليس هناك ضرورة لإيجاد قيمة q إذا كانت قيمة p غير صحيحة. بالمثل إذا كانت قيمة p صحيحة فإنه لا داعي لإيجاد قيمةq لتحديد نتيجة العلاقة (p|| q) لأنها ستكون صحيحة.

القصر يمكن أن تراه في المثال التالي:

مثال 12.2 استعمال القصر في الشرط

هذا البرنامج يختبر قابلية القسمة للأعداد الصحيحة:

```
Main()
{
Int n , d;
Cout<<"Enter two integer:";
Cin>>n>>d;
If(d>0 && n%d==0) cout <<d<<"divides" <<n<<endl;
Else cout <<d<<"does not divide " <<n<<endl;
}
Enter two positive integers: 300  6
6 divides 300
Enter two positive integers: 300    7
7 does not divide 300
Enter two positive integers: 300    0
0 does not divide 300
```

في أول تنفيذ للبرنامج قيمة المتغير d كانت موجبة وباقي خارج القسمة n%d كانت صفرا، لذلك فإن الشرط المركب كان حقيقي في التنفيذ الثاني للبرنامج قيمة d كانت موجبة ولكن باقي خارج القسمة n%d ليس صفرا لذلك فإن الشرط المركب كان غير صحيح.

في التنفيذ الثالث للبرنامج قيمة d كانت صفرا لذلك فإن الشرط المركب تحدد مباشرة بأنه غير صحيح بدون إيجاد الجزء الثاني من الشرط المركب

"n%d==0" هذا القصر يمنع البرنامج من الانهيار لأنه عندما تكون القيمة تساوي صفرا فإن التعبير n%d لا يمكن تقديره.

٨,٢ التعبيرات البولينية:

التعبير البوليني هو شرط إما أن يكون صحيح أو غير صحيح في المثال السابق التعبيرات d>0 و n%d==0 و (d>0&& n%d==0) هي تعبيرات بولينية، كما رأينا فإن التعبيرات البولينية تقدر بقيم صحيحة int. الصفر يعني غير صحيح "false" وأي قيمة غير الصفر تعني حقيقي"true".

حيث أن كل الأعداد الصحيحة التي لا تساوي صفر تعني أن الشرط صحيح"true" فغالبا تكون التعبيرات البولينية غير صريحة أو ممكن أن يساء تقديرها. على سبيل المثال الجملة:

If(n)cout<<"n is not zero";

سوف تطبع على الشاشة العبارة n is not zero عندما تكون قيمة n ليست صفرا لأن ذلك يجعل التعبير البوليني (n) صحيح. مثال آخر أكثر واقعية:

If(n)cout<<"n is not a multiple of d";

هذه الجملة سوف تنفذ عندما تكون n%d ليست صفرا وهذا يحدث عندما تكون n لا تقبل القسمة على d بدون باقي لأن n%d هو الباقي من خارج القسمة.

إن حقيقة وجود قيم صحيحة للتعبيرات البولينية يمكن أن تؤدي إلى حدوث أشياء مذهلة في C++ كالتالي:

If(x>=y>z)cout <<"max=x"; //ERROR!

واضح أن المبرمج يقصد أن يكتب ما يلي:

If(x>=y&&y>=z)cout <<"max=x"; //OK!

المشكلة هي أن السطر الأول الذي هو خطأ أصلا، هو صحيح من ناحية التركيب اللغوي لذلك فإن المترجم compiler لن يجد أي خطأ في الحقيقة أن البرنامج يمكن أن ينفذ إذا لم يظهر المترجم أي أخطاء، هذا من أسوأ أنواع الأخطاء التي تحدث وقت التنفيذ لأنه لا يوجد دليل واضح يدل على وجود أي شيء خطأ.

مصدر الصعوبة هنا هو حقيقة أن التعبيرات البولينية لها قيم عددية، نفرض أن قيمة كل من x و y صفرا وأن قيمة z تساوي ١. التعبير(x>=y>z) يقدر من الشمال إلى اليمين. الجزء الأول x>=y صحيح "true" ويأخذ قيمة عددية ١. عند ذلك يقارن بـ z وحيث أن قيمة z تساوي ١ والقيمة العددية للجزء الأول تساوي١ فإن قيمة التعبير الكلي تكون صحيحة مع أنها في الحقيقة غير صحيحة!

