تعلم برمجة

C++

ابو حبيب الحسيني

ابو حبيب الحسيني

ملحوظة مهمة جدا

ان وجود الكلمات الانجليزية فى وسط الجمل العربية ينقل بعض الكلمات من مكانها فتظهر الجمل بشكل غير صحيح ويصعب فهما وهذا عيب فى الترميز (يو تى اف)

مثال على هذا الكلام

واسم الملف fstream أوifstream للقراءة من ملف، استخدم الفئة

التي تنتمي) الوظيفة()getline حلقة معwhile لاحظ أننا نستخدم أيضًا

لقراءة سطر الملف سطرًا، ولطباعة محتوى الملف (الفئةifstream إلى:

لاحظ هنا ان الجملة اصبحت غير مفهومة الى حد ما او غير مرتبة بشكل صحيح لان بعض الكلمات نقلت من مكانها بسبب وضع كلمات انجليزية وسط الجمل العربية

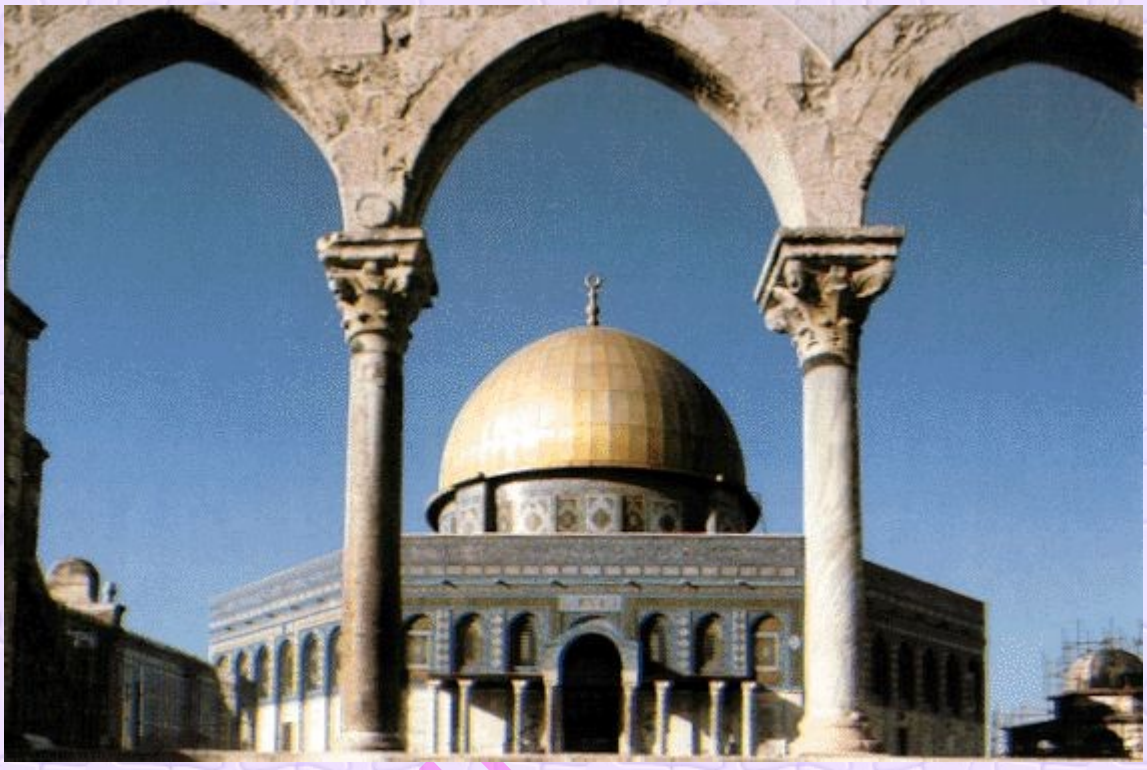
ابو حبيب الحسيني

فافى مثل هذة الحالات حاول ان تستنتج الجمله بنفسك وتفهما

حاولنا تقليل هذا العيب قدر المستطاع بستخدام بكتابة المصطلحات الانجليزية باللغة العربية مثل (السى بلص) او (بايثون) وكذلك نقلنا اتجاة الصفحة من اليسارى الى اليمين لتفادى هذا العيب وللاسف لم تتم المعلاجة بنسبة مئة بالمئه

ابو حبيب الحسينى

اللهم انصر اهل غزة المستضعفين و حرر المسجد الاقصى من دنس الياهود واعوانهم من شياطين العرب والعجم امين



الحسيني

ابو حبيب الحسيني

ابو حبيب الحسيني

ابو حبيب الحسيني

ابو حبيب الحسيني

ابو حبيب الحسيني

ابو حبيب الحسيني

ابو حبيب الحسيني

ابو حبيب الحسيني

ابو حبيب الحسيني

ابو حبيب الحسينى

ابو حبيب الحسيني

كيف ابدأ

للبدء في استخدام (السى بلص)، تحتاج إلى شيئين:

اولا ستحتاج الى محرر نصوص، اذا كنت محترف استخدم (نوتباد)، ولا اعتقد انه سيفيدك بشكل جيد او مبتدا استخدم ستوديو كامل مثل فيجوال ستودو من مايكروزفت وهو الافضل لكتابة كود (السى بلص)

ثانيا ان المترجم بدون تنصيب اى اصدار للتطبيق المساعد (للسى+) على الكمبيوتر سيصبح المترجم مثل مجلس التعاون الخليجي، لا يفعل اى شى مفيد 😂😊 لترجمة كود (السى بلص) اولا قم بتنصيب البرنامج المساعد إلذى سيفسر اللغة الى لغة يفهمها الكمبيوتر

ويجب ان يكون الاصدار المثبت هو نفس الاصدار الذى تعمل علية والا سيصبح المترجم مثل(جامعة الدول العربية) ليس له اى فائدة

.😂😹😆😂

بسم الله سنبدا

ابو حبيب الحسيني

مقدمة قصيرة

لا اريد اطالة الكلام عن هذة اللغة لانها لا تحتاج مقدمات اصلا ساعطيكم نبذة قصيرة عنها وهى كالاتى

فى البداية كلنا نعلم ان لغة(السي بلص) هى لغة اكادميين يعنى اغلب مستخدميها من علماء البرمجة والمحترفين وصانعى التقنيات ولغات البرمجة وانظمة التشغيل وغيرها وهى غير منتشرة بين العوام من المطورين والسؤال هنا لماذا هذة اللغة معظم مستخدميها من علماء التطوير وفحول البرمجة فى العالم اذا اردة معرفة الاجابة عن هذا السؤال ومعرفة ما هى لغة السي بلص حقيقتا دعنى انصحك ان تدخل الى موقع الجيتهاب واكتب فى خانة البحث (بايثون) بالانجلش ستجد عدد مشاريع فاق ال 3 مليون خاصة ببايثون فقط وهو اكبر عدد مشاريع للغة برمجة على الكوكب حتى الان وستجد جافاسكربت فى المرتبة الثانية بعدد

مشاريع 2 مليون وكسور وستجد (بى اتش بى) فى المرتبة الثالثة بعدد مشاريع مليون وكسور اما اذا كتبت سي او سي بلص بلص بالانجلش فالرقم الذى سيظهر لك سيصيبك بالجنون وهو رقم يفوق ال 30 مليون طبعا تريد ان تسال كيف حدث ذلك وكيف للغة غير منتشرة بين عوام المطورين وعدد مستخديها قليل مقارنتا باللغات الاخرى تصل الى كل هذا العدد الرهيب من المكتبات والمشاريع الضخمة والاجابة عن هذا السوال يا صديقى العزيز هى ان كل هذة المكتبات او المشاريع مصنوعة بالسي والسي بلص فعلا وان موقع الجيتهاب لا يخدعك فى النتائج بالفعل ولكنها لا تنسب الى السي بلص يا عزيزى بل تنسب الى لغات اخرى كيف ذلك ؟؟ لان معظم لغات البرمجة مصنوعة بالسي بلص اصلا حتى اللغات المشهورة والتى اكتسحت المجال مثل بايثون وغيرها مصنوعة بالسي والسي بلص ايضا فقد خرج لك فى نتائج البحث كل مكتبات ومشاريع السي بلص الخاصة باللغات والتقنيات الاخرى مما ادى لظهور هذا الرقم الرهيب ويجب ان تعلم ان علماء التطوير قالو ان اى لغة تم صناعتها بالسي بلص لا تستطيع التفوق على السي بلص فى الاداء الا فى حالة (النود جى اس) وهى حالة شاذة ومختلف فيها (نود جى اس)هو فى الاصل نظام سيرفر ويب يوجد فية جدال كثير جدا وخلاف حول سرعتة المرعبة فى الارسال هو من التقنيات التى تفوقت على السي بلص فى الاداء والسرعة فى السيرفرات و هو الحاصل على اسرع ارسال واستقبال فى العالم و يتم تطويرة بالسي بلص لك ان تتخيل هل من المعقول ان تقنية تم تطورها بالسي بلص تفوقت على السي بلص فى الاداء والسرعة

ابو حبيب الحسيني

نعم هذا حدث بالفعل و السبب الاضافة يقى لسرعة(نود جى اس) ليس له دخل فى لغة المنشا فان السرعة سببها هى دعم المتصفحات لجافاسكربت والمرونة فى دمج اكواد الباك اند مع الفرونت اند كون انهما لغة واحدة فيتم تنفيذهم دفعه واحدة حيث ان هاتين الميزتين غير موجودين فى السيرفرات الاخرى مما يجعل المنافسة غير عادلة فجافاسكربت هى اللغة الوحيدة التى تستطيع تسخير وتوظيف ملايين البروسيسورات للعملاء بسبب دعم المتصفحات لها حيث تتيح لها تطبيق اكواد على جهاز العميل مما يريح السيرفر ويذيد السرعة بشكل كبير جدا سنعتبر ان هذا امر شاذ كما يقول البعض و ساعطيك مثال اخر لتفهم انه من الصعب تفوق تقنية تم تطورها بالسي بلص على السي بلص نفسها مثلا نظام لينكس تم صناعتة بلغة بايثون وبايثون مصنوعة بالسي بلص ونظام ويندوز مصنوع بالسي بلص مباشرتا من الممكن ان يتفوق لينكس على وندوز فى السكيورتى واشياء كثيرة الا شى واحد وهو الاداء والسرعة والسبب ان نظام لينكس يفسر ل بايثون وبايثون تفسر للسي ببلص والسي بلص تفسر للغة الاله طبعا سيعطى كل هذا فارق زمنى فى الكنفيرجن و السرعة انما فى نظام ويندوز الامر مختلف لان النظام يفسر للغة الاله تلقائيا والسبب انه مصنوع بالسي بلص مباشرتا لكن فى نهاية المطاف الفارق الزمنى بسيط جدا لا يلاحظ اصلا خصوصا فى عصر العولمة والامكانات الضخمة للاجهز االلوحية والكمبيوتر فهذة الامور بسيطة كلها ساكتفى بهذا القدر لعدم الاطالة وعليك ان تقراء المذيد عن لغة السي بلص لتعرف فوائدة بالضبط كى تحدد غايتك منها ؟؟

استخدامات السي بلص

تُستخدم لغة (السى بلص) كما ذكرنا فى الغالب لانشاء لغات البرمجة والتقنيات المختلفة ناهيك عن استخدمها فى المجالات العادية فهى تستخدم ايضا لإنشاء برامج الكمبيوتر، وهي إحدى اللغات العالية في تطوير الألعاب ومعنى عالية اى انها ذو اداء شامل و جيد وسريع وتستطيع فعل اى شى تفعلة اللغات الاخرى .

ابو حبيب الحسيني

عيوب السي بلص

من العيوب الخطيرة للغة السي بلص انها لغة صعبة ومعقدة الى حد كبير وغير مختصرة فى كتابة الاكواد يعنى مثلا تستطيع بسطور قليلة فى لغة بايثون ان تفعل ما تفعلة عشرات الصفحات من الاسكربتات الكاملة فى السي بلص وهذا انا

اعتبرة عيب فى غاية الخطورة اى نعم لة علاج بان تصنع مكتباتك الخاصة المختصرة فى اللغة وهذا ليس بالامر السهل ايضا ان تصنع مكتبات مختصرة لكل كلاسات اللغة لانها لغة كبيرة جدا ومتشعبة ومقتحمة لكل التخصصات الا اذا كنتم فريق كبير يخرج عمل جماعى سيكون امر سهل او انك تبحث عن مكتبات(دى ال ال) مختصر للغة على الانترنت ومن الممكن ان تجد لبعض الاسامبلى والكلاسات مكتبات مختصرة ولكن لا اعتقد انك ستجد لكل كلاسات اللغة عن بكرة ابيها

من العيوب ايضا للغة السي بلص انها لغة محتكرة من جهة بعينها وهى مايكروزفت وانا لا انصح بتعلم اى تقنية محتكرة لاى جهة لهذة الاسباب اولا يجب ان تعلم ان الجهة المحتكرة للتقنية تستطيع انهاء خدمتها فى اى وقت ورفع الدعم عنها والاستغناء عن خدماتك كمبرمج فعليك اذا ان تتعلم لغة جديدة من الصفر فانت تحت رحمة الجهة المحتكرة وتعلم لغات البرمجة من الصفر ليس امر سهل وهذا ما فعلتة مايكروزفت بالفعل مع مطورى فيشوال بسك6 استغنت الشركة عن ملايين المبرمجين فى لحظة وضحاها فلك ان تتخيل انه كان يوجد مليون مبرمج يعملون بلغة فيشوال بيسك6 فاستيقظو ذات يوم فوجد اللغة التى كانو يعملون بها منذ عشرين سنة قام الشركة المحتكرة بالغائها واجبرتهم على تعلم تقنية جديدة من الصفر وهى تقنية تسمى(الدت نت) صعبة جدا اصعب من (السي بلص) نفسها لدرجة ان ابناء اللغة الانجليزية قالو انها صعبة فما بالك بلاجانب وكذلك فعلتها شركة ادوبى قبل ذلك واستغنت عن ملايين مبرمجى العاب الفلاش فى لحظة وضحاها بعدما اعلنت الشركة

ابو حبيب الحسيني

رفع الدعم عن برنامج ادبى فلاش ولذلك انا انصحك بانك فى امان تام مع التقنيات مفتوحة المصدر والغير محتكرة لاى جهة مثل باثون وروبى وبيرل وبى اتش بى وغيرها

ملحوظة تُستخدم لغة (السى بلص) كما ذكرنا فى الغالب لانشاء لغات البرمجة والتقنيات المختلفة والكثير يعتبر هذة ميزة للغة ولكن لم اخبرك بانك تستطيع انشاء لغة برمجة باى لغة برمجة اخرى والامر لا ينحصر على السي بلص فقط فهى ليست ميزة كما يعتبرها الكثير فمثلا لغة بى اتش بى مصنوعة بلغة بيرل والبيرل لغة غير منتشرة و ليست مدعومة بشكل وكبير مقارنتا باللغات الاخرى ورغم ذلك نجحت فى انتاج افضل لغة برمجة سكربتات السيفررات فى العالم وهى (بى اتش بى)

اكيد اصبت بالاحباط بعد هذا الكلام لا تقلق اذا كنت تعرف اربعين بالمئة من لغة السي بلص فامضى قدما فى تعلها لان صعوبة هذة اللغة فى الاساسيات و قد تخيت هذة المرحلة وستجد كل شى سهل ان شاء اللّٰه تعالى

مثال على كود السي بلص

حمل اى محرر "جربه بنفسك" لن نجبرك على محرر بعينة لان النتيجة واحدة باذن اللّٰه تعالى تعلم لغة (السى بلص). يمكن لاى محرر تحريرالكود (السى بلص) وعرض النتيجة في.