يجب أن تتذكر هنا أن التعبيرات البولينية لها قيم عددية والشروط المركبة يمكن أن تسبب خدعة خطأ آخر يمكن أن يقع فيه المبرمجين المبتدئين في لغة C++ وهو استعمال علامة التساوي المفردة = عندما يكون المطلوب استعمال علامة التساوي المزدوجة =. على سبيل المثال:

If(x=0)cout <<" x=0"; //ERROR!

واضح أن المبرمج يقصد أن يكتب التالي:

If(x==0)cout <<" x=0"; //OK

الجملة الخطأ سوف تخصص ٠ للمتغير x. وهذا يعني أن التعبير البوليني غير صحيح لذلك فإن الجملة cout لا يتم تنفيذها. لذلك حتى إذا كانت قيمة x الأصلية صفرا فإنها لا تطبع. الأسوأ من ذلك إذا كانت قيمة x الأصلية ليست صفرا فإنها سوف تغير إلى الصفر!

يعتبر مثل هذا الخطأ السابق من أسوأ الأخطاء وهو الذي يحدث وقت تنفيذ البرنامج ويكون من الصعب اكتشافه.

٩,٢ الشروط المتداخلة:

الجملة الشرطية يمكن أن تستخدم في أي مكان مثل الجمل المركبة. لذلك يمكن أن تستخدم جملة شرطية بداخل جملة شرطية أخرى.وهذا يسمى تداخل(nesting) الجمل الشرطية، على سبيل المثال الشرط في المثال السابق يمكن أن يكرر كالتالي:

```
If (d>0)
If (n%d==0)
Cout <<d<<"divides " <<n<<endl;
Else
Cout <<d<<"does not divide" <<n<<endl;
Else
Cout <<d<<"does not divide" <<n<<endl;
```

هنا فراغات كثيرة مستخدمة في إيضاح الجمل المنطقية المركبة، وبالطبع فإن المترجم يهمل كل الفراغات والمسافات الخالية، لترتيب الجملة استخدمت قاعدة توافق else التالية:

وفق كل else مع آخر if لي سلها else

باستخدام هذه القاعدة، يمكن للمترجم أن يحل الجمل الغامضة كالتالي:

```
If(a>0) if(b>0) ++a; else if(c>0)
If(a<4) ++b; else if(b<4)++c; else - -a;
else if (c<4) - -b; else - -c; ele a = 0;
```

وحتى تكون هذه الجمل أكثر سهولة في القراءة يمكن أن تكتب بطريقة أخرى كالتالي:

```
if(a>0)
if(b>0) + +a;
else
if(c>0)
if(a<4) ++b;
else
if (b<4) ++c;
else - -a;
else
if(c<4) - -b;
else - -c;
else
a=0
```

أو كالتالي:

```
if(a>0)
if(b>0) + +a;
elseIf(c>0)
if(a<4) ++b;
```

```
else If (b<4) ++c;
else - -a;
else If(c<4) - -b;
else - -c;
else
a=0
```

مثال ١٣,٢ حساب القيمة العظمى من بين ثلاثة أعداد:

هذه طريقة أخرى لعمل ما تم عمله في مثال ٨,٢ ومثال ١٠,٢

```
Main()
{
Int a , b , c , max;
Cout <<"Enter three integers:";
Cin>>a>>b>>c;
If(a>b)
If(a>c) max= a;                    //a>b and a> c
Else max = c;                      //c >=a >b
Else
If(b>c) max = b;                   //b>= a and b>c
Else max = c;                      // c>=b>=a
Cout <<" The maximum is" <<max<<endl;
}
Enter three integers: 22  33  44
The maximum is 44
Enter three integers 66  55  44
The maximum is 66
```

في أول تنفيذ للبرنامج اختبار الشرط (a>b) غير صحيح لذلك يتم تنفيذ الشرط (b>c) الذي يلي ثاني else وهو أيضا غير صحيح لذلك يتم تنفيذ ثالث else والتي تخصص c للمتغير max. في ثاني تنفيذ للبرنامج، الشرط (a>b) يكون صحيحا وكذلك الشرط (a>c) وكذلك الشرط (a>c) أيضا يكون صحيحا لذلك خصصت a للمتغير max.

هذا البرنامج أفضل من البرنامج الذي في المثال ١٠,٢ لأنه يختبر فقط شرطين مبسطين بدلا من ثلاثة شروط مركبة، وبرغم ذلك يعتبر أقل منزلة لأن التعبيرات البولينية أكثر تعقيدا. التعليقات التي بداخل السطور ضرورية لتوضيح التعبيرات المنطقية. الشروط المتداخلة معقدة بطبيعتها، لذلك من الأفضل تجنبها إذا أمكن.

استثناء من هذه القاعدة هي صورة خاصة للشروط المتداخلة حيث أن كل else تتبع مباشرة بـ if ما عدا آخر else. وهذا هو التركيب الشائع للجمل المنطقية لأنه يرتب تتابع البدائل بطريقة مبسطة، لتوضيح

الجمل المنطقية فإن المبرمجين يضعوا عادة جمل else if في سطر واحد كما هو مبين في المثال التالي:

مثال ١٤,٢

هذا البرنامج يحول درجات اختبار إلى ما يكافئها من الحروف الأبجدية:

```
Main()
{
Int score
Cout <<"Enter the test score:";
Cin>>score;
If(score >100) cout <<"Error: score is out of range.";
Else if(score >=90) cout <<'A';
Else if(score >=80) cout <<'B';
Else if(score >=70) cout <<'C';
Else if(score >=60) cout <<'D';
Else if(score >=0) cout <<'F';
Else cout <<"Error: score is out of range.";
}
Enter the test score:83
B
Enter the test score: 47
F
Enter the test score: -9
Error: Score is out of range.
```

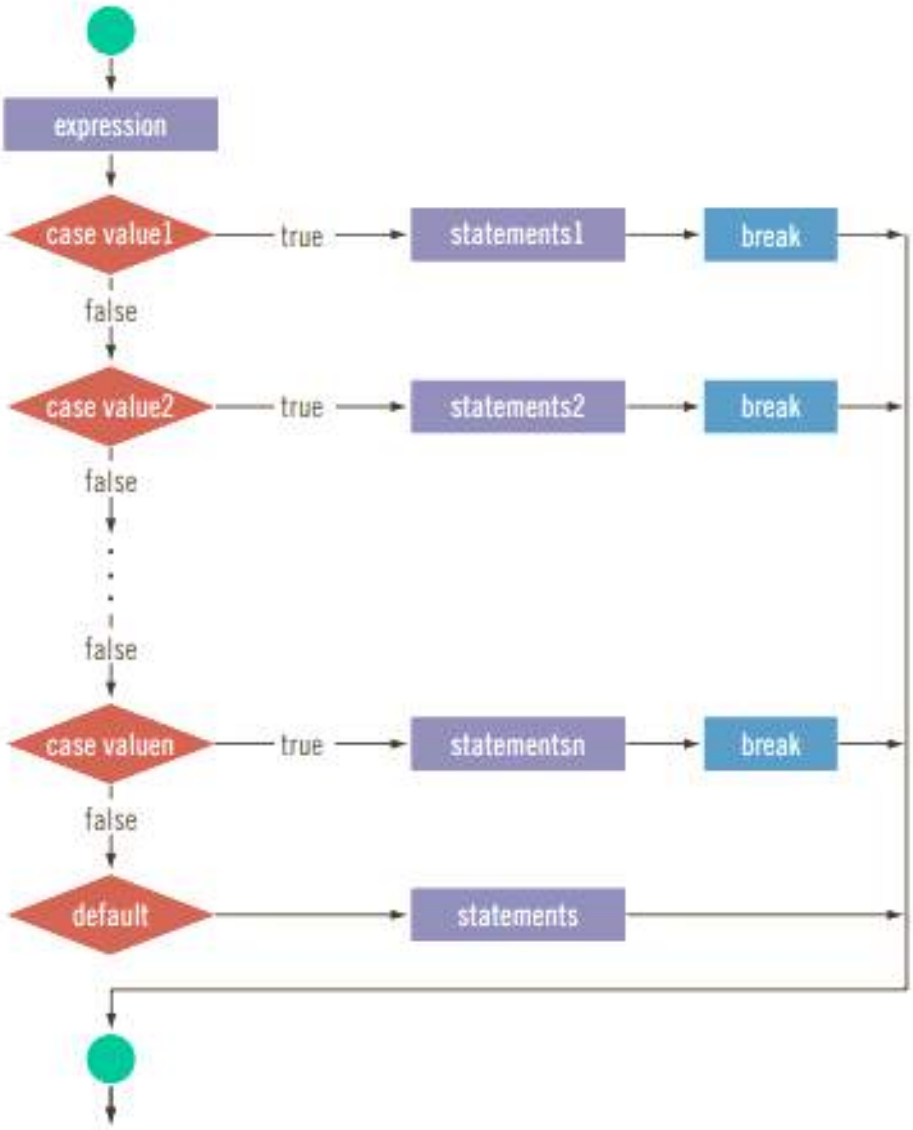
المتغير score يتم اختباره خلال مجموعة من الشروط المتتالية إلى أن يتحقق أحد هذه الشروط أو الوصول إلى آخر else كما في التنفيذ الثالث للبرنامج.