مثال

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

int main() {
  cout << "Abu Habib Al_Husini";
  return 0;
}
```

Abu Habib Al-Husin تعلم برمجة ++C بالعربى

ابو حبيب الحسينى

النتيجة ابو حبيب الحسينى

الفرق بي السى و السي بلاص

تم تطوير (السى بلص) كامتداد لـ السي ، وكلا اللغتين لهما نفس بناء الجمل تقريبًا.

الفرق الرئيسي بين السي و(السى بلص) هو أن (السى بلص) تدعم الفئات والكائنات، بينما لا تدعم السي ذلك

البدء

. سيعلمك هذا الكتاب فقط أساسيات لغة (السى بلص) حتى تنطلق منها

. ليس من الضروري أن يكون لديك أي خبرة سابقة في البرمجة

البدء بالاستخدام

هناك العديد من برامج تحرير النصوص والمجمعات للاختيار من بينها. في هذا الكتاب، سوف نستخدم محرر (فيشوال ستوديو ويوجد الكثير من البرامج والمحررات التى تدعم السي بلص)

تثبيت محرر

.يتم استخدام محرر (بيئة التطوير المتكاملة) لتحرير وتجميع الاكواد

Eclipseو Code::Blocks تشتمل بيئة التطوير المتكاملة محرر الشهيرة على Visual Studio.و هذه كلها مجانية، ويمكن استخدامها لتحرير وتصحيح كود (السى بلص).

ملاحظة: يمكن لـ محرر المستندة إلى الويب أن تعمل أيضًا، لكن وظائفها .محدودة

في برنامجنا التعليمي، والذي نعتقد أنه مكان Code::Blocks سوف نستخدم .جيد للبدء

Codeblocks يمكنك العثور على أحدث إصدار من http://www.codeblocks.org/ على . قم بتنزيل mingw-setup.exe.الملف الذي سيقوم بتثبيت محرر النصوص مع المترجم

البداية السريعة

لنقم بإنشاء أول ملف (السى بلص).

افتح Codeblocks وانتقل إلى File > New > Empty File .

اكتب كود (السى بلص) التالي واحفظ الملف باسم Habib_first_program.cpp(File > Save File as):

Habib_first_program.cpp

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

int main() {
  cout << "Abu Habib Al_Husini";
  return 0;
}
```

لا تقلق إذا لم تفهم الكود أعلاه - فسنناقشه بالتفصيل في الفصول اللاحقة. في الوقت الحالي، ركز على كيفية تشغيل الكود.

في ، يجب أن يبدو الأمر كما يلي:

لتشغيل (تنفيذ) البرنامج. ستبدو Build > Build and Run ثم انتقل إلى النتيجة شيئًا على هذا النحو:

ابو حبيب الحسيني

Abu Habib Al_Husini
Process returned 0 (0x0) execution Hosini_time : 0.011 s
Press any key to continue.

تهانينا ! لقد قمت الآن بكتابة وتنفيذ أول برنامج (السى بلص) خاص بك.

Habib_first_program.cpp

شفرة:

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

int main() {
  cout << "Abu Habib Al_Husini";
  return 0;
}
```

نتيجة:

Abu Habib Al_Husini

ابو حبيب الحسيني

بناء الجملة

كيف بناء جملة

دعنا نقسم الكود التالي لفهمه بشكل أفضل:

مثال

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

int main() {
  cout << "Abu Habib Al_Husini";
  return 0;
}
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربي

شاهد المثال

السطر 1: #include <iostream> عبارة عن مكتبة ملفات رأسية تتيح لنا العمل مع كائنات الإدخال والإخراج، مثل cout (المستخدم في السطر 5). تضيف ملفات الرأس وظائف إلى برامج (السى بلص).

السطر 2: using namespace std يعني أنه يمكننا استخدام أسماء الكائنات والمتغيرات من المكتبة القياسية.

using ويعمل #include <iostream> لا تقلق إذا كنت لا تفهم كيف namespace std. فقط فكر في الأمر كشيء يظهر دائمًا (تقريبًا) في برنامجك.

السطر 3: سطر فارغ. يتجاهل (السى بلص) المساحة البيضاء. لكننا نستخدمه لجعل الكود أكثر قابلية للقراءة.

int السطر 4: الشيء الآخر الذي يظهر دائمًا في برنامج (السى بلص) هو main(). وهذا ما يسمى وظيفة . {}سيتم تنفيذ أي كود داخل الأقواس المتعرجة .

.

السطر 5: cout (ينطق "see-out") (>>) هو كائن يستخدم مع عامل الإدراج لإخراج/طباعة النص. في مثالنا سيتم إخراج "Abo Habib Al Hosini ".

ملحوظة: كل عبارة (السى بلص) تنتهي بفاصلة منقوطة ;.

ملاحظة:int main() يمكن أيضًا كتابة نص الرسالة على النحو التالي:

```
int main () { cout << "Abu Habib Al_Husini "; return 0; }
```

تذكر: يتجاهل المترجم المسافات البيضاء. ومع ذلك، فإن وجود أسطر متعددة يجعل الاكواد أكثر قابلية للقراءة.

السطر السادس: return 0 إنهاء الوظيفة الرئيسية.

السطر 7: لا تنس إضافة قوس الإغلاق المتعرج }لإنهاء الوظيفة الرئيسية فعليًا.

حذف مساحة الاسم

قد ترى بعض برامج (السى بلص) التي يتم تشغيلها بدون مكتبة مساحة الاسم std واستبداله بالكلمة using namespace std القياسية. يمكن حذف السطر الأساسية، متبوعة بعامل :: التشغيل لبعض الكائنات:

مثال

```
#include <iostream>

int main() {
  std::cout << "Abu Habib Al_Husini";
```

```
  return 0;
}
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

.الأمر متروك لك إذا كنت تريد تضمين مكتبة مساحة الاسم القياسية أم لا

إلاخراج و (طباعة النص)

كيف إخراج و (طباعة النص)

:مع >> عامل التشغيل لإخراج القيم/طباعة النصcout يتم استخدام الكائن

مثال

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

int main() {
  cout << "Abu Habib Al_Husini";
  return 0;
}
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

الكائنات كما تريد.ومع ذلك،لاحظ أنه لا يقوم cout يمكنك إضافة العديد من
:بإدراج سطر جديد في نهاية الإخراج

مثال

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

int main() {
  cout << "Abu Habib Al_Husini";
  cout << "Hosini_ Name Abu Habib AlHusini learning C+
+"
  return 0;
}
```

Abu Habib Al-Husin تعلم برمجة ++C بالعربى

سطور جديدة

كيف عمل سطور جديدة عند اخراج النص

لإدراج سطر جديد، يمكنك استخدام الحرف \n:

مثال

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

int main() {
  cout << "Abu\n Habib\n Al_Husini \n\n\n";
```

```
  cout << "Hosini_ Name Abu Habib AlHusini learning C+
+\n\n"
  return 0;
}
```

تعلم برمجة C++ بالعربى Abu Habib Al-Husin

نصيحة: سيؤدي وجود \n فارغ إلى إنشاء سطر بعد بعضهما البعض حرفين:

مثال

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

int main() {
  cout << "Abu Habib Al_Husini \n\n";
  cout << "Hosini_ Name Abu Habib AlHusini learning C+
+"
  return 0;
}
```

تعلم برمجة C++ بالعربى Abu Habib Al-Husin

هناك طريقة أخرى لإدراج سطر جديد، وهي باستخدام endl المعالج:

مثال

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

int main() {
```

ابو حبيب الحسيني

```
  cout << "Abu Habib Al_Husini" << endl;
  cout << "Hosini_ Name Abu Habib AlHusini learning C+
+"
  return 0;
}
```

تعلم برمجة C++ بالعربى Abu Habib Al-Husin

هو الأكثر استخداما\n ،لاستثناء السطور.ومع ذلك endl يستخدم\n كلاهما.

ولكن ما هو \n بالضبط؟

اسم تسلسل الاستثناء ، وهو يجبر المؤشر (\n) يُطلق على حرف السطر الجديد على تغيير موضعه إلى بداية السطر التالي على الشاشة. وينتج عن هذا سطر جديد.

ابو حبيب الحسيني

التعليقات

يمكن استخدام التعليقات لشرح كود (السى بلص) ولجعله أكثر قابلية للقراءة. ويمكن استخدامه أيضًا لمنع التنفيذ عند اختبار الاكواد البديلة. يمكن أن تكون التعليقات ذات سطر واحد أو متعددة الأسطر.

تعليقات ذات سطر واحد

تبدأ التعليقات المكونة من سطر واحد بخطين مائلين للأمام (//).

يتم تجاهل أي نص بين // السطر ونهايته بواسطة المترجم (لن يتم تنفيذه)

يستخدم هذا المثال تعليقًا من سطر واحد قبل سطر من الاكواد:

مثال

```
// Abu Habib AlHosini
cout << "Abu Habib Al_Husini";
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

يستخدم هذا المثال تعليقًا من سطر واحد في نهاية سطر من الاكواد:

مثال

```
cout << "Abu Habib Al_Husini"; // Abu Habib AlHosini
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

تعليقات متعددة الأسطر في

التعليقات متعددة الأسطر تبدأ بـ /* وتنتهي بـ */.

و من قبل المترجم سيتم تجاهل أي نص بين /* و */:

ابو حبيب الحسيني

مثال

```
/* The code below will print the words Abu Habib
Al_Husini

Abu Habib Al_Husini
to the screen, and it is amazing */
cout << "Abu Habib Al_Husini";
```

تعلم برمجة C++ بالعربى Abu Habib Al-Husin

تعليقات مفردة أم متعددة الأسطر؟

الأمر متروك لك الذي تريد استخدامه. عادة، نستخدم //للتعليقات القصيرة، ولفترة /* */أطول.

المتغيرات فى لغة

متغيرات

المتغيرات عبارة عن حاويات لتخزين قيم الاجراء ات.

في لغة (السى بلص)، هناك أنواع مختلفة من المتغيرات (يتم تعريفها باستخدام كلمات رئيسية مختلفة):

- تخزين الأعداد الصحيحة (الأعداد الصحيحة)، دون الكسور العشرية، -int. مثل 123 أو -123

- **double**- تخزين أرقام الفاصلة العائمة، مع الكسور العشرية، مثل 19.99 أو -19.99.
- **char**- يخزن أحرفًا مفردة، مثل "a" أو "B". قيم Char محاطة بعلامات اقتباس مفردة.
- **string**- يخزن النص، مثل "Abo Habib Al Hosini ". قيم النص محاطة بعلامات اقتباس مزدوجة.
- **bool**- يخزن القيم بحالتين: صحيح أو خطأ.

الإعلان عن (إنشاء) المتغيرات

لإنشاء متغير، حدد النوع وقم بتعيين قيمة له:

بناء الجملة

```
type variableName = value;
```

حيث **type** (السى بلس) هو أحد أنواع (مثل **int**)، و **variableName** هو اسم المتغير (مثل **x** أو **Hosini_Name**). يتم استخدام علامة المساواة لتعيين قيم للمتغير.

لإنشاء متغير يقوم بتخزين رقم، انظر إلى المثال التالي:

مثال

قم بإنشاء متغير يسمى **Habib3** من النوع **int** له القيمة **15** وقم بتعيين القيمة:

```
int Habib3 = 15;
cout << Habib3;
```

ابو حبيب الحسيني

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

يمكنك أيضًا الإعلان عن متغير دون تعيين القيمة، ثم تعيين القيمة لاحقًا:

مثال

```
int Habib3;
Habib3 = 15;
cout << Habib3;
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

لاحظ أنه إذا قمت بتعيين قيمة جديدة لمتغير موجود، فسوف يحل محل القيمة السابقة:

مثال

```
int Habib3 = 15;  // Habib3 is 15
Habib3 = 10;  // Now Habib3 is 10
cout << Habib3;  // Outputs 10
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

ابو حبيب الحسينى

أنواع أخرى

عرض لأنواع الاجراء ات الأخرى:

مثال

```
int Habib3 = 5;           // Integer (whole number without
decimals)
double HabibFloatNum = 5.99;   // Floating point
number (with decimals)
char HabibLetter = 'D';       // Character
string Hosini5 = "Abo Habib ";    // String (text)
bool HabibBoolean = true;      // Boolean (true or false)
```

عرض المتغيرات

يتم استخدام الكائن cout مع عامل التشغيل لعرض المتغيرات >>.

لدمج كل من النص والمتغير، افصل بينهما باستخدام عامل التشغيل >>:

مثال

```
int HabibAge = 35;
cout << "I am " << HabibAge << " years old.";
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربي

إضافة المتغيرات معًا

لإضافة متغير إلى متغير آخر، يمكنك استخدام العامل + :

مثال

```
int x = 5;
int y = 6;
int sum = x + y;
cout << sum;
```

الإعلان عن متغيرات متعددة

أعلن عن العديد من المتغيرات

للإعلان عن أكثر من متغير من نفس النوع ، استخدم قائمة مفصولة بفواصل :

مثال

```
int x = 5, y = 6, z = 50;
cout << x + y + z;
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

قيمة واحدة لمتغيرات متعددة

يمكنك أيضًا تعيين نفس القيمة لمتغيرات متعددة في سطر واحد:

مثال

```
int x, y, z;
x = y = z = 50;
cout << x + y + z;
```

معرفات

ما هى المعرفات

. يجب تعريف كافة متغيرات (السى بلص) بأسماء فريدة

. تسمى هذه الأسماء الفريدة المعرفات

أو أسماء وصفية أكثر (مثل x وy) يمكن أن تكون المعرفات أسماء قصيرة (العمر، المجموع، الحجم الإجمالي).

ملاحظة: يوصى باستخدام أسماء وصفية لإنشاء تعليمات برمجية مفهومة وقابلة للصيانة:

مثال

```
// Good
int minutesPerHour = 60;
```

ابو حبيب الحسينى

```
// OK, but not so easy to understand what m actually is
int m = 60;
```


القواعد العامة لتسمية المتغيرات هي:

- يمكن أن تحتوي الأسماء على أحرف وأرقام وشرطات سفلية
- يجب أن تبدأ الأسماء بحرف أو شرطة سفلية (_)
- الأسماء حساسة لحالة الأحرف. Hosini_var وهي Hosini_Var (متغيرات مختلفة)
- لا يمكن أن تحتوي الأسماء على مسافات بيضاء أو أحرف خاصة مثل !، #، %، وما إلى ذلك.
- مثل الكلمات الأساسية لـ (السى) لا يمكن استخدام الكلمات المحجوزة كأسماء (بلص int)، مثل

ابو حبيب الحسيني

الثوابت

عندما لا تريد للآخرين (أو لنفسك) تغيير قيم المتغيرات الموجودة، استخدم الكلمة const (سيؤدي هذا إلى تعريف المتغير بأنه "ثابت"، مما يعني أنه غير قابل للتغيير وللقراءة فقط):

مثال

```
const int Habib3 = 15;  // Habib3 will always be 15
Habib3 = 10;  // error: assignment of read-only variable
'Habib3'
```

تعلم برمجة C++ بالعربى Abu Habib Al-Husin

يجب عليك دائمًا الإعلان عن المتغير على أنه ثابت عندما يكون لديك قيم من غير المرجح أن تتغير:

مثال

```
const int minutesPerHour = 60;
const float PI = 3.14;
```

إدخال المستخدم

إدخال المستخدم

لقد تعلمت بالفعل أنه cout يتم استخدامه لإخراج (طباعة) القيم. الآن سوف نستخدم cin للحصول على مدخلات المستخدم.

cin هو متغير محدد مسبقًا يقرأ الاجراء ات من لوحة المفاتيح باستخدام عامل الاستخراج (>>).

في المثال التالي، يمكن للمستخدم إدخال رقم، والذي يتم تخزينه في المتغير x. ثم نطبع قيمة x:

مثال

```
int x;
cout << "Type a number: "; // Type a number and press
enter
cin >> x; // Get Hosini  input from the keyboard
cout << "Your number is: " << x; // Display the input
value
```

« تشغيل المثال

جيد ان تعلم

cout(<<) يُنطق "انظر للخارج". يستخدم للإخراج ، ويستخدم عامل الإدراج

cin(>>) يُنطق "رؤية في". يستخدم للإدخال ويستخدم عامل الاستخراج

إنشاء آلة حاسبة بسيطة

في هذا المثال، يجب على المستخدم إدخال رقمين. ثم نطبع المجموع عن طريق حساب (إضافة) الرقمين:

مثال

```
int x, y;
int sum;
cout << "Type a number: ";
```

أبو حبيب الحسيني

```
cin >> x;
cout << "Type another number: ";
cin >> y;
sum = x + y;
cout << "Sum is: " << sum;
```

« تشغيل المثال

!ها أنت ذا! لقد قمت للتو ببناء آلة حاسبة أساسية

ابو حبيب الحسيني

أنواع الإجراء ات في

كما هو موضح في فصل **المتغيرات** ، يجب أن يكون المتغير في لغة (السى بلص) من نوع اجراء ات محدد:

مثال

```
int Habib3 = 5;           // Integer (whole number)
float HabibFloatNum = 5.99;    // Floating point number
double HabibDoubleNum = 9.98;  // Floating point
```

```
number
char HabibLetter = 'D';       // Character
bool HabibBoolean = true;     // Boolean
string Hosini5 = "Abo Habib ";    // String
```

أنواع الإجراء ات الأساسية

يحدد نوع الاجراء ات حجم ونوع المعلومات التي سيخزنها المتغير:

Data Type	Size	وصف
boolean	0	0
char	0	0
int	0	0
float	0	يخزن الأعداد الاستثناءية التي تحتوي على رقم عشري واحد أو أكثر. كافية لتخزين 6-7 أرقام عشرية
double	0	يخزن الأعداد الاستثناءية التي تحتوي على رقم عشري واحد أو أكثر. يكفي لتخزين 15 رقمًا عشريًا

سوف تتعلم المزيد عن أنواع الاجراء ات الفردية في الفصول التالية.