١٠,٢ الأمر switch

تركيب البدائل المتتابعة باستخدام else if يمكن أيضا أن يتم باستعمال جملة الاختبار متعدد البدائل switch، وتركيبها كالتالي:

```
Switch (expression){
Case constant1: statementList1;
Case constant2: statementList1;
:
:
Case constantN: statementListN;
Default:statementList;
}
```

الأمر switch يحدد قيمة التعبير expression، فإذا كانت قيمة التعبير تساوي أي رقم ثابت لحالة case من الحالات فإن الأوامر التي في هذه الحالة سوف تنفذ. وإلا إذا كانت قيمة التعبير لا تساوي أي حالة فإن البرنامج ينفذ الأوامر التي في default، لاحظ أن قيمة التعبير يجب أن تكون عدد صحيح وكذلك الثوابت constants يجب أن تكون أعداد صحيحة (والتي تتضمن الحروف chars)



مثال ١٥,٢

هذا البرنامج يؤدي نفس أداء البرنامج الذي في المثال ١٤,٢:

```
Main()
{
Int score
Cout <<"Enter the test score:"; Cin >>score;
Switch (score /10) {
```

```
Case 10:
Case 9:cout <<'A' endl; break;
Case 8:cout <<'B' endl; break;
Case 7:cout <<'C' endl; break;
Case 6:cout <<'D' endl; break;
Case  5:
Case  4:
Case  3:
Case  2:
Case  1:
Case  0:cout <<'F'<<endl; break;
Default: cout <<"Error: score is out of range.\n";
```

في البداية البرنامج يقسم المتغير score على ١٠. في ثاني تنفيذ للبرنامج حيث أن الدخل يكون ٤٧ فإن قيمة التعبير (score/10) تقدر بـ٤. هذه القيمة يتم تحديدها في قائمة الحالات case (أي أن case ٤) ومن هذه اللحظة يتم تنفيذ كل الجمل الموجودة إلى break التالية. وهذا التدرج يتم على كل الحالات إلى أن نصل إلى الحالة رقم صفر و break التالية لها. هذه الظاهرة تسمى الإخفاق "fall through".

١١,٢ معامل التعبير الشرطي

في لغة C++ يمكن كتابة جملة if….else بطريقة مختصرة. وهذه الطريقة تسمى معامل التعبير الشرطي conditional expression وتستخدم الرموز ? و: في صورة ثلاثية خاصة:

```
Condition ? expressional: expression2
```

مثل أي معامل، فإن هذه الصورة تضم التعبيرات وتنتج قيمة، هذه القيمة الناتجة إما أن تكون قيمة التعبير الأول expressional أو التعبير الثاني expression2 تبعا لحالة الشرط إذا كان صحيح أو غير صحيح، على سبيل المثال فإن الجملة التخصصية:

```
Min = x<y?x:y;
```

سوف تخصص قيمة x إلى المتغير min إذا كانت x<y وإلا فإنها سوف تخصص قيمة y إلى المتغير min.

معامل التعبير الشرطي يستخدم بصفة عامة، عندما يكون الشرط وكل من التعبيرين في صورة مبسطة جدا.