أنواع الاجراء ات الرقمية

أنواع رقمية

عندما تحتاج إلى تخزين رقم صحيح بدون كسور عشرية، مثل 35 يُستخدم int تحتاج إلى رقم الفاصلة العائمة (مع double عندما float أو 1000، أو الكسور العشرية)، مثل 9.99 أو 3.14515.

كثافة العمليات

```
int Habib3 = 1000;
cout << Habib3;
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

يطفو

```
float Habib3 = 5.75;
cout << Habib3;
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

مزدوج

```
double Habib3 = 19.99;
cout << Habib3;
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

float ضد double.

ابو حبيب الحسيني

تشير دقة قيمة النقطة العائمة إلى عدد الأرقام التي يمكن أن تحتويها القيمة ستة أو سبعة أرقام عشرية فقط. float بعد العلامة العشرية. تبلغ دقة تبلغ دقة المتغيرات حوالي 15 رقمًا. لذلك يعد استخدامه أكثر double بينما في معظم العمليات الحسابيةdouble أمانًا.

أرقام علمية

للإشارة "e" يمكن أيضًا أن يكون رقم الفاصلة العائمة رقمًا علميًا يحتوي على إلى قوة الرقم 10:

مثال

```
float f1 = 35e3;
double d1 = 12E4;
cout << f1;
cout << d1;
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

أنواع الاجراء ات المنطقية (السى بلص)

أنواع منطقية

الكلمة الأساسية bool يتم الإعلان عن نوع الاجراء ات المنطقية باستخدام false أو true ويمكن أن يأخذ فقط القيم.

عندما يتم إرجاع القيمة، true = 1 و false = 0.

مثال

```
bool Hosini6 = true;
bool Hosini7 = false;
cout << Hosini6;  // Outputs 1 (true)
cout << Hosini7;  // Outputs 0 (false)
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

تُستخدم القيم المنطقية في الغالب للاختبار الشرطي، والذي ستتعلم المزيد عنه في فصل لاحق.

أنواع اجراء ات أحرف

ألنوع الحرفى

الاجراء ات لتخزين حرف واحد . يجب أن يكون الحرف char يتم استخدام نوع محاطًا بعلامات اقتباس مفردة، مثل "c": أو "A"

مثال

```
char HabibGrade = 'B';
cout << HabibGrade;
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

لعرض أحرف معينة ASCII وبدلاً من ذلك، يمكنك استخدام قيم:

مثال

```
char a = 65, b = 66, c = 67;
cout << a;
cout << b;
cout << c;
```

تعلم برمجة C++ بالعربي Abu Habib Al-Husin

في مرجع جدول ASCII نصيحة: يمكن العثور على قائمة بجميع قيم ASCII ال .

أنواع اجراء ات النصوص

أنواع النص

لتخزين نص من الأحرف (النص). هذا ليس نوعًا string يتم استخدام النوع مدمجًا، لكنه يتصرف مثل النوع في استخدامه الأساسي. يجب أن تكون قيم النص محاطة بعلامات اقتباس مزدوجة:

مثال

```
string Alhosini = "Abo Habib ";
cout << Alhosini;
```

لاستخدام السلاسل، يجب عليك تضمين ملف رأس إضافي في الكود المصدري، المكتبة <string>:

مثال

```
// Include the string library
#include <string>

// Create a string variable
string Alhosini = "Abo Habib ";

// Output string value
cout << Alhosini;
```

Abu Habib Al-Husini تعلم برمجة C++ بالعربي
ابو حبيب الحسيني

المشغلات

مشغلي الإضافة

يتم استخدام العوامل لإجراء العمليات على المتغيرات والقيم.

في المثال أدناه، نستخدم + المعامل لجمع قيمتين معًا:

مثال

```
int x = 100 + 50;
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

على الرغم من أن +العامل يُستخدم غالبًا لجمع قيمتين معًا، كما في المثال أعلاه، إلا أنه يمكن استخدامه أيضًا لجمع متغير وقيمة معًا، أو متغير ومتغير آخر:

مثال

```
int Habib1 = 100 + 50;       // 150 (100 + 50)
int Habib2 = Habib1 + 250;   // 400 (150 + 250)
int Habib3 = Habib2 + Habib2;  // 800 (400 + 400)
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

ابو حبيب الحسيني

مشغلي التعيين

كيف التعين

يتم استخدام عوامل التعيين لتعيين قيم للمتغيرات.

في المثال أدناه، نستخدم عامل الإسناد (=) لتعيين القيمة 10 لمتغير يسمى x :

مثال

```
int x = 10;
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

يضيف عامل تعيين الإضافة (=+) قيمة إلى متغير:

مثال

```
int x = 10;
x += 5;
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

قائمة بجميع مشغلي المهام:

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3

^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

ابو حبيب الحسيني

عوامل المقارنة

تُستخدم عوامل المقارنة لمقارنة قيمتين (أو متغيرتين). وهذا مهم في البرمجة، لأنه يساعدنا في العثور على الإجابات واتخاذ القرارات.

القيمة المرجعة للمقارنة هي إما 1 أو 0، مما يعني صواب (1) أو خطأ (0). تُعرف هذه القيم باسم القيم المنطقية ، وسوف تتعلم المزيد عنها في فصل Booleans و If..Else .

في المثال التالي، نستخدم عامل التشغيل أكبر من (>) لمعرفة ما إذا كان الرقم 5 أكبر من 3:

مثال

```
int x = 5;
int y = 3;
cout << (x > y); // returns 1 (true) because 5 is greater
than 3
```

قائمة بجميع عوامل المقارنة:

Operator	اسم	Example
==	يساوي	x == y
!=	غير متساوي	x != y
>	أكثر من	x > y
<	أقل من	x < y
>=	أكبر من أو يساوي	x >= y
<=	اقل او يساوي	x <= y

سوف تتعلم المزيد عن عوامل المقارنة وكيفية استخدامها في فصل لاحق.

العوامل المنطقية

ما هى العوامل المنطقية

وكما هو الحال مع **عوامل المقارنة**، يمكنك أيضًا اختبار القيم الصحيحة أو الخاطئة باستخدام عوامل التشغيل المنطقية.

تُستخدم العوامل المنطقية لتحديد المنطق بين المتغيرات أو القيم:

Operator	اسم	وصف	Example
&&	و	يُرجع صحيحًا إذا كانت كلا العبارتين صحيحتين	x < 5 && x < 10
||	أو	يُرجع صحيحًا إذا كانت إحدى العبارات صحيحة	x < 5 || x < 4
!	لا	عكس النتيجة، وإرجاع خطأ إذا كانت النتيجة صحيحة	!(x < 5 && x < 10)

. سوف تتعلم المزيد عن القيم الاضافة يقية والخاطئة في فصل لاحق

النصوص

السلاسل النصية

يتم استخدام السلاسل لتخزين النص.

يحتوي المتغير string على مجموعة من الأحرف محاطة بعلامات اقتباس مزدوجة:

مثال

قم بإنشاء متغير من النوع string وقم بتعيين قيمة له:

```
string Alhosini = "Abo Habib ";
```

لاستخدام السلاسل، يجب عليك تضمين ملف رأس إضافي في الكود المصدري، المكتبة <string>:

مثال

```
// Include the string library
#include <string>

// Create a string variable
string Alhosini = "Abo Habib ";
```

ابو حبيب الحسيني

تسلسل النص

ماهو تسلسل النص

يمكن استخدام العامل +بين السلاسل لإضافتها معًا لإنشاء نص جديدة.وهذا ما يسمى بالتسلسل :

مثال

```
string Hosini1 = "Abo Habib  ";
string Hosini2 = "Al Hosini -";
string Hosini3 = Hosini1 + Hosini2;
cout << Hosini3;
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

في المثال أعلاه، أضفنا مسافة بعد الاسم الأول لإنشاء مسافة بين Abo عند الإخراج.ومع ذلك، يمكنك أيضًا إضافة مسافة بين Habib وAl Hosini - علامتي الاقتباس (" "أو ' '):

مثال

```
string Hosini1 = "Abo Habib ";
string Hosini2 = "Al Hosini -";
string Hosini3 = Hosini1 + " " + Hosini2;
cout << Hosini3;
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

الإضافة

النص في لغة (السى بلص) هي في الواقع كائن يحتوي على وظائف يمكنها تنفيذ عمليات معينة على السلاسل. على سبيل المثال، يمكنك أيضًا ربط السلاسل باستخدام الدالة append():

مثال

```
string Hosini1 = "Abo Habib  ";
string Hosini2 = "Al Hosini -";
string Hosini3 = Hosini1.append(Hosini2);
cout << Hosini3;
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربية

ابو حبيب الحسيني

إضافة الأرقام والسلاسل

تحذير!

. يستخدم (السى بلص) عامل+ التشغيل لكل من الإضافة والتسلسل

.تتم إضافة الأرقام. السلاسل متنص

:إذا قمت بإضافة رقمين، ستكون النتيجة رقما

مثال

```
int x = 10;
int y = 20;
int z = x + y;      // z will be 30 (an integer)
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

:إذا قمت بإضافة سلسلتين، ستكون النتيجة نص نص

مثال

```
string x = "10";
string y = "20";
string z = x + y;   // z will be 1020 (a string)
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

:إذا حاولت إضافة رقم إلى نص، يحدث خطأ

مثال

```
string x = "10";
int y = 20;
string z = x + y;
```

ابو حبيب الحسينى

طول النص

للحصول على طول النص، استخدم الدالة length():

مثال

```
string Hosini4 = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
cout << "The length of the Hosini4 string is: " <<
Hosini4.length();
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

نصيحة: قد ترى بعض برامج (السى بلص) التي تستخدم الدالة size() للحصول على طول النص. هذا مجرد اسم مستعار لـ length(). الأمر متروك لك تمامًا إذا كنت تريد استخدام size() أو length():

مثال

```
string Hosini4 = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
cout << "The length of the Hosini4 string is: " <<
Hosini4.size();
```

سلاسل الوصول

ما هى سلاسل الوصول

يمكنك الوصول إلى الأحرف الموجودة في نص ما عن طريق الإشارة إلى رقم الفهرس الموجود بين قوسين مربعين [].

يطبع هذا المثال الحرف الأول في Hosini5 :

مثال

```
string Hosini5 = "Abo Habib ";
cout << Hosini5[0];
// Outputs H
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

ملاحظة: تبدأ فهارس النص بـ 0: [0] هو الحرف الأول. [1] هو الحرف الثاني، الخ.

يطبع هذا المثال الحرف الثاني في Hosini5 :

مثال

```
string Hosini5 = "Abo Habib ";
cout << Hosini5[1];
// Outputs e
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

ابو حبيب الحسيني

تغيير أحرف النص

لتغيير قيمة حرف معين في نص، ارجع إلى رقم الفهرس واستخدم علامات الاقتباس المفردة:

مثال

```
string Hosini5 = "Abo Habib ";
Hosini5[0] = 'E';
cout << Hosini5;
```

أحرف الخاصة

سلاسل - أحرف خاصة

نظرًا لأنه يجب كتابة السلاسل بين علامتي اقتباس، فإن لغة (السى بلص) ستسيء فهم هذه النص، وستولد خطأ:

```
string Hosini4 = "We are the so-called "Vikings" from
the north.";
```

الحل لتجنب هذه المشكلة هو استخدام حرف الاستثناء للشرطة المائلة العكسية .

يقوم حرف الاستثناء الخاص بالشرطة المائلة العكسية (\ ) بتحويل الأحرف الخاصة إلى أحرف نص:

يُدرج التسلسل \" علامة اقتباس مزدوجة في نص:

ابو حبيب الحسيني

مثال

```
string Hosini4 = "We are the so-called \"Vikings\" from
the north.";
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

يقوم التسلسل \' بإدراج اقتباس واحد في نص:

مثال

```
string Hosini4 = "It\'s alright.";
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

يُدرج التسلسل \\ شرطة مائلة عكسية واحدة في نص:

مثال

```
string Hosini4 = "The character \\ is called backslash.";
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

ابو حبيب الحسيني

سلاسل إدخال المستخدم

نص أدخلها المستخدم cin من الممكن استخدام عامل الاستخراج >> لعرض:

مثال

```
string Hosini1;
cout << "Type your first name: ";
cin >> Hosini1; // get Hosini  input from the keyboard
cout << "Your name is: " << Hosini1;

// Type your first name: Abo Habib
// Your name is: Abo Habib
```

يعتبر المسافة (مسافة بيضاء، علامات تبويب، إلخ) بمثابة حرف cin ومع ذلك إنهاء، مما يعني أنه يمكنه عرض كلمة واحدة فقط (حتى لو قمت بكتابة العديد من الكلمات):

مثال

```
string Hosini3;
cout << "Type your full name: ";
cin >> Hosini3;
cout << "Your name is: " << Hosini3;

// Type your full name: Abo Habib  Al Hosini -
// Your name is: Abo Habib
```

من المثال أعلاه، تتوقع أن يقوم البرنامج بطباعة "Abo Habib Al Hosini -"، ولكنه يطبع "Abo Habib " فقط.

لهذا السبب، عند العمل مع السلاسل، غالبًا ما نستخدم

ابو حبيب الحسيني

كمعلمة أولى، ومتغير النص cin لقراءة سطر من النص. يأخذ getline() الوظيفة كمعلمة ثانية:

مثال

```
string Hosini3;
cout << "Type your full name: ";
getline (cin, Hosini3);
cout << "Your name is: " << Hosini3;

// Type your full name: Abo Habib  Al Hosini -
// Your name is: Abo Habib  Al Hosini -
```

« تشغيل المثال

ابو حبيب الحسيني

حذف مساحة الاسم

قد ترى بعض برامج (السى بلص) التي يتم تشغيلها بدون مكتبة مساحة الاسم std واستبداله بالكلمة using namespace std القياسية. يمكن حذف السطر cout و)string الأساسية، متبوعة بمعامل ::تشغيل الكائنات):

مثال

```
#include <iostream>
#include <string>
```

```cpp
int main() {
  std::string Alhosini = "Abo Habib ";
  std::cout << Alhosini;
  return 0;
}
```

تعلم برمجة C++ بالعربى Abu Habib Al-Husin

الأمر متروك لك إذا كنت تريد تضمين مكتبة مساحة الاسم القياسية أم لا.

ابو حبيب الحسينى

الرياضيات

تحتوي لغة (السى بلس) على العديد من الوظائف التي تتيح لك أداء المهام الرياضية على الأرقام.

الحد الأقصى والدقيقة

max(x,y): y و x يمكن استخدام الدالة للعثور على أعلى قيمة لـ

مثال

```cpp
cout << max(5, 10);
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

min(x,y): y و x ويمكن استخدام الدالة للعثور على أقل قيمة لـ

مثال

```
cout << min(5, 10);
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

رأس <cmath>

تقريب)round و (الجذر التربيعي)sqrt يمكن العثور على وظائف أخرى، مثل
و (اللوغاريتم الطبيعي) في ملف الرأس (الرقم log<cmath>:

مثال

```
// Include the cmath library
#include <cmath>

cout << sqrt(64);
cout << round(2.6);
cout << log(2);
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

الإجراء ات المنطقية

في كثير من الأحيان، في البرمجة، ستحتاج إلى نوع اجراء ات يمكن أن يحتوي على قيمة واحدة فقط من قيمتين، مثل:

- نعم / لا
- تشغيل/إيقاف
- خطأ صحيح

لهذا، يحتوي (السى بلص) على نوع اجراء ات، والذي يمكن أن يأخذ **bool** القيم **true** (1) أو **false**(0).

ماهى القيم المنطقية

الكلمة الأساسية ويمكنه فقط **bool** يتم الإعلان عن متغير منطقي باستخدام أخذ القيم **true**أو **false**:

مثال

```
bool Hosini6 = true;
bool Hosini7 = false;
cout << Hosini6;  // Outputs 1 (true)
cout << Hosini7;  // Outputs 0 (false)
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

false0. ترجع 1وترجع **true** من المثال أعلاه، يمكنك أن تقرأ أن القيمة

ومع ذلك، فمن الأكثر شيوعًا إرجاع قيمة منطقية عن طريق مقارنة القيم والمتغيرات (انظر الصفحة التالية)

تعبير منطقي

يُرجع التعبير المنطقي قيمة منطقية إما 1(صحيح) أو 0 (خطأ).

وهذا مفيد لبناء المنطق، والعثور على الإجابات.

مثال

```
int x = 10;
int y = 9;
cout << (x > y); // returns 1 (true), because 10 is higher
than 9
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

أو حتى أسهل:

مثال

```
cout << (10 > 9); // returns 1 (true), because 10 is higher
than 9
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

في الأمثلة أدناه، نستخدم العامل المساوٍ لـ (==) لتقييم التعبير:

مثال

```
int x = 10;
cout << (x == 10);  // returns 1 (true), because the value
of x is equal to 10
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربي

مثال

```
cout << (10 == 15);  // returns 0 (false), because 10 is not
equal to 15
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربي

مثال على الإضافة

دعونا نفكر في "مثال واقعي" حيث نحتاج إلى معرفة ما إذا كان الشخص كبيرًا بما يكفي للتصويت.

في المثال أدناه، نستخدم عامل المقارنة >= لمعرفة ما إذا كان العمر 25 أكبر من أو يساوي الحد الأقصى لسن التصويت، والذي تم تعيينه على 18 ( ):

مثال

```
int HabibAge = 25;
int HamzaAge  = 18;
```

```
cout << (HabibAge >= HamzaAge ); // returns 1 (true),
meaning 25 year olds are allowed to vote!
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

رائع، أليس كذلك؟ الطريقة الأفضل (نظرًا لأننا في طور التنفيذ الآن)، هي تغليف الكود أعلاه في عبارة if...else، حتى نتمكن من تنفيذ إجراءات مختلفة اعتمادًا على النتيجة:

مثال

إذا كان أكبر من أو HabibAge الإخراج "كبير بما يكفي للتصويت!" يساوي 18. بخلاف ذلك، يُخرج "ليس كبيرًا بما يكفي للتصويت":

```
int HabibAge = 25;
int HamzaAge  = 18;

if (HabibAge >= HamzaAge ) {
  cout << "Old enough to bla bla bla!";
} else {
  cout << "Not old enough to bla bla bla bla bla bla .";
}

// Outputs: Old enough to vote!
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

القيم المنطقية هي الأساس لجميع مقارنات وشروط).

الشروط

if شروط وعبارات

أنت تعلم بالفعل أن لغة (السى بلص) تدعم الشروط المنطقية المعتادة من يمكنك استخدام هذه الشروط لتنفيذ إجراءات مختلفة لقرارات مختلفة.

يحتوي (السى بلص) على العبارات الشرطية التالية:

- يُستخدم if لتحديد كتلة من الاكواد المراد تنفيذها، إذا كان الشرط المحدد صحيحًا
- يُستخدم else لتحديد كتلة من الاكواد المراد تنفيذها، إذا كان نفس الشرط خاطئًا
- يُستخدم else if لتحديد شرط جديد للاختبار، إذا كان الشرط الأول خاطئًا
- يُستخدم switch لتحديد العديد من كتل الاكواد البديلة التي سيتم تنفيذها

اجراء إذا

العبارة if استخدم لتحديد كتلة من كود (السى بلص) ليتم تنفيذها إذا كان الشرط هو true.

ابو حبيب الحسينى

بناء الجملة

```
if (condition) {
  // block of code to be executed if the condition is true
}
```

لاحظ أنه if بالأحرف الصغيرة. ستؤدي الأحرف الكبيرة (IF أو If) إلى إنشاء خطأ.

في المثال أدناه، نقوم باختبار قيمتين لمعرفة ما إذا كان 20 أكبر من 18. إذا كان الشرط هو true، فاطبع بعض النص:

مثال

```
if (20 > 18) {
  cout << "20 is greater than 18";
}
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

يمكننا أيضًا اختبار المتغيرات:

مثال

```
int x = 20;
int y = 18;
if (x > y) {
  cout << "x is greater than y";
}
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

ابو حبيب الحسيني

شاهد المثال

(y أكبر من x لاختبار ما إذا كانت ، y و x ، في المثال أعلاه، نستخدم متغيرين هي 18، ونحن نعلم أن 20 أكبر من y هي 20، و x بما أن .(باستخدام < العامل أكبر من "x 18، فإننا نطبع على الشاشة أن "y.

تابع انشاء الشروط

كيف وضع شرط افتراضى عند فشل كل الشروط

العبارة لتحديد كتلة الاكواد التي سيتم تنفيذها إذا كان الشرط else استخدم هو false.

بناء الجملة

```
if (condition) {
  // block of code to be executed if the condition is true
} else {
  // block of code to be executed if the condition is false
}
```

ابو حبيب الحسيني

مثال

```
int Hosini_time = 20;
if (Hosini_time < 18) {
  cout << "Good day.";
} else {
  cout << "Good evening.";
}
// Outputs "Good evening."
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

شاهد المثال

في المثال أعلاه، الوقت (20) أكبر من 18، وبالتالي فإن الشرط الحالة ونطبع على الشاشة "مساء else ولهذا السبب ننتقل إلى false. هو "الخير". إذا كان الوقت أقل من 18، فسيقوم البرنامج بطباعة "يوم جيد".

كيف وضع شروط بديلة

استخدم else if العبارة لتحديد شرط جديد إذا كان الشرط الأول هو false.

بناء الجملة

```
if (condition1) {
  // block of code to be executed if condition1 is true
} else if (condition2) {
```

```
  // block of code to be executed if the condition1 is
false and condition2 is true
} else {
  // block of code to be executed if the condition1 is
false and condition2 is false
}
```

مثال

```
int Hosini_time = 22;
if (Hosini_time < 10) {
  cout << "Good morning.";
} else if (Hosini_time < 20) {
  cout << "Good day.";
} else {
  cout << "Good evening.";
}
// Outputs "Good evening."
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى
ابو حبيب الحسيني

شاهد المثال

في المثال أعلاه، الوقت (22) أكبر من 10، لذا فإن الشرط الأول هو false. لذا ننتقل إلى الشرط التالي، في else if أيضًا، هو false- ونطبع على الشاشة "مساء الخير".

ومع ذلك، إذا كان الوقت 14، فسيطبع برنامجنا "يوم جيد".

كيف وضع الشروط بالطريقة المختصرة

هناك أيضًا اختصار if else. والذي يُعرف باسم العامل الثلاثي لأنه يتكون من ثلاثة معاملات. يمكن استخدامه لاستبدال عدة أسطر من الاكواد بسطر واحد. غالبًا ما يتم استخدامه لاستبدال عبارات if البسيطة:

بناء الجملة

```
variable =
(condition) ? expressionTrue : expressionFalse;
```

بدلاً من الكتابة:

مثال

```
int Hosini_time = 20;
if (Hosini_time < 18) {
  cout << "Good day.";
} else {
  cout << "Good evening.";
}
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

يمكنك ببساطة أن تكتب:

مثال

```
int Hosini_time = 20;
string result = (Hosini_time < 18) ? "Good day." : "Good
evening.";
cout << result;
```

وضع الشروط بدالة سويتش

طريقة سويتش

استخدم switch العبارة لتحديد إحدى كتل الاكواد العديدة التي سيتم تنفيذها.

بناء الجملة

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

هذه هي الطريقة التي يعمل بها:

ابو حبيب الحسيني

- يتم تقييم التعبير مرة واحدة switch.
- تتم مقارنة قيمة التعبير مع قيم كل منهما case.
- إذا كان هناك تطابق، فسيتم تنفيذ كتلة الاكواد المرتبطة.
- تعتبر الكلمات الأساسية break و default اختيارية، وسيتم وصفها لاحقًا في هذا الفصل.

يستخدم المثال أدناه رقم يوم الأسبوع لحساب اسم يوم الأسبوع:

مثال

```
int day = 4;
switch (day) {
  case 1:
    cout << "Monday";
    break;
  case 2:
    cout << "Tuesday";
    break;
  case 3:
    cout << "Wednesday";
    break;
  case 4:
    cout << "Thursday";
    break;
  case 5:
    cout << "Friday";
    break;
  case 6:
    cout << "Saturday";
```

ابو حبيب الحسيني

```
    break;
  case 7:
    cout << "Sunday";
    break;
}
// Outputs "Thursday" (day 4)
```

تعلم برمجة C++ بالعربى Abu Habib Al-Husin

كيف انهاء دالة سويتش الشرطية

كلمة أساسية، فإنه يخرج من كتلة break عندما يصل (السى بلس) إلى التبديل.

سيؤدي هذا إلى إيقاف تنفيذ المزيد من الاكواد واختبار الحالة داخل الكتلة.

عندما يتم العثور على تطابق، ويتم إنجاز المهمة، فقد حان وقت الاستراحة. ليست هناك حاجة لمزيد من الاختبارات.

يمكن أن يوفر الفاصل الكثير من وقت التنفيذ لأنه "يتجاهل" تنفيذ بقية الاكواد الموجودة في كتلة التبديل.

وضع اجراء افتراضى لدالة سويتش الشرطية

الأساسية بعض الاكواد ليتم تشغيلها في حالة عدم وجود default تحدد الكلمة تطابق لحالة الأحرف:

مثال

```
int day = 4;
switch (day) {
  case 6:
    cout << "Today is Saturday";
    break;
  case 7:
    cout << "Today is Sunday";
    break;
  default:
    cout << "Looking forward to the Weekend";
}
// Outputs "Looking forward to the Weekend"
```

حلقة ويل

الحلقات

يمكن للحلقات تنفيذ كتلة من الاكواد طالما تم الوصول إلى شرط محدد.

تعتبر الحلقات مفيدة لأنها توفر الوقت، وتقلل من الأخطاء، وتجعل الاكواد أكثر قابلية للقراءة.

ابو حبيب الحسيني

كيف استخدام حلقة ويل

while تتكرر الحلقة عبر كتلة من الاكواد طالما أن الشرط المحدد هو true:

بناء الجملة

```
while (condition) {
  // code block to be executed
}
```

في المثال أدناه، سيتم تشغيل الاكواد الموجودة في الحلقة مرارًا وتكرارًا، طالما أن المتغير (i) أقل من 5:

مثال

```
int i = 0;
while (i < 5) {
  cout << i << "\n";
  i++;
}
```

ابو حبيب الحسيني
Abu Habib Al-Husin تعلم برمجة C++ بالعربى

حلقة دو ويل

كيف استخدام حلقة دو يل

الحلقة. ستنفذ هذه الحلقة كتلة الاكواد while هي البديل من do/while الحلقة مرة واحدة، قبل التحقق مما إذا كان الشرط صحيحًا، ثم تكرر الحلقة طالما كان الشرط صحيحًا.

بناء الجملة

```
do {
  // code block to be executed
}
while (condition);
```

حلقة. سيتم دائمًا تنفيذ الحلقة مرة واحدة على do/while يستخدم المثال أدناه الأقل، حتى لو كان الشرط خاطئًا، لأنه يتم تنفيذ كتلة الاكواد قبل اختبار الشرط:

مثال

```
int i = 0;
do {
  cout << i << "\n";
  i++;
}
while (i < 5);
```

ابو حبيب الحسيني
Abu Habib Al-Husin تعلم برمجة C++ بالعربى

لا تنس زيادة المتغير المستخدم في الشرط، وإلا فلن تنتهي الحلقة أبدًا!

تابع الحلقات

حلقة فور

عندما تعرف بالضبط عدد المرات التي تريد فيها تكرار مجموعة من الاكواد، استخدم الحلقة for بدلاً من الحلقة while:

بناء الجملة

```
for (statement 1; statement 2; statement 3) {
  // code block to be executed
}
```

statement 1 (مرة واحدة) قبل تنفيذ كتلة الاكواد يتم تنفيذ الاجراء.

statement 2 يحدد الاجراء شرط تنفيذ كتلة الاكواد.

statement 3 (في كل مرة) بعد تنفيذ كتلة الاكواد يتم تنفيذ الاجراء.

المثال أدناه سيطبع الأرقام من 0 إلى 4:

مثال

```
for (int i = 0; i < 5; i++) {
  cout << i << "\n";
}
```

ابو حبيب الحسيني

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

شاهد المثال

(int i = 0). تقوم العبارة 1 بتعيين متغير قبل بدء الحلقة

يحدد الاجراء 2 شرط تشغيل الحلقة (يجب أن يكون أقل من 5). إذا كان الشرط صحيحًا، ستبدأ الحلقة من جديد، وإذا كان خاطئًا، فستنتهي الحلقة.

في كل مرة يتم فيها تنفيذ كتلة الاكواد في الحلقة (i++) يزيد الاجراء 3 القيمة.

مثال آخر

سيطبع هذا المثال القيم الزوجية فقط بين 0 و10:

مثال

```
for (int i = 0; i <= 10; i = i + 2) {
  cout << i << "\n";
}
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

حلقات متداخلة

من الممكن أيضًا وضع حلقة داخل حلقة أخرى. وهذا ما يسمى حلقة متداخلة.

سيتم تنفيذ "الحلقة الداخلية" مرة واحدة لكل تكرار للحلقة الخارجية:

ابو حبيب الحسيني

مثال

```
// Outer loop
for (int i = 1; i <= 2; ++i) {
  cout << "Outer: " << i << "\n"; // Executes 2
Hosini_times

  // Inner loop
  for (int j = 1; j <= 3; ++j) {
    cout << " Inner: " << j << "\n"; // Executes 6
Hosini_times (2 * 3)
  }
}
```

Abu Habib Al-Husini تعلم برمجة C++ بالعربي
ابو حبيب الحسيني

حلقة foreach

هناك أيضًا " حلقة لكل حلقة" (تم تقديمها في الإصدار 11 من لغة (السى بلص) (2011)، والتي تُستخدم حصريًا للحلقة عبر العناصر الموجودة في المصفوفة (أو مجموعات الاجراء ات الأخرى):

بناء الجملة

```
for (type variableName : arrayName) {
  // code block to be executed
}
```

يقوم المثال التالي بإخراج كافة العناصر في المصفوفة، باستخدام حلقة " for-each ":

مثال

```
int Hosini0[5] = {10, 20, 30, 40, 50};
for (int i : Hosini0) {
  cout << i << "\n";
}
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربي

استخدام كلمة بريك

القفز

العبارة المستخدمة في فصل سابق من هذا الكتاب break لقد رأيت بالفعل الاجراء switch . تم استخدامه "للقفز" من .

يمكن أيضًا استخدام العبارة break للانتقال من الحلقة .

يقفز هذا المثال خارج الحلقة عندما i يساوي 4:

مثال

```
for (int i = 0; i < 10; i++) {
  if (i == 4) {
    break;
  }
  cout << i << "\n";
}
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

(تابع حلقة فور

تكرارًا واحدًا (في الحلقة)، في حالة حدوث شرط continue تاستثناء العبارة محدد، وتستمر مع التكرار التالي في الحلقة.

يتخطى هذا المثال قيمة 4:

مثال

```
for (int i = 0; i < 10; i++) {
  if (i == 4) {
    continue;
  }
  cout << i << "\n";
}
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

استثناء واستمرار الحلقة

يمكنك أيضًا استخدام حلقات breakin continuewhile:

مثال استثناء

```
int i = 0;
while (i < 10) {
  cout << i << "\n";
  i++;
  if (i == 4) {
    break;
  }
}
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

متابعة المثال

```
int i = 0;
while (i < 10) {
  if (i == 4) {
    i++;
    continue;
  }
  cout << i << "\n";
  i++;
}
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

ابو حبيب الحسيني

المصفوفات

تُستخدم المصفوفات لتخزين قيم متعددة في متغير واحد، بدلاً من الإعلان عن متغيرات منفصلة لكل قيمة.