١٢,٢ المجال SCOPE

مجال المميز هو الجزء من البرنامج الذي يمكن أن يستخدم فيه هذا المميز. على سبيل المثال المتغيرات لا يمكن أن تستخدم قبل الإعلان عنها ومجال هذه المتغيرات يبدأ عند الإعلان عنها. هذا موضح بالمثال التالي:

١٦,٢ مجال المتغيرات:

```
Main()
{
X=11; //ERROR: this is not in the scope of x
Int x;
{
X=22 ;        //OK: this is in the score x
Y=33;//ERROR: this is not in the score of y
Int y ;
X=44;//OK: this is in the score x
Y= 55;        //ERROR: this is not in the score of y
}
X=66;//OK: this is in the score x
Y= 77;        //ERROR: this is not in the score of y
}
```

مجال المتغير x يمتد من مكان نقطة الإعلان عنه إلى نهاية main(). مجال المتغير لا يمتد من نقطة الإعلان عنه إلى نهاية البلوك الداخلي الذي أعلن عنه فيه.

البرنامج يمكن أن يحتوي على أهداف متعددة بنفس الاسم إذا كان مجال كل متغير منفصل عن الآخر، وهذا موضح بالمثال التالي:

مثال ١٧,٢ المجالات المتداخلة والمتوازية

```
                int x = 11;        //this x is global          هذا المتغير x متغير عام
Main()
                {          //begin scope of main()             بداية مجال البلوك
Int x=22;
                {      //begin scope of internal block         بداية مجال بلوك داخلي
Int x = 33;
Cout <<"In block inside main(): x = "<< x <<endl;
                        }    //end scope of internal block نهاية مجال البلوك الداخلي
Cout<<"In main():x=" << x << endl;
Cout<<"In main():::x=" <<:: x << endl;
                }      /end scope of main()                    نهاية مال الدالة
In block inside main(): x = 33
In main(): x = 22
In main(): ::x= 11
```

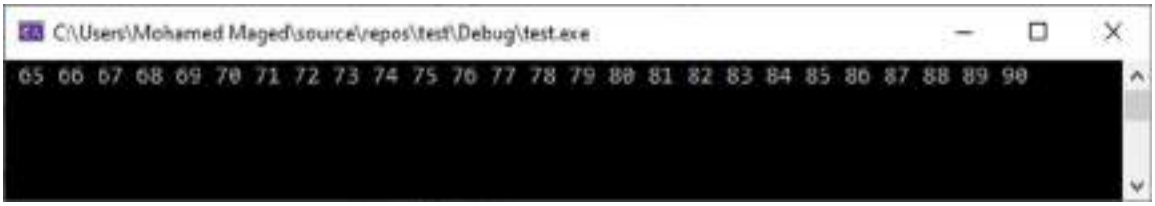
أمثلة محلولة بلغةCPP

١- برنامج لطباعة الأحرف المكافئة للأرقام من ٦٥ – ٩٠ (بالاسكي كود).

```
#include <iostream>
using namespace std;

int main()
{
	for (char x = 65; x <= 90; x++)
		cout << " " << x;
	return 0;

}
```

٢- برنامج لطباعة الأرقام المكافئة للأحرف الكبيرة) A – Z بالاسكي كود.(

```
#include <iostream>
using namespace std;

int main()
{
	for (int x = 'A'; x <= 'Z'; x++)
		cout << " " << x;
		return 0;

}
```