للإعلان عن مصفوفة، حدد نوع المتغير، ثم حدد اسم المصفوفة متبوعًا بأقواس مربعة وحدد عدد العناصر التي يجب تخزينها:

```
string Hosini9[4];
```

لقد أعلنا الآن عن متغير يحتوي على مصفوفة من أربع سلاسل. لإدراج قيم فيها، يمكننا استخدام مصفوفة حرفية - ضع القيم في قائمة مفصولة بفواصل، داخل الأقواس المتعرجة:

```
string Hosini9[4] = {"Habib", "Al_Husini
*_*", "Ali", "Mahmoud"};
```

لإنشاء مصفوفة مكونة من ثلاثة أعداد صحيحة، يمكنك كتابة:

```
int Habib3[3] = {10, 20, 30};
```

الوصول إلى عناصر المصفوفة

يمكنك الوصول إلى عنصر المصفوفة من خلال الإشارة إلى رقم الفهرس الموجود بين قوسين مربعين [].

ابو حبيب الحسيني

يصل هذا الاجراء إلى قيمة العنصر الأول في المستخدمين :

مثال

```
string Hosini9[4] = {"Habib", "Al_Husini
*_*", "Ali", "Mahmoud"};
cout << Hosini9[0];
// Outputs Habib
```

تعلم برمجة C++ بالعربي Abu Habib Al-Husin

ملاحظة: تبدأ فهارس المصفوفات بالرقم 0: [0] هو العنصر الأول. [1] هو العنصر الثاني، الخ.

تغيير عنصر مصفوفة

لتغيير قيمة عنصر معين،راجع رقم الفهرس:

```
Hosini9[0] = "Osman";
```

مثال

```
string Hosini9[4] = {"Habib", "Al_Husini
*_*", "Ali", "Mahmoud"};
Hosini9[0] = "Osman";
cout << Hosini9[0];
// Now outputs Osman instead of Habib
```

كيف تشغيل الحلقة على المصفوفات

استخدام الحلقة على المصفوفة

يمكنك تكرار عناصر المصفوفة باستخدام for الحلقة.

يقوم المثال التالي بإخراج كافة العناصر الموجودة في مصفوفة المستخدمين :

مثال

```
string Hosini9[5] = {"Habib", "Al_Husini
*_*", "Ali", "Mahmoud", "Tesla"};
for (int i = 0; i < 5; i++) {
  cout << Hosini9[i] << "\n";
}
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

يُخرج هذا المثال فهرس كل عنصر مع قيمته:

مثال

```
string Hosini9[5] = {"Habib", "Al_Husini
*_*", "Ali", "Mahmoud", "Tesla"};
for (int i = 0; i < 5; i++) {
  cout << i << " = " << Hosini9[i] << "\n";
}
```

ويوضح هذا المثال كيفية التكرار عبر مجموعة من الأعداد الصحيحة:

مثال

```
int Hosini0[5] = {10, 20, 30, 40, 50};
for (int i = 0; i < 5; i++) {
  cout << Hosini0[i] << "\n";
}
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

ابو حبيب الحسيني

حلقة foreach

هناك أيضًا " حلقة لكل حلقة" (تم تقديمها في الإصدار 11 (2011) من (السى بلص))، والتي تُستخدم حصريًا للحلقة عبر العناصر الموجودة في المصفوفة:

بناء الجملة

```
for (type variableName : arrayName) {
  // code block to be executed
}
```

يقوم المثال التالي بإخراج كافة العناصر في المصفوفة، باستخدام حلقة " for-each ":

مثال

```
int Hosini0[5] = {10, 20, 30, 40, 50};
for (int i : Hosini0) {
  cout << i << "\n";
}
```

حذف حجم المصفوفة

حذف حجم المصفوفة

في (السى بلص)، ليس عليك تحديد حجم المصفوفة. المترجم ذكي بما يكفي لتحديد حجم المصفوفة بناءً على عدد القيم المدرجة:

```
string Hosini9[] = {"Habib", "Al_Husini *_*", "Ali"}; //
Three arrays
```

المثال أعلاه يساوي:

```
string Hosini9[3] = {"Habib", "Al_Husini *_*", "Ali"}; //
Also three arrays
```

ومع ذلك، يعتبر النهج الأخير بمثابة "ممارسة جيدة"، لأنه سيقلل من فرصة حدوث أخطاء في برنامجك.

ابو حبيب الحسيني

حذف عناصر الإعلان

من الممكن أيضًا الإعلان عن مصفوفة دون تحديد العناصر في الإعلان، وإضافتها لاحقًا:

مثال

```
string Hosini9[5];
Hosini9[0] = "Habib";
Hosini9[1] = "Al_Husini *_*";
...
```

حجم المصفوفة

الحصول على حجم المصفوفة

للحصول على حجم المصفوفة، يمكنك استخدام العامل sizeof():

مثال

```
int Hosini0[5] = {10, 20, 30, 40, 50};
cout << sizeof(Hosini0);
```

نتيجة:

ابو حبيب الحسيني

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

لماذا تظهر النتيجة 20بدلا من 5عندما يحتوي المصفوفة على 5 عناصر؟

وذلك لأن sizeof() عامل التشغيل يُرجع حجم النوع بالبايت .

النوع عادة ما يكون 4 int لقد تعلمت من فصل أنواع الاجراء ات أن 4 بايت، لذلك من المثال أعلاه، 4 × 5 (5 عناصر × 4 بايت) = 20 بايت .

لمعرفة عدد العناصر الموجودة في المصفوفة ، عليك قسمة حجم المصفوفة على حجم نوع الاجراء ات الذي تحتوي عليه:

مثال

```
int Hosini0[5] = {10, 20, 30, 40, 50};
int getArrayLength = sizeof(Hosini0) / sizeof(int);
cout << getArrayLength;
```

نتيجة:

```
5
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

استخدام sizeof ()

في فصل المصفوفات والحلقات ، كتبنا حجم المصفوفة في حالة الحلقة (i < 5). هذا ليس مثاليًا، لأنه سيعمل فقط مع المصفوفات ذات الحجم المحدد.

النهج الموضح في المثال أعلاه، يمكننا الآن sizeof() ومع ذلك، باستخدام إنشاء حلقات تعمل مع المصفوفات من أي حجم، وهو أمر أكثر استدامة.

بدلاً من الكتابة:

```
int Hosini0[5] = {10, 20, 30, 40, 50};
for (int i = 0; i < 5; i++) {
  cout << Hosini0[i] << "\n";
}
```

من الأفضل أن تكتب:

مثال

```
int Hosini0[5] = {10, 20, 30, 40, 50};
for (int i = 0; i < sizeof(Hosini0) / sizeof(int); i++) {
  cout << Hosini0[i] << "\n";
}
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

لاحظ أنه في الإصدار 11 (2011) من (السى بلص)، يمكنك أيضًا استخدام حلقة "for-each" :

مثال

```
int Hosini0[5] = {10, 20, 30, 40, 50};
for (int i : Hosini0) {
  cout << i << "\n";
}
```

من الجيد معرفة الطرق المختلفة للتكرار عبر المصفوفة، حيث قد تواجهها جميعًا في برامج مختلفة.

المصفوفات متعددة الأبعاد في لغة

ماهي المصفوفات متعددة الأبعاد

المصفوفة متعددة الأبعاد هي مصفوفة من المصفوفات.

للإعلان عن مصفوفة متعددة الأبعاد، حدد نوع المتغير، وحدد اسم المصفوفة متبوعًا بأقواس مربعة تحدد عدد العناصر التي تحتوي عليها المصفوفة الرئيسية، متبوعة بمجموعة أخرى من الأقواس المربعة التي تشير إلى عدد العناصر الموجودة في المصفوفات الفرعية :

```
string Habib7[2][4];
```

كما هو الحال مع المصفوفات العادية، يمكنك إدراج قيم بمصفوفة حرفية - قائمة مفصولة بفواصل داخل الأقواس المتعرجة. في المصفوفة متعددة الأبعاد، كل عنصر في المصفوفة الحرفية هو مصفوفة حرفية أخرى.

```
string Habib7[2][4] = {
  { "A", "B", "C", "D" },
  { "E", "F", "G", "H" }
};
```

تضيف كل مجموعة من الأقواس المربعة في إعلان المصفوفة بُعدًا آخر للمصفوفة. يقال إن مصفوفة مثل تلك المذكورة أعلاه لها بعدان.

يمكن أن تحتوي المصفوفات على أي عدد من الأبعاد. كلما زادت أبعاد المصفوفة، أصبح الكود أكثر تعقيدًا. المصفوفة التالية لها ثلاثة أبعاد:

```
string Habib7[2][2][2] = {
  {
    { "A", "B" },
    { "C", "D" }
  },
  {
    { "E", "F" },
    { "G", "H" }
  }
};
```

كيف الوصول إلى عناصر مصفوفة متعددة الأبعاد

للوصول إلى عنصر في مصفوفة متعددة الأبعاد، حدد رقم فهرس في كل بعد من أبعاد المصفوفة.

يصل هذا الاجراء إلى قيمة العنصر في الصف الأول (0) والعمود الثالث (2) من مصفوفة الحروف.

ابو حبيب الحسيني

مثال

```
string Habib7[2][4] = {
  { "A", "B", "C", "D" },
  { "E", "F", "G", "H" }
};

cout << Habib7[0][2];  // Outputs "C"
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربي

تذكر أن: فهارس المصفوفات تبدأ بالرقم 0: [0] هو العنصر الأول. [1] هو العنصر الثاني، الخ.

كيف تغيير العناصر في مصفوفة متعددة الأبعاد

لتغيير قيمة عنصر ما، ارجع إلى الرقم القياسي للعنصر في كل بعد من الأبعاد:

مثال

```
string Habib7[2][4] = {
  { "A", "B", "C", "D" },
  { "E", "F", "G", "H" }
};
Habib7[0][0] = "Z";
```

```
cout << Habib7[0][0];  // Now outputs "Z" instead of "A"
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

كيف استخدام الحلقة على مصفوفة متعدد الأبعاد

للتكرار عبر مصفوفة متعددة الأبعاد، تحتاج إلى حلقة واحدة لكل بُعد من أبعاد المصفوفة.

يقوم المثال التالي بإخراج كافة العناصر الموجودة في مصفوفة الحروف :

مثال

```
string Habib7[2][4] = {
  { "A", "B", "C", "D" },
  { "E", "F", "G", "H" }
};

for (int i = 0; i < 2; i++) {
  for (int j = 0; j < 4; j++) {
    cout << Habib7[i][j] << "\n";
  }
}
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

يوضح هذا المثال كيفية التكرار عبر مصفوفة ثلاثية الأبعاد:

مثال

```
string Habib7[2][2][2] = {
  {
    { "A", "B" },
    { "C", "D" }
  },
  {
    { "E", "F" },
    { "G", "H" }
  }
};

for (int i = 0; i < 2; i++) {
  for (int j = 0; j < 2; j++) {
    for (int k = 0; k < 2; k++) {
      cout << Habib7[i][j][k] << "\n";
    }
  }
}
```

ابو حبيب الحسيني
Abu Habib Al-Husin تعلم برمجة C++ بالعربى

لماذا المصفوفات متعددة الأبعاد ؟

تعتبر المصفوفات متعددة الأبعادرائعة في تمثيل الشبكات. يوضح هذا المثال الاستخدام العملي لهم. في المثال التالي، نستخدم مصفوفة متعددة الأبعاد لتمثيل لعبة سفينة حربية صغيرة:

مثال

```
// We put "1" to indicate there is a ship.
bool ships[4][4] = {
  { 0, 1, 1, 0 },
  { 0, 0, 0, 0 },
  { 0, 0, 1, 0 },
  { 0, 0, 1, 0 }
};

// Keep track of how many hits the player has and how
many turns they have played in these variables
int hits = 0;
int numberOfTurns = 0;

// Allow the player to keep going until they have hit all
four ships
while (hits < 4) {
  int row, column;

  cout << "Selecting coordinates\n";

  // Ask the player for a row
  cout << "Choose a row number between 0 and 3: ";
  cin >> row;

  // Ask the player for a column
  cout << "Choose a column number between 0 and 3: ";
  cin >> column;
```

ابو حبيب الحسيني

```
  // Check if a ship exists in those coordinates
  if (ships[row][column]) {
    // If the player hit a ship, remove it by setting the
value to zero.
    ships[row][column] = 0;

    // Increase the hit counter
    hits++;

    // Tell the player that they have hit a ship and how
many ships are left
    cout << "Hit! " << (4-hits) << " left.\n\n";
  } else {
    // Tell the player that they missed
    cout << "Miss\n\n";
  }

  // Count how many turns the player has taken
  numberOfTurns++;
}

cout << "Hosini11!\n";
cout << "You won in " << numberOfTurns << " turns";
```

(الهيكلة)

ابو حبيب الحسيني

ما هى الهياكل

الهياكل هي طريقة لتجميع العديد من المتغيرات ذات الصلة في مكان واحد. يُعرف كل متغير في البنية كعضو في البنية

على عكس **المصفوفة** ، يمكن أن تحتوي البنية على العديد من أنواع الاجراء ات المختلفة (int، string، bool، وما إلى ذلك).

إنشاء هيكل

الكلمة الأساسية struct لإنشاء بنية، استخدم وأعلن عن كل عضو من أعضائها داخل الأقواس المتعرجة.

في المثال أدناه (HabibStructure) بعد التصريح، حدد اسم متغير البنية:

```
struct {          // Structure declaration
  int Habib3;       // Member (int variable)
  string Hosini5;  // Member (string variable)
} HabibStructure;     // Structure variable
```

أعضاء هيكل الوصول

للوصول إلى أعضاء البنية، استخدم بناء الجملة النقطي (.):

مثال

تعيين الاجراء ات لأعضاء الهيكل وطباعتها:

ابو حبيب الحسيني

```
// Create a structure variable called HabibStructure
struct {
  int Habib3;
  string Hosini5;
} HabibStructure;

// Assign values to members of HabibStructure
HabibStructure.Habib3 = 1;
HabibStructure.Hosini5 = "Abu Habib Al_Husini";

// Print members of HabibStructure
cout << HabibStructure.Habib3 << "\n";
cout << HabibStructure.Hosini5 << "\n";
```

Abu Habib Al-Husini تعلم برمجة C++ بالعربي

هيكل واحد في متغيرات متعددة

يمكنك استخدام الفاصلة (,) لاستخدام بنية واحدة في العديد من المتغيرات:

```
struct {
  int Habib3;
  string Hosini5;
} Hosini_Struct1, Hosini_Struct2, Hosini_Struct3; // Multiple structure variables separated with commas
```

يوضح هذا المثال كيفية استخدام بنية في متغيرين مختلفين:

مثال

استخدم بنية واحدة لتمثيل سيارتين:

```
struct {
  string Name;
  string Hosini2;
  int year;
} Alhusini1, Alhusini2; // We can add variables by
separating them with a comma here

// Put data into the first structure
Alhusini1.Name = "Al_Husini *_*";
Alhusini1.Hosini2 = "AlMasre";
Alhusini1.year = 1999;

// Put data into the second structure
Alhusini2.Name = "Ali";
Alhusini2.Hosini2 = "Mustafa";
Alhusini2.year = 1969;

// Print the structure members
cout << Alhusini1.Name << " " << Alhusini1.Hosini2 << "
" << Alhusini1.year << "\n";
cout << Alhusini2.Name << " " << Alhusini2.Hosini2 << "
" << Alhusini2.year << "\n";
```

ابو حبيب الحسيني

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

تابع الهياكل

ومن خلال إعطاء اسم للبنية، يمكنك التعامل معها كنوع اجراء ات. هذا يعني أنه يمكنك إنشاء متغيرات بهذه البنية في أي مكان في البرنامج وفي أي وقت.

الأساسيةstruct لإنشاء بنية مسماة، ضع اسم البنية مباشرة بعد الكلمة:

```
struct HabibDataType { // This structure is named
"HabibDataType"
  int Habib3;
  string Hosini5;
};
```

للإعلان عن متغير يستخدم البنية، استخدم اسم البنية كنوع اجراء ات المتغير:

مثال

استخدم بنية واحدة لتمثيل سيارتين:

```
// Declare a structure named "Hosini "
struct Hosini  {
  string Name;
  string Hosini2;
  int year;
};
```

ابو حبيب الحسيني

```
int main() {
  // Create a Hosini  structure and store it in Alhusini1;
  Hosini  Alhusini1;
  Alhusini1.Name = "Al_Husini *_*";
  Alhusini1.Hosini2 = "AlMasre";
  Alhusini1.year = 1999;

  // Create another Hosini  structure and store it in
Alhusini2;
  Hosini  Alhusini2;
  Alhusini2.Name = "Ali";
  Alhusini2.Hosini2 = "Mustafa";
  Alhusini2.year = 1969;

  // Print the structure members
  cout << Alhusini1.Name << " " << Alhusini1.Hosini2 << "
" << Alhusini1.year << "\n";
  cout << Alhusini2.Name << " " << Alhusini2.Hosini2 << "
" << Alhusini2.year << "\n";

  return 0;
}
```

ابو حبيب الحسيني

المرجع

كيف إنشاء المراجع

المتغير المرجعي هو "مرجع" لمتغير موجود، ويتم إنشاؤه باستخدام &عامل التشغيل:

```
string hAbIB = "Al Husini";  // hAbIB variable
string &meal = hAbIB;    // reference to hAbIB
```

للإشارة meal أو الاسم المرجعي hAbIB الآن، يمكننا استخدام اسم المتغير المتغير hAbIB إلى:

مثال

```
string hAbIB = "Al Husini";
string &meal = hAbIB;

cout << hAbIB << "\n"; // Outputs Al Husini
cout << meal << "\n"; // Outputs Al Husini
```

عنوان الذاكرة

كيف استخدام عنوان الذاكرة

في المثال من الصفحة السابقة، &تم استخدام عامل التشغيل لإنشاء متغير مرجعي. ولكن يمكن استخدامه أيضًا للحصول على عنوان الذاكرة للمتغير؛ وهو الموقع حيث يتم تخزين المتغير على الكمبيوتر.

ابو حبيب الحسيني

عند إنشاء متغير في لغة (السى بلص)، يتم تعيين عنوان ذاكرة للمتغير.وعندما نخصص قيمة للمتغير، يتم تخزينها في عنوان الذاكرة هذا.

للوصول إليه، استخدم & العامل، وستمثل النتيجة مكان تخزين المتغير:

مثال

```
string hAbIB = "Al Husini";

cout << &hAbIB; // Outputs 0x6dfed4
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

لاحظ أنك قد لا (0x..). ملحوظة: عنوان الذاكرة مكتوب بالنظام الست عشري تحصل على نفس النتيجة في برنامجك.

ولماذا من المفيد معرفة عنوان الذاكرة؟

تعد المراجع والمؤشرات (التي ستتعرف عليها في الفصل التالي) مهمة في لغة (السى بلص)، لأنها تمنحك القدرة على معالجة الاجراء ات الموجودة في ذاكرة الكمبيوتر - مما قد يؤدي إلى تقليل الاكواد وتحسين الأداء .

هاتان الميزتان هما أحد الأشياء التي تجعل (السى بلص) تتميز عن لغات البرمجة الأخرى، مثل Java و Python .

مؤشرات

إنشاء المؤشرات

لقد تعلمت من الفصل السابق أنه يمكننا الحصول على عنوان الذاكرة للمتغير باستخدام & المعامل:

مثال

```
string hAbIB = "Al Husini"; // A hAbIB variable of type string

cout << hAbIB;  // Outputs the value of hAbIB (Al Husini)
cout << &hAbIB; // Outputs the memory address of hAbIB (0x6dfed4)
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

لكن المؤشر هو متغير يقوم بتخزين عنوان الذاكرة كقيمة له.

من نفس النوع، (مثل string أو int) يشير متغير المؤشر إلى نوع اجراء ات ويتم إنشاؤه باستخدام * عامل التشغيل. يتم تعيين عنوان المتغير الذي تعمل معه للمؤشر:

مثال

```
string hAbIB = "Al Husini";  // A hAbIB variable of type string
string* ptr = &hAbIB;    // A pointer variable, with the name ptr, that stores the address of hAbIB
```

```
// Output the value of hAbIB (Al Husini)
cout << hAbIB << "\n";

// Output the memory address of hAbIB (0x6dfed4)
cout << &hAbIB << "\n";

// Output the memory address of hAbIB with the pointer
(0x6dfed4)
cout << ptr << "\n";
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربي

شاهد المثال

باستخدام علامة string، يشير إلى متغير ptr، قم بإنشاء متغير مؤشر بالاسم string* ptr). لاحظ أن نوع المؤشر يجب أن يتطابق مع نوع النجمة *(المتغير الذي تعمل به.

hAbIB استخدم &عامل التشغيل لتخزين عنوان الذاكرة للمتغير المسمى وتعيينه للمؤشر.

الآن، ptr يحمل قيمة hAbIB عنوان الذاكرة .

نصيحة: هناك ثلاث طرق للإعلان عن متغيرات المؤشر، لكن الطريقة الأولى هي المفضلة:

```
string* Hosini5; // Preferred
string *Hosini5;
string * Hosini5;
```

استخدام المؤشر للحصول على القيمة

الحصول على عنوان الذاكرة والقيمة

في المثال من الصفحة السابقة، استخدمنا متغير المؤشر للحصول على عنوان الذاكرة للمتغير (يستخدم مع العامل & المرجعي). ومع ذلك، يمكنك أيضًا استخدام المؤشر للحصول على قيمة المتغير، باستخدام العامل *(عامل عدم المرجعية):

مثال

```
string hAbIB = "Al Husini";  // Variable declaration
string* ptr = &hAbIB;    // Pointer declaration

// Reference: Output the memory address of hAbIB with
the pointer (0x6dfed4)
cout << ptr << "\n";

// Dereference: Output the value of hAbIB with the
pointer (Al Husini)
cout << *ptr << "\n";
```

تعديل المؤشرات

أبو حبيب الحسيني

كيف تعديل قيمة المؤشر

يمكنك أيضًا تغيير قيمة المؤشر. لكن لاحظ أن هذا سيؤدي أيضًا إلى تغيير قيمة المتغير الأصلي:

مثال

```
string hAbIB = "Al Husini";
string* ptr = &hAbIB;

// Output the value of hAbIB (Al Husini)
cout << hAbIB << "\n";

// Output the memory address of hAbIB (0x6dfed4)
cout << &hAbIB << "\n";

// Access the memory address of hAbIB and output its
value (Al Husini)
cout << *ptr << "\n";

// Change the value of the pointer
*ptr = "Hamburger";

// Output the new value of the pointer (Hamburger)
cout << *ptr << "\n";

// Output the new value of the hAbIB variable
(Hamburger)
cout << hAbIB << "\n";
```

ابو حبيب الحسيني

وظائف

الوظيفة عبارة عن كتلة من الاكواد التي تعمل فقط عند استدعائها.

يمكنك تمرير الاجراء ات، المعروفة باسم المعلمات، إلى دالة.

تُستخدم الوظائف لتنفيذ إجراءات معينة، وهي مهمة لإعادة استخدام الكود: حدد الكود مرة واحدة، واستخدمه عدة مرات.

كيف إنشاء وظيفة

والتي يتم ،main() يوفر (السى بلص) بعض الوظائف المحددة مسبقًا، مثل استخدامها لتنفيذ الاكواد. ولكن يمكنك أيضًا إنشاء وظائفك الخاصة لتنفيذ إجراءات معينة.

لإنشاء دالة (يُشار إليها غالبًا باسم "التصريح")، حدد اسم الوظيفة، متبوعًا بالأقواس () :

بناء الجملة

```
void HabibFunction() {
  // code to be executed
}
```

شاهد المثال

هو اسم الدالة HabibFunction().

ابو حبيب الحسيني

يعني أن الدالة ليس لها قيمة إرجاع. سوف تتعلم المزيد عن قيم void. الإرجاع لاحقًا في الفصل التالي

- داخل الدالة (النص)، قم بإضافة الاكواد التي تحدد ما يجب أن تفعله الدالة

استدعاء وظيفة

لا يتم تنفيذ الوظائف المعلنة على الفور. ويتم "حفظها لاستخدامها لاحقًا"، وسيتم تنفيذها لاحقًا عند استدعائها.

لاستدعاء دالة، اكتب اسم الدالة متبوعا بقوسين () وفاصلة منقوطة;

في المثال التالي، HabibFunction() يتم استخدامه لطباعة النص (الإجراء)، عندما يتم استدعاؤه:

مثال

في الداخل main اتصل HabibFunction():

```
// Create a function
void HabibFunction() {
  cout << "Abu Habib Al husini";
}

int main() {
  HabibFunction(); // call the function
  return 0;
}
```

ابو حبيب الحسيني

```
// Outputs "Abu Habib Al husini"
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

يمكن استدعاء الدالة عدة مرات:

مثال

```
void HabibFunction() {
  cout << "Abu Habib Al husini\n";
}

int main() {
  HabibFunction();
  HabibFunction();
  HabibFunction();
  return 0;
}

// Abu Habib Al husini
// Abu Habib Al husini
// Abu Habib Al husini
```

ابو حبيب الحسيني

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

إلإعلان عن الوظيفة وتعريفها

تتكون دالة (السى بلص) من جزأين:

الإعلان: نوع الإرجاع واسم الوظيفة والمعلمات (إن وجدت).
التعريف: نص الوظيفة (الكود المراد تنفيذه).

```
void HabibFunction() { // declaration
  // the body of the function (definition)
}
```

ملحوظة: إذا تم الإعلان عن وظيفة محددة من قبل المستخدم، مثل التي تم الإعلان عنها بعد HabibFunction() تلك الوظيفة main()، فسيحدث خطأ :

مثال

```
int main() {
  HabibFunction();
  return 0;
}

void HabibFunction() {
  cout << "Abu Habib Al husini";
}

// Error
```

ابو حبيب الحسيني

تعلم برمجة C++ بالعربى Abu Habib Al-Husin

ومع ذلك، من الممكن فصل الإعلان عن تعريف الوظيفة - لتحسين الكود

ستشاهد غالبًا برامج (السى بلص) التي تحتوي على إعلان الوظيفة سيؤدي هذا إلى تحسين تنظيم main(). وتعريف الوظيفة أدناه main() أعلاه الاكواد وأسهل في القراءة:

مثال

```
// Function declaration
void HabibFunction();

// The main method
int main() {
  HabibFunction();  // call the function
  return 0;
}

// Function definition
void HabibFunction() {
  cout << "Abu Habib Al husini";
}
```

ابو حبيب الحسيني

معلمات الوظيفة

المعلمات والحجج

يمكن تمرير المعلومات إلى الوظائف كمعلمة. تعمل المعلمات كمتغيرات داخل الوظيفة.

يتم تحديد المعلمات بعد اسم الوظيفة، داخل الأقواس. يمكنك إضافة أي عدد تريده من المعلمات، ما عليك سوى الفصل بينها بفاصلة:

بناء الجملة

```
void functionName(parameter1, parameter2, parameter
3) {
  // code to be executed
}
```

كمعلمة. عندما يتم fname اسم string يحتوي المثال التالي على دالة تأخذ استدعاء الدالة، نقوم بتمرير الاسم الأول، والذي يتم استخدامه داخل الدالة لطباعة الاسم الكامل:

مثال

```
void HabibFunction(string fname) {
  cout << fname << " Al Hosini \n";
}

int main() {
  HabibFunction("Habib9");
  HabibFunction("Habib8");
  HabibFunction("Habib5");
  return 0;
}

// Habib9 Al Hosini
// Habib8 Al Hosini
// Habib5 Al Hosini
```

أبو حبيب الحسيني

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

عندما يتم تمرير معلمة إلى الدالة، يطلق عليها اسم الوسيطة . لذا، من المثال أعلاه: fname هي معلمة ، بينما Habib9، Habib8 و Habib5 هي وسيطا ت .

المعلمات الافتراضية لـ

قيمة المعلمة الافتراضية

يمكنك أيضًا استخدام قيمة معلمة افتراضية، باستخدام علامة التساوي (=).

إذا قمنا باستدعاء الدالة بدون وسيطة، فإنها تستخدم القيمة الافتراضية ("النرويج"):

مثال

```
void HabibFunction(string country = "Ahmed") {
  cout << country << "\n";
}

int main() {
  HabibFunction("Hamza");
  HabibFunction("Ali");
  HabibFunction();
  HabibFunction("Mahmoud");
  return 0;
}
```

ابو حبيب الحسيني

```
// Hamza
// Ali
// Ahmed
// Mahmoud
```


تُعرف المعلمة ذات القيمة الافتراضية غالبًا باسم " المعلمة الاختيارية ". من المثال أعلاه، country هي معلمة اختيارية وهي القيمة "Ahmed" الافتراضية.

معلمات متعددة لـ فى الدالة

معلمات متعددة

داخل الوظيفة، يمكنك إضافة أي عدد تريده من المعلمات:

مثال

```
void HabibFunction(string fname, int age) {
  cout << fname << " Al Hosini . " << age << " years old. \
n";
}

int main() {
  HabibFunction("Habib9", 3);
```

```
  HabibFunction("Habib8", 14);
  HabibFunction("Habib5", 30);
  return 0;
}

// Habib9 Al Hosini . 3 years old.
// Habib8 Al Hosini . 14 years old.
// Habib5 Al Hosini . 30 years old.
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

لاحظ أنه عند العمل مع معلمات متعددة، يجب أن يحتوي استدعاء الدالة على نفس عدد الوسائط كما توجد معلمات، ويجب تمرير الوسائط بنفس الترتيب.