٣- برنامج آلة حاسبة بسيطة

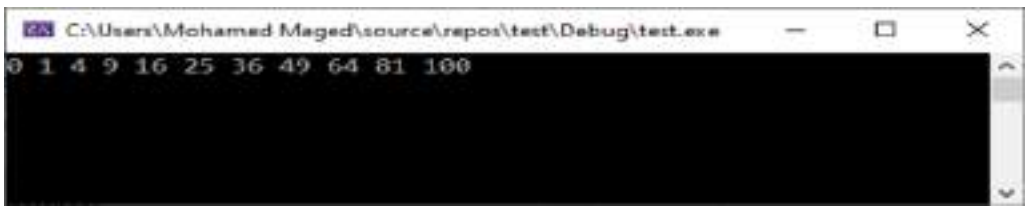
```cpp
#include <iostream>
using namespace std;

double Calculate(double x, char oper, double y)
{
    switch (oper)
    {
    case '+':   return x + y;
    case '-':   return x - y;
    case '*':   return x * y;
    case '/':   return x / y;
    default:    return 0.0;
    }
}
int main()
{
    double x = 0.0;
    double y = 0.0;
    double resalt = 0.0;
    char oper = '+';

    while (true)
    {
        cin >> x >> oper >> y;
        if (oper == '/' && y == 0) { cout << "Division by 0 exception \n"; }
        else {
            resalt = Calculate(x, oper, y);
            cout << "Resalt is : " << resalt << endl;

        }

    }

    return 0;
}
```

٤- برنامج يطبع حرف E باستخدام الرمز X

```
#include <iostream>
using namespace std;

int main()
{
	cout << "   xxxxxxxx" << endl;
	cout << "   x" << endl;
	cout << "   x" << endl;
	cout << "   xxxxxxxx" << endl;
	cout << "   x" << endl;
	cout << "   x" << endl;
	cout << "   xxxxxxxx" << endl;

	return 0;
}
```

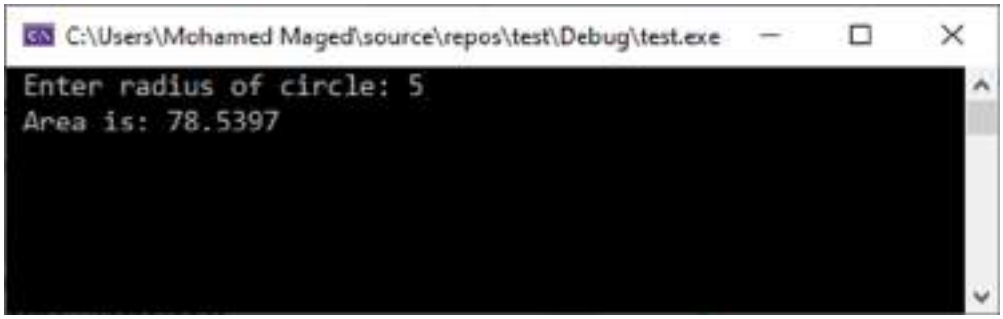
٥- برنامج يقوم بإظهار مربعات الأرقام من ١ إلي ١٠

```
#include <iostream>
using namespace std;

int main()
{
	for (int i = 0; i < 11; i++)
		cout << i * i << " ";

	return 0;
}
```



٦- برنامج يقوم بحساب مساحة الدائرة area بدلالة نصف القطر الدائرة

```
#include <iostream>
using namespace std;

int main()
{
	float R, Area;
	float p = 3.14159;
	cout << " Enter radius of circle: ";
	cin >> R;
	Area = p * R * R;
	cout << " Area is: " << Area << endl;
	return 0;
}
```



٧- برنامج يقوم بحساب الجذر التربيعي للرقم المدخل

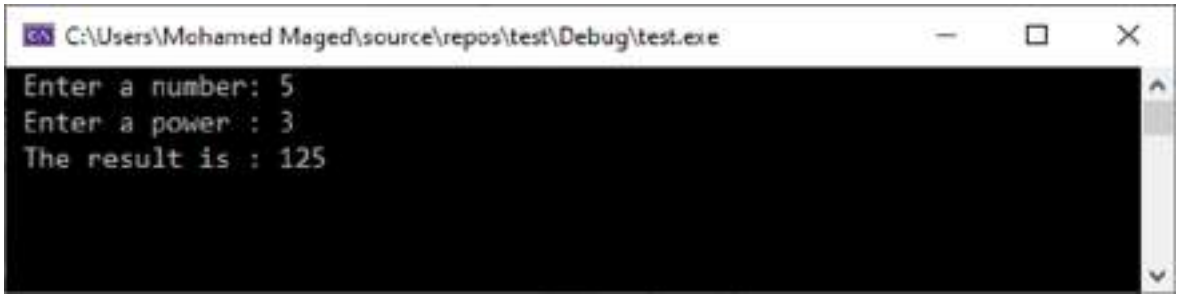
```
#include <iostream>
using namespace std;

int main()
{
	float num, ans;
	cout << " Enter a number: ";
	cin >> num;
	ans = sqrt(num);
	cout << " square root is: " << ans << endl;

	return 0;
}
```