الكلمة الأساسية المرتجعة

إرجاع القيم

الأساسية المستخدمة في الأمثلة السابقة إلى أن الدالة لا void تشير الكلمة يجب أن تُرجع قيمة. إذا كنت تريد أن تقوم الدالة بإرجاع قيمة، فيمكنك void، بدلاً من (وما إلى ذلك int، string مثل) استخدام نوع اجراء ات الكلمة الأساسية داخل الدالة return واستخدام:

مثال

```
int HabibFunction(int x) {
  return 5 + x;
}

int main() {
  cout << HabibFunction(3);
  return 0;
}

// Outputs 8 (5 + 3)
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى
ابو حبيب الحسينى

يُرجع هذا المثال مجموع دالة بمعلمتين :

مثال

```
int HabibFunction(int x, int y) {
  return x + y;
}

int main() {
  cout << HabibFunction(5, 3);
  return 0;
}

// Outputs 8 (5 + 3)
```

يمكنك أيضًا تخزين النتيجة في متغير:

مثال

```
int HabibFunction(int x, int y) {
  return x + y;
}

int main() {
  int z = HabibFunction(5, 3);
  cout << z;
  return 0;
}
// Outputs 8 (5 + 3)
```

ابو حبيب الحسيني

الوظائف - التمرير حسب المرجع

كيف يتم التمرير حسب المرجع

في الأمثلة من الصفحة السابقة، استخدمنا المتغيرات العادية عندما مررنا معلمات إلى دالة. يمكنك أيضًا تمرير مرجع إلى الوظيفة. يمكن أن يكون هذا مفيدًا عندما تحتاج إلى تغيير قيمة الوسائط:

مثال

```
void Habib0(int &x, int &y) {
  int z = x;
  x = y;
  y = z;
}

int main() {
  int Habib6 = 10;
  int Habib9 = 20;

  cout << "Before swap: " << "\n";
  cout << Habib6 << Habib9 << "\n";

  // Call the function, which will change the values of
Habib6 and Habib9
  Habib0(Habib6, Habib9);

  cout << "After swap: " << "\n";
  cout << Habib6 << Habib9 << "\n";

  return 0;
}
```

ابو حبيب الحسيني

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

كيف تمرير المصفوفات كبرامترات للدالة

يمكنك أيضًا تمرير **المصفوفات** إلى دالة:

مثال

```
void HabibFunction(int Hosini0[5]) {
  for (int i = 0; i < 5; i++) {
    cout << Hosini0[i] << "\n";
  }
}

int main() {
  int Hosini0[5] = {10, 20, 30, 40, 50};
  HabibFunction(Hosini0);
  return 0;
}
```

أبو حبيب الحسيني

تعلم برمجة C++ بالعربي Abu Habib Al-Husin

شاهد المثال

تأخذ الدالة (HabibFunction) مصفوفة كمعلمة لها (int Hosini0[5]).
وتقوم بالتكرار خلال عناصر المصفوفة باستخدام الحلقة for.

المصفوفة Hosini0 نقوم بتمرير ، عندما يتم استدعاء الدالة بالداخل main()
التي تقوم بإخراج عناصر المصفوفة.

لاحظ أنه عند استدعاء الدالة، ما عليك سوى استخدام اسم المصفوفة عند تمريرها كوسيطة HabibFunction(Hosini0). ومع ذلك، يلزم الإعلان الكامل عن المصفوفة في معلمة الدالة (int Hosini0[5]).

التحميل الزائد للوظيفة

كيف يتم التحميل الزائد

مع التحميل الزائد للوظيفة ، يمكن أن يكون لوظائف متعددة نفس الاسم مع معلمات مختلفة:

مثال

```
int HabibFunction(int x)
float HabibFunction(float x)
double HabibFunction(double x, double y)
```

أبو حبيب الحسيني

خذ بعين الاعتبار المثال التالي، الذي يحتوي على دالتين تضيفان أرقامًا من نوع مختلف:

مثال

```
int plusFuncInt(int x, int y) {
  return x + y;
}

double plusFuncDouble(double x, double y) {
  return x + y;
}

int main() {
  int Habib31 = plusFuncInt(8, 5);
  double Habib32 = plusFuncDouble(4.3, 6.26);
  cout << "Int: " << Habib31 << "\n";
  cout << "Double: " << Habib32;
  return 0;
}
```

أبو حبيب الحسيني
Abu Habib Al-Husin تعلم برمجة C++ بالعربي

بدلاً من تحديد وظيفتين يجب أن تقوما بنفس الشيء، فمن الأفضل تحميل إحداهما بشكل زائد.

لتعمل لكل **plusFunc** في المثال أدناه، قمنا بتحميل الدالة بشكل زائد من **int** و **double**:

مثال

```
int plusFunc(int x, int y) {
  return x + y;
}

double plusFunc(double x, double y) {
  return x + y;
}

int main() {
  int Habib31 = plusFunc(8, 5);
  double Habib32 = plusFunc(4.3, 6.26);
  cout << "Int: " << Habib31 << "\n";
  cout << "Double: " << Habib32;
  return 0;
}
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى
ابو حبيب الحسيني

ملاحظة: يمكن أن يكون للوظائف المتعددة نفس الاسم طالما أن عدد و/أو نوع المعلمات مختلف.

الإرجاع من داخل الوظيفة

الاسترجاع هي تقنية إجراء استدعاء دالة نفسها. توفر هذه التقنية طريقة لتقسيم المشكلات المعقدة إلى مشكلات بسيطة يسهل حلها.

قد يكون من الصعب بعض الشيء أن نفهم الاسترجاع. أفضل طريقة لمعرفة كيفية عملها هي تجربتها.

مثال على الاسترجاع

من السهل القيام بجمع رقمين معًا، لكن إضافة نطاق من الأرقام أكثر تعقيدًا. في المثال التالي، يتم استخدام الاسترجاع لإضافة نطاق من الأرقام معًا عن طريق تقسيمها إلى مهمة بسيطة تتمثل في إضافة رقمين:

مثال

```
int sum(int k) {
  if (k > 0) {
    return k + sum(k - 1);
  } else {
    return 0;
  }
}

int main() {
  int result = sum(10);
  cout << result;
  return 0;
}
```

ابو حبيب الحسيني

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

شاهد المثال

إلى مجموع كل k يتم استدعاء الدالة، فإنها تضيف معلمة sum() عندما تقوم الدالة بإرجاع 0 ، 0 k وترجع النتيجة. عندما يصبح k الأرقام الأصغر منها فقط. عند التشغيل، يتبع البرنامج الخطوات التالية:

تكون 0، فإن البرنامج يتوقف عند هذا k وبما أن الدالة لا تستدعي نفسها عندما الحد ويعيد النتيجة.

يجب أن يكون المطور حذرًا للغاية فيما يتعلق بالتكرار لأنه قد يكون من السهل جدًا الانزلاق إلى كتابة وظيفة لا تنتهي أبدًا، أو وظيفة تستخدم كميات زائدة من الذاكرة أو طاقة المعالج. ومع ذلك، عند كتابتها بشكل صحيح، يمكن أن تكون الاسترجاع طريقة فعالة للغاية وأنيقة رياضيًا للبرمجة.

OOP

ما هو OOP؟

OOP تعني البرمجة الشيئية.

البرمجة الإجرائية تدور حول كتابة الإجراءات أو الوظائف التي تنفذ العمليات على الاجراء ات، في حين أن البرمجة الموجهة للكائنات تدور حول إنشاء كائنات تحتوي على كل من الاجراء ات والوظائف.

تتميز البرمجة الموجهة للكائنات بالعديد من المزايا مقارنة بالبرمجة الإجرائية:

- OOP أسرع وأسهل في التنفيذ
- OOP يوفر بنية واضحة للبرامج

ابو حبيب الحسيني

- في الحفاظ على كود (السى بلص) جافًا "لا تكرر نفسك"، OOP يساعد ويجعل صيانة الكود وتعديله وتصحيحه أسهل
- إنشاء تطبيقات كاملة قابلة لإعادة الاستخدام برمز أقل ووقت OOP يتيح تطوير أقصر

يتعلق بتقليل تكرار الاكواد. يجب عليك (DRY) "نصيحة: مبدأ "لا تكرر نفسك استخراج الرموز المشتركة للتطبيق، ووضعها في مكان واحد وإعادة استخدامها بدلاً من تكرارها.

ما هي الفئات والكائنات؟

الفئات والكائنات هما الجانبان الرئيسيان للبرمجة الموجهة للكائنات.

انظر إلى الرسم التوضيحي التالي لمعرفة الفرق بين الفئة والكائنات:

عندما يتم إنشاء الكائنات الفردية، فإنها ترث كافة المتغيرات والوظائف من الفئة.

سوف تتعلم المزيد عن **الفئات والكائنات** في الفصل التالي.

فئات وكائنات

كيف التعامل مع الفئات/كائنات

هي لغة برمجة موجهة للكائنات (السى بلص).

يرتبط كل شيء في (السى بلص) بالفئات والكائنات، بالإضافة إلى سماتها وأساليبها. على سبيل المثال: في الواقعية، السيارة هي شيء . تتمتع السيارة بخصائص مثل الوزن واللون وطرق مثل القيادة والفرامل.

السمات والأساليب هي في الأساس متغيرات ووظائف تنتمي إلى الفصل. ويُشار إليهم غالبًا باسم "أعضاء الفصل".

الفئة هي نوع اجراء ات محدد من قبل المستخدم يمكننا استخدامه في برنامجنا، ويعمل كمنشئ كائن، أو "مخطط" لإنشاء الكائنات.

إنشاء كلاس

الكلمة الأساسية **class** لإنشاء كلاس، استخدم:

مثال

إنشاء فئة تسمى " **Class**":

```
class Class {     // The class
  public:         // Access specifier
    int Habib3;       // Attribute (int variable)
    string Hosini5;  // Attribute (string variable)
};
```

شاهد المثال

- **class** الأساسية لإنشاء فئة تسمى **Class**.

- الأساسية هي محدد الوصول ، الذي يحدد أنه يمكن **public** الكلمة الوصول إلى أعضاء (السمات والأساليب) للفئة من خارج الفئة. سوف تتعلم المزيد حول **محددات الوصول** لاحقًا.

ابو حبيب الحسيني

- ومتغير Habib3 يوجد داخل الفصل متغير عدد صحيح
عندما يتم الإعلان عن المتغيرات داخل فئة ما، فإنها Hosini5. نص
تسمى السمات .

- أخيرًا، قم بإنهاء تعريف الفئة بفاصلة منقوطة ;.

كيف إنشاء الكائنات

في (السى بلص)، يتم إنشاء كائن من فئة. لقد قمنا بالفعل بإنشاء فئة Class, تسمى
لذا يمكننا الآن استخدامها لإنشاء كائنات.

Class لإنشاء كائن، متبوعًا باسم الكائن.
حدد اسم الفئة.

استخدم بناء الجملة (Hosini5 وHabib3) للوصول إلى سمات الفئة
النقطي (.) على الكائن:

مثال

أنشئ كائنًا يسمى "Hosini_Obj" وقم بالوصول إلى السمات:

```
class Class {     // The class
  public:         // Access specifier
    int Habib3;       // Attribute (int variable)
    string Hosini5;  // Attribute (string variable)
};

int main() {
  Class Hosini_Obj;  // Create an object of Class
```

```
  // Access attributes and set values
  Hosini_Obj.Habib3 = 15;
  Hosini_Obj.Hosini5 = "Some text";

  // Print attribute values
  cout << Hosini_Obj.Habib3 << "\n";
  cout << Hosini_Obj.Hosini5;
  return 0;
}
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

كائنات متعددة

يمكنك إنشاء كائنات متعددة من فئة واحدة:

مثال

```
// Create a Hosini  class with some attributes
class Hosini  {
  public:
    string Name;
    string Hosini2;
    int year;
};

int main() {
```

```
  // Create an object of Hosini
  Hosini  HabibObJ1;
  HabibObJ1.Name = "Al_Husini *_*";
  HabibObJ1.Hosini2 = "AlMasre";
  HabibObJ1.year = 1999;

  // Create another object of Hosini
  Hosini  HabibObJ2;
  HabibObJ2.Name = "Ali";
  HabibObJ2.Hosini2 = "Mustafa";
  HabibObJ2.year = 1969;

  // Print attribute values
  cout << HabibObJ1.Name << " " <<
HabibObJ1.Hosini2 << " " << HabibObJ1.year << "\n";
  cout << HabibObJ2.Name << " " <<
HabibObJ2.Hosini2 << " " << HabibObJ2.year << "\n";
  return 0;
}
```

ابو حبيب الحسيني

طرق الفئات

كيف استخدام الأساليب او الطرق من داخل الكلاس

الأساليب هي وظائف تنتمي إلى الفئة.

هناك طريقتان لتحديد الوظائف التي تنتمي إلى فئة:

- تعريف الطبقة الداخلية
- تعريف الطبقة الخارجية

في المثال التالي، قمنا بتعريف دالة داخل الفصل، وقمنا بتسميتها "HabibMethod".

ملاحظة: يمكنك الوصول إلى الأساليب تمامًا مثلما تصل إلى السمات؛ عن طريق إنشاء كائن من الفئة واستخدام بناء الجملة النقطي (.):

داخل المثال

```
class Class {       // The class
  public:           // Access specifier
    void HabibMethod() {  // Method/function defined ins
      cout << "Abu Habib Al_Husini";
    }
};

int main() {
  Class Hosini_Obj;    // Create an object of Class
  Hosini_Obj.HabibMethod();  // Call the method
  return 0;
}
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

لتعريف دالة خارج تعريف الفئة، عليك أن تعلن عنها داخل الفئة ثم تحددها خارج الفئة. يتم ذلك عن طريق تحديد اسم الفئة، متبوعًا بـ:: عامل تحليل النطاق، متبوعًا باسم الوظيفة:

مثال خارجي

```
class Class {       // The class
  public:           // Access specifier
    void HabibMethod();  // Method/function declaration
};

// Method/function definition outside the class
void Class::HabibMethod() {
  cout << "Abu Habib Al_Husini";
}

int main() {
  Class Hosini_Obj;    // Create an object of Class
  Hosini_Obj.HabibMethod(); // Call the method
  return 0;
}
```

ابو حبيب الحسيني
Abu Habib Al-Husin تعلم برمجة C++ بالعربى

كيف اضافة معلمات او براميترات

يمكنك أيضًا إضافة معلمات:

مثال

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

```
class Hosini  {
  public:
    int speed(int maxSpeed);
};

int Hosini ::speed(int maxSpeed) {
  return maxSpeed;
}

int main() {
  Hosini  Hosini_Obj; // Create an object of Hosini
  cout << Hosini_Obj.speed(200); // Call the method with
an argument
  return 0;
}
```

المنشئو

ما هو المنشا فى الكلاس

المُنشئ في لغة (السى بلص) هو أسلوب خاص يتم استدعاؤه تلقائيًا عند إنشاء كائن من فئة ما.

لإنشاء مُنشئ، استخدم نفس اسم الفئة، متبوعًا بالأقواس ():

ابو حبيب الحسينى

مثال

```
class Class {    // The class
  public:        // Access specifier
    Class() {    // Constructor
      cout << "Abu Habib Al_Husini";
    }
};

int main() {
  Class Hosini_Obj;   // Create an object of Class (this
will call the constructor)
  return 0;
}
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

ابو حبيب الحسيني

ملاحظة: المُنشئ له نفس اسم الفئة، وهو دائمًا public وليس لديه أي قيمة مُرجعة.

معلمات المنشئ داخل الكلاس

يمكن للمنشئين أيضًا أخذ المعلمات (تمامًا مثل الوظائف العادية)، والتي يمكن أن تكون مفيدة لتعيين القيم الأولية للسمات.

تحتوي الفئة التالية على سمات Name ومنشئ Hosini2 بمعلمات year مختلفة. داخل المُنشئ، قمنا بتعيين السمات مساوية لمعلمات المُنشئ عندما نستدعي المنشئ (عن طريق إنشاء كائن من الفئة)، (إلخ. Name=x)

فإننا نمرر المعلمات إلى المنشئ، والذي سيضبط قيمة السمات المقابلة على النحو نفسه:

مثال

```
class Hosini  {       // The class
  public:        // Access specifier
    string Name;  // Attribute
    string Hosini2;  // Attribute
    int year;     // Attribute
    Hosini (string x, string y, int z) { // Constructor with
parameters
      Name = x;
      Hosini2 = y;
      year = z;
    }
};

int main() {
  // Create Hosini  objects and call the constructor with
different values
  Hosini  HabibObJ1("Al_Husini *_*", "AlMasre", 1999);
  Hosini  HabibObJ2("Ali", "Mustafa", 1969);

  // Print values
  cout << HabibObJ1.Name << " " <<
HabibObJ1.Hosini2 << " " << HabibObJ1.year << "\n";
  cout << HabibObJ2.Name << " " <<
HabibObJ2.Hosini2 << " " << HabibObJ2.year << "\n";
```

ابو حبيب الحسيني

```
  return 0;
}
```


تمامًا مثل الوظائف، يمكن أيضًا تعريف المنشئات خارج الفصل. أولاً، قم بتعريف المُنشئ داخل الفئة، ثم قم بتعريفه خارج الفئة عن طريق تحديد اسم الفئة، متبوعًا بعامل تحليل النطاق :: ، متبوعًا باسم المُنشئ (وهو نفس الفئة):

مثال

```
class Hosini {       // The class
  public:        // Access specifier
    string Name;  // Attribute
    string Hosini2;  // Attribute
    int year;     // Attribute
    Hosini (string x, string y, int z); // Constructor
declaration
};

// Constructor definition outsحرر the class
Hosini ::Hosini (string x, string y, int z) {
  Name = x;
  Hosini2 = y;
  year = z;
}

int main() {
  // Create Hosini  objects and call the constructor with
different values
```

```
  Hosini  HabibObJ1("Al_Husini *_*", "AlMasre", 1999);
  Hosini  HabibObJ2("Ali", "Mustafa", 1969);

  // Print values
  cout << HabibObJ1.Name << "
" << HabibObJ1.Hosini2 << " " << HabibObJ1.year << "\
n";
  cout << HabibObJ2.Name << " " <<
HabibObJ2.Hosini2 << " " << HabibObJ2.year << "\n";
  return 0;
}
```

محددات الوصول

كيف استخدام محددات الوصول فى الكلاسات

الأساسية التي تظهر في جميع public الآن، أنت على دراية تامة بالكلمة
أمثلة فصولنا:

مثال