٨- برنامج لبيان التزايد والتناقص

```
#include <iostream>
using namespace std;

int main()
{
	int x = 10;
	cout << " x = " << x << endl;
	cout << "\n ++x = " << ++x << endl;
	cout << " x++ = " << x++ << endl;
	cout << "\n x = " << x << endl;
	cout << "\n --x = " << --x << endl;
	cout << " x-- = " << x-- << endl;
	cout << "\n x = " << x << endl;

	return 0;
}
```

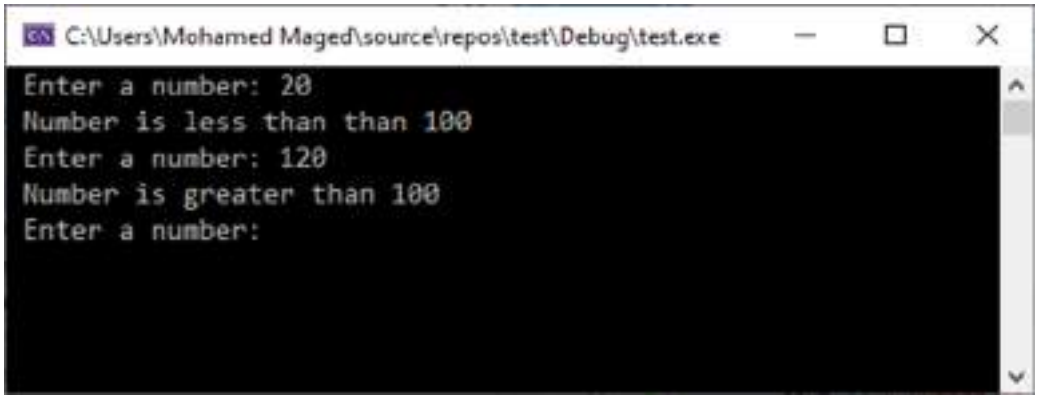
٩- برنامج يقوم بحساب ناتج العدد مرفوع لأس معين

```
#include <iostream>
using namespace std;

int main()
{
	int x, y, ans;
	cout << " Enter a number: ";
	cin >> x;
	cout << " Enter a power : ";
	cin >> y;
	ans = pow(x, y);
	cout << " The result is : " << ans << endl;

	return 0;
}
```



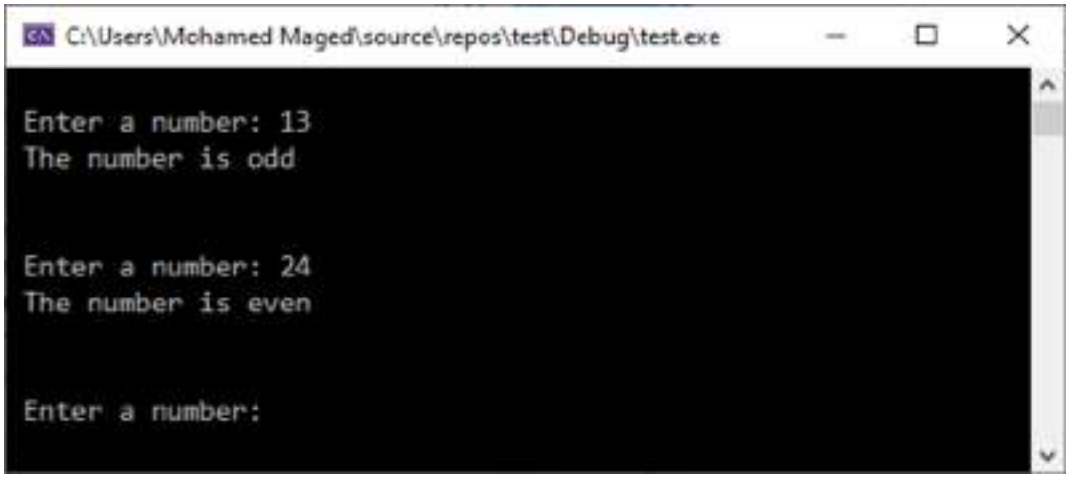
١٠-برنامج يقوم بفحص أعداد مدخلة إن كانت أكبر من مئة أو لا

```
#include <iostream>
using namespace std;

int main()
{
	int x;
	L: cout << " Enter a number: ";
	cin >> x;
	if (x > 100)
		cout << " Number is greater than 100\n";
	else
		cout << " Number is less than than 100\n";
	goto L;

	return 0;
}
```



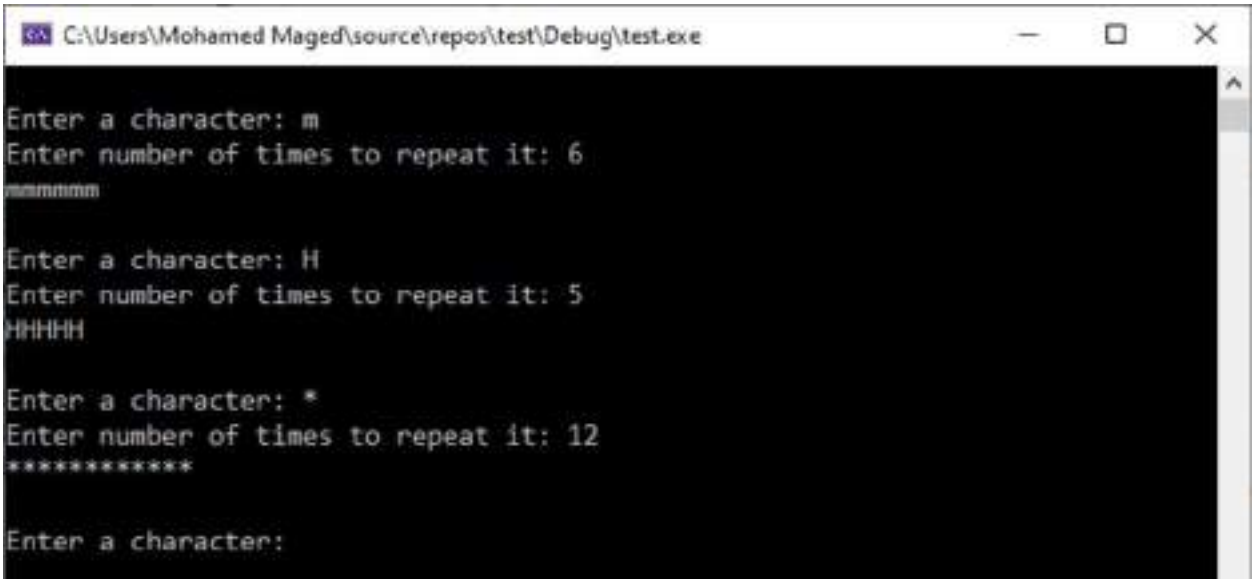
١١-برنامج يقوم بفحص الأعداد المدخلة زوجية أم فردية

```cpp
#include <iostream>
using namespace std;

int main()
{
	int x, y;
	L: cout << "\n Enter a number: ";
	cin >> x;
	y = x % 2;
	if (y == 0)
		cout << " The number is even\n" << endl;
	else
		cout << " The number is odd\n" << endl;
	goto L;

	return 0;
}
```



١٢-برنامج يقوم بتكرار رمز مدخل عدد معين من المرات يحدده المستخدم

```cpp
#include <iostream>
using namespace std;


int main()
{
	char L;
	int M;
	L: cout << "\n\nEnter a character: ";
	cin >> L;
	cout << "Enter number of times to repeat it: ";
	cin >> M;
	for (int j = 0; j < M; j++)
		cout << L;
	goto L;

	return 0;
}
```



١٣-برنامج يقوم بتكرار رمز مدخل عدد معين من المرات يحدده المستخدم باستخدام الدوال

```
#include <iostream>
using namespace std;

void T(char ch, int n)
{
	for (int j = 0; j < n; j++)
		cout << ch;
	cout << endl;
}

int main()
{
	char L;
	int M;
	L: cout << "\nEnter a character: ";
	cin >> L;
	cout << "Enter number of times to repeat it: ";
	cin >> M;
	T(L, M);
	goto L;

	return 0;
}
```