```
class Class {  // The class
  public:      // Access specifier
    // class members goes here
};
```

ابو حبيب الحسينى

الأساسية هي محدد الوصول. تحدد محددات الوصول كيفية public الكلمة الوصول إلى أعضاء الفئة (السمات والأساليب). في المثال أعلاه، الأعضاء .مما يعني أنه يمكن الوصول إليهم وتعديلهم من خارج الكود -public هم

ومع ذلك، ماذا لو أردنا أن يكون الأعضاء خاصين ومخفيين عن العالم الخارجي؟

:في (السى بلص)، هناك ثلاثة محددات الوصول

- public- يمكن الوصول إلى الأعضاء من خارج الفصل.
- private- لا يمكن الوصول إلى الأعضاء (أو مشاهدتهم) من خارج الفصل.
- protected- لا يمكن الوصول إلى الأعضاء من خارج الفصل، ولكن يمكن الوصول إليهم في الفصول الموروثة. سوف تتعلم المزيد عن الوراثة في وقت لاحق.

:private الأعضاءpublic وفي المثال التالي نوضح الاختلافات بين

مثال

```
class Class {
  public:   // Public access specifier
    int x;  // Public attribute
  private:  // Private access specifier
    int y;  // Private attribute
};

int main() {
  Class Hosini_Obj;
  Hosini_Obj.x = 25;  // Allowed (public)
  Hosini_Obj.y = 50;  // Not allowed (private)
```

ابو حبيب الحسيني

```
  return 0;
}
```

إذا حاولت الوصول إلى عضو خاص، يحدث خطأ:

```
error: y is private
```

تعلم برمجة C++ بالعربي Abu Habib Al-Husin

ملحوظة: من الممكن الوصول إلى أعضاء خاصين في الفصل باستخدام طريقة عامة داخل نفس الفصل.راجع الفصل التالي (**التغليف**) حول كيفية القيام بذلك.

نصيحة: يعتبر الإعلان عن سمات صفك خاصة (بقدر ما تستطيع) من الممارسات الجيدة. سيؤدي هذا إلى تقليل احتمال قيامك (أو الآخرين) بإفساد الكود. وهذا أيضًا هو العنصر الرئيسي لمفهوم **التغليف** ، والذي ستتعلم المزيد عنه في الفصل التالي.

ملاحظة: افتراضيًا، يكون جميع أعضاء الكلاس private إذا لم تحدد محدد وصول:

مثال

```
class Class {
  int x;  // Private attribute
```

```
  int y;  // Private attribute
};
```

التغليف

ما هو مفهوم التغليف

معنى التغليف هو التأكد من إخفاء الاجراء ات "الحساسة" عن المستخدمين. ولتحقيق ذلك، يجب عليك إعلان متغيرات/سمات الفئة على إذا كنت تريد أن يقرأ .(لا يمكن الوصول إليها من خارج الفئة)private أنها الآخرون قيمة عضو خاص أو يعدلوها، فيمكنك توفير أساليب الحصول والتعيين العامة.

الوصول إلى الخصائص الاعضاء

للوصول إلى سمة خاصة، استخدم أساليب "الحصول" و"الضبط" العامة:

مثال

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

class Habib{
  private:
    // Private attribute
    int salary;
```

```
  public:
    // Setter
    void Hosini_Set(int s) {
      salary = s;
    }
    // Getter
  int Hosini_Get() {
      return salary;
    }
};

int main() {
  Hosini_Obj;
  Hosini_Obj.Hosini_Set(50000);
  cout << Hosini_Obj.Hosini_Get();
  return 0;
}
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربى

شاهد المثال

والتي تم تقييد الوصول إليها. private هي السمة salary

تأخذ الطريقة العامة Hosini_Set() المعلمة (s) وتعينها للسمة salary(الراتب = s).

تقوم الطريقة العامة Hosini_Get() بإرجاع قيمة السمة الخاصة salary.

الفئة. يمكننا الآن Employee نقوم بإنشاء كائن من ،main() في الداخل الطريقة لتعيين قيمة السمة الخاصة Hosini_Set() استخدام

الطريقة الموجودة على الكائن Hosini_Get() على 50000. ثم نستدعي لإرجاع القيمة.

لماذا التغليف؟

- من الممارسات الجيدة الإعلان عن خصوصية سمات صفك (بقدر ما تستطيع). يضمن التغليف تحكمًا أفضل في اجراء اتك، لأنه يمكنك (أو غيرك) تغيير جزء واحد من الكود دون التأثير على الأجزاء الأخرى
- زيادة أمن الاجراء ات

الوراثة

ما هى الوراثة

في لغة (السى بلص)، من الممكن وراثة السمات والأساليب من فئة إلى أخرى. نقوم بتجميع "مفهوم الوراثة " إلى فئتين:

- فئة مشتقة (ابن) - الطبقة التي ترث من فئة أخرى.
- الفئة الأساسية (الأصل) - الفئة الموروثة منها.

الرمز: للوراثة من فئة، استخدم.

habib يرث الصنف (الابن) السمات والأساليب من Hosini. في المثال أدناه الصنف (الأصل):

ابو حبيب الحسينى

مثال

```
// Base class
class habib {
  public:
    string Name = "Ali";
    void Hamza() {
      cout << "Abu Habib Al Husini \n" ;
    }
};

// Derived class
class Hosini : public habib {
  public:
    string Hosini2 = "Mustafa";
};

int main() {
  Hosini  Alhusini;
  Alhusini.Hamza();
  cout << Alhusini.Name + " " + Alhusini.Hosini2;
  return 0;
}
```

ابو حبيب الحسيني

تعلم برمجة C++ بالعربى Abu Habib Al-Husin

لماذا ومتى نستخدم "الوراثة "؟

- إنه مفيد لإعادة استخدام الاكواد: إعادة استخدام سمات وأساليب فئة موجودة عند إنشاء فئة جديدة.

الوراثة متعدد المستويات

كيف استخدام الوراثة متعدد المستويات

يمكن أيضًا اشتقاق فئة من فئة واحدة مشتقة بالفعل من فئة أخرى.

في المثال التالي، Hosini_GrandChild مشتق من الفئة Hosini_Child(المشتقة من Class).

مثال

```
// Base class (parent)
class Class {
  public:
    void HabibFunction() {
      cout << "Some content in parent class." ;
    }
};

// Derived class (child)
class Hosini_Child: public Class {
```

ابو حبيب الحسيني

```
};

// Derived class (grandchild)
class Hosini_GrandChild: public Hosini_Child {
};

int main() {
  Hosini_GrandChild Hosini_Obj;
  Hosini_Obj.HabibFunction();
  return 0;
}
```

تابع الوراثة المتعددة

مثال اخر على الوراثة المتعدد

يمكن أيضًا اشتقاق الفئة من أكثر من فئة أساسية واحدة، باستخدام قائمة مفصولة بفواصل:

مثال

```
// Base class
class Class {
  public:
    void HabibFunction() {
      cout << "Some content in parent class." ;
```

ابو حبيب الحسيني

```
    }
};

// Another base class
class Hosini_OtherClass {
  public:
    void Hosini_OtherFunction() {
      cout << "Some content in another class." ;
    }
};

// Derived class
class Hosini_ChildClass: public Class, public Hosini_Oth
erClass {
};

int main() {
  Hosini_ChildClass Hosini_Obj;
  Hosini_Obj.HabibFunction();
  Hosini_Obj.Hosini_OtherFunction();
  return 0;
}
```

أبو حبيب الحسيني

الوصول إلى الوراثة

اساليب الوصول

لقد تعلمت من فصل **محددات الوصول** أن هناك ثلاثة محددات متوفرة في لغة (السى بلس). حتى الآن، استخدمنا فقط public(يمكن الوصول إلى الأعضاء إلا) private و (أعضاء الفصل من خارج الفصل لا يمكن الوصول إلى protected يشبه private، ولكن يمكن المحدد الثالث .(داخل الفصل الوصول إليه أيضًا في الفئة الموروثة :

مثال

```
// Base class
class Habib{
  protected: // Protected access specifier
    int salary;
};

// Derived class
class Programmer: public Habib{
  public:
    int bonus;
    void Hosini_Set(int s) {
      salary = s;
    }
    int Hosini_Get() {
      return salary;
    }
};

int main() {
```

ابو حبيب الحسيني

```
  Programmer Hosini_Obj;
  Hosini_Obj.Hosini_Set(50000);
  Hosini_Obj.bonus = 15000;
  cout << "Salary: " << Hosini_Obj.Hosini_Get() << "\n";
  cout << "Bonus: " << Hosini_Obj.bonus << "\n";
  return 0;
}
```

تعدد الأشكال

كيف عمل تعدد الأشكال

تعدد الأشكال يعني "أشكال عديدة"، ويحدث عندما يكون العديد من الفئات التي ترتبط ببعضها البعض عن طريق الوراثة .

كما حددنا في الفصل السابق؛ **يتيح لنا الوراثة** وراثة السمات والأساليب من فئة أخرى. يستخدم تعدد الأشكال تلك الأساليب لأداء مهام مختلفة. وهذا يسمح لنا بتنفيذ إجراء واحد بطرق مختلفة.

والتي لها طريقة Abib على سبيل المثال، فكر في فئة أساسية تسمى Hosini(). يمكن أن تكون فئات الحيوانات المشتقة هي الخنازير، والقطط، والكلاب، والطيور - ولها أيضًا تطبيقها الخاص لصوت الحيوان (صوت الخنزير، ومواء القطة، وما إلى ذلك):

ابو حبيب الحسيني

مثال

```
// Base class
class Abib {
  public:
    void Hosini() {
      cout << "The Abib makes a sound \n";
    }
};

// Derived class
class Pig : public Abib {
  public:
    void Hosini() {
      cout << "The pig says: wee wee \n";
    }
};

// Derived class
class Ha : public Abib {
  public:
    void Hosini() {
      cout << "The Ha says: bow wow \n";
    }
};
```

تذكر من **فصل الوراثة** أننا نستخدم :الرمز للوراثة من الفصل.

ابو حبيب الحسيني

الآن يمكننا إنشاء كائنات Pig وتجاوز Ha الطريقة Hosini():

مثال

```
// Base class
class Abib {
  public:
    void Hosini() {
      cout << "The Abib makes a sound \n";
    }
};

// Derived class
class Pig : public Abib {
  public:
    void Hosini() {
      cout << "The pig says: wee wee \n";
    }
};

// Derived class
class Ha : public Abib {
  public:
    void Hosini() {
      cout << "The Ha says: bow wow \n";
    }
};
```

ابو حبيب الحسيني

لماذا ومتى نستخدم "الوراثة " و"تعدد الأشكال"؟

- إنه مفيد لإعادة استخدام الاكواد: إعادة استخدام سمات وأساليب فئة موجودة عند إنشاء فئة جديدة.

نظام الملفات

كيف التعامل مع الملفات

تتيح لنا المكتبة fstream العمل مع الملفات.

لاستخدام fstream المكتبة، قم بتضمين كل من الملف <fstream>: وملف الرأس <iostream> القياسي

مثال

```
#include <iostream>
#include <fstream>
```

هناك ثلاث فئات مدرجة في fstream المكتبة، والتي تستخدم لإنشاء الملفات أو كتابتها أو قراءتها:

كيف إلإنشاء والكتابة فى ملف

أو حدد اسم الملف fstream الفئة ofstream لإنشاء ملف، استخدم إما.

للكتابة في الملف، استخدم عامل الإدراج (<<).

مثال

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

int main() {
  // Create and open a text file
  ofstream Hosini_File("Habib_File_Name.txt");

  // Write to the file
  Hosini_File << "Files can be tricky, but it is fun
enough!";

  // Close the file
  Hosini_File.close();
}
```

لماذا نغلق الملف؟

تعتبر هذه ممارسة جيدة، ويمكنها تنظيف مساحة الذاكرة غير الضرورية.

ابو حبيب الحسينى

كيف قراءة ملف

واسم الملف fstream أو ifstream للقراءة من ملف، استخدم الفئة التي تنتمي) الوظيفة getline() حلقة مع while لاحظ أننا نستخدم أيضًا لقراءة سطر الملف سطرًا، ولطباعة محتوى الملف (الفئة ifstream إلى:

مثال

```
// Create a text string, which is used to output the text
file
string HabibTxt;

// Read from the text file
ifstream Hosini_ReadFile("Habib_File_Name.txt");

// Use a while loop together with the getline() function
to read the file line by line
while (getline (Hosini_ReadFile, HabibTxt)) {
  // Output the text from the file
  cout << HabibTxt;
}

// Close the file
Hosini_ReadFile.close();
```

ابو حبيب الحسيني

استثناءات

ماهى استثناءات

عند تنفيذ كود (السى بلص)، يمكن أن تحدث أخطاء مختلفة: أخطاء في الترميز يقوم بها المبرمج، أو أخطاء بسبب إدخال خاطئ، أو أشياء أخرى غير متوقعة.

عند حدوث خطأ، سيتوقف (السى بلص) عادةً ويصدر رسالة خطأ. المصطلح الفني لهذا هو: سيطرح (السى بلص) استثناءً (رمي خطأ).

كيف التقاط الخطا اذا حدث

تتكون معالجة الاستثناءات في لغة (السى بلص) من ثلاث كلمات رئيسية : و try:throwcatch

تحديد كتلة من الاكواد ليتم اختبارها بحثًا عن الأخطاء أثناء try يتيح لك الاجراء تنفيذها.

الأساسية استثناءً عند اكتشاف مشكلة، مما يتيح لنا إنشاء throw تطرح الكلمة خطأ مخصص.

تحديد كتلة من الاكواد التي سيتم تنفيذها، في حالة حدوث catch يتيح لك الاجراء خطأ في كتلة المحاولة.

في أزواج catch الرئيسية try تأتي الكلمات:

مثال

```
try {
  // Block of code to try
  throw exception; // Throw an exception when a
problem arise
}
```

ابو حبيب الحسيني

```
catch () {
  // Block of code to handle errors
}
```

خذ بعين الاعـتبار المثال التالي:

مثال

```
try {
  int age = 15;
  if (age >= 18) {
    cout << "Access granted - you are old enough.";
  } else {
    throw (age);
  }
}
catch (int Habib3) {
  cout << "Access denied - You must be at least 18 years
old.\n";
  cout << "Age is: " << Habib3;
}
```

ابو حبيب الحسيني
Abu Habib Al-Husin تعلم برمجة C++ بالعربى

شاهد المثال

كان المتغير أقل من 18، age الكتلة لاختبار بعض الاكواد: إذا try نستخدم الكتلة الخاصة بنا catch باستثناء، ونتعامل معه في throw فسنقوم.

تأخذ catch. الكتلة، نكتشف الخطأ ونقوم بشيء حياله في catch لأننا نطرح استثناءً (Habib3) (متغيرًا int العبارة معلمة : في مثالنا نستخدم int النوع في try الكتلة (age)). لإخراج قيمة age.

بدلاً من 15, مما يعني if age is 20 على سبيل المثال) إذا لم يحدث أي خطأ (أنه سيكون أكبر من 18 ، catch الكتلة فسيتم تخطي:

مثال

```
int age = 20;
```

Abu Habib Al-Husin تعلم برمجة C++ بالعربي

الكلمة الأساسية لإخراج رقم مرجعي، مثل رقم/رمز throw يمكنك أيضًا استخدام خطأ مخصص لأغراض التنظيم:

مثال

```
try {
  int age = 15;
  if (age >= 18) {
    cout << "Access granted - you are old enough.";
  } else {
    throw 505;
  }
}
catch (int Habib3) {
  cout << "Access denied - You must be at least 18 years
old.\n";
```

ابو حبيب الحسيني

```
  cout << "Error number: " << Habib3;
}
```

التعامل مع أي نوع من الاستثناءات بهذة (...) العلامة

إذا كنت لا تعرف throw النوع المستخدم في try الكتلة، فيمكنك استخدام catch صيغة "النقاط الثلاث" (...) داخل الكتلة، والتي ستتعامل مع أي نوع من الاستثناءات:

مثال

```
try {
  int age = 15;
  if (age >= 18) {
    cout << "Access granted - you are old enough.";
  } else {
    throw 505;
  }
}
catch (...) {
  cout << "Access denied - You must be at least 18 years
old.\n";
}
```

ابو حبيب الحسيني

ابو حبيب الحسيني

ابو حبيب الحسيني