esprit

Ecole Supérieure Privée
d'Ingénierie et de Technologies

www.esprit.ens.tn

Se former autrement

pour une nouvelle génération d'ingénieurs

Test & validation

Conformément à la certification ISTQB (International Software Testing Qualification Board)

Tests et validation

- Introduction
- Chapitre 1 : Fondamentaux de Test
- Chapitre 2 : Test pendant le cycle de vie
- Chapitre 3 : Techniques statiques
- Chapitre 4 : Technique de tests
- Chapitre 5 : Gestion des tests
- Chapitre 6 : Outils de support aux tests
- Laboratoire de tests

esprit
Ecole Supérieure Privée d'Ingénierie et de Technologies

Plan

- Objectifs
- Introduction
- Le processus de développement de tests.
- Catégories de techniques de tests
- Techniques basées sur les spécifications boîte noire.
- Techniques de tests basées sur la structure boîte blanche.
- Techniques basées sur l'expérience.
- Conclusion
- Exercices

Objectifs

- Différencier la spécification de conception de test de la spécification des cas de test et des spécifications de procédures de test.
- Comprendre et différencier les termes : condition de test, cas de test et procédure de test.
- Savoir différencier les techniques basées sur les spécifications (boite blanche) et la technique boite noire)
- Comprendre les différences entre les tests basés sur les spécifications, les tests basés sur les structures et les tests basés sur l'expérience .

Introduction

- Les tests sont définis selon un processus.
- Plusieurs types de tests.
- Plusieurs techniques de tests.
- ….

- Comment concevoir ces tests?
- Existe-t-il différents techniques de tests?
- Quels sont les meilleures techniques à appliquer?

Processus de développement de Test(1)

- Phase d'analyse de conception des tests,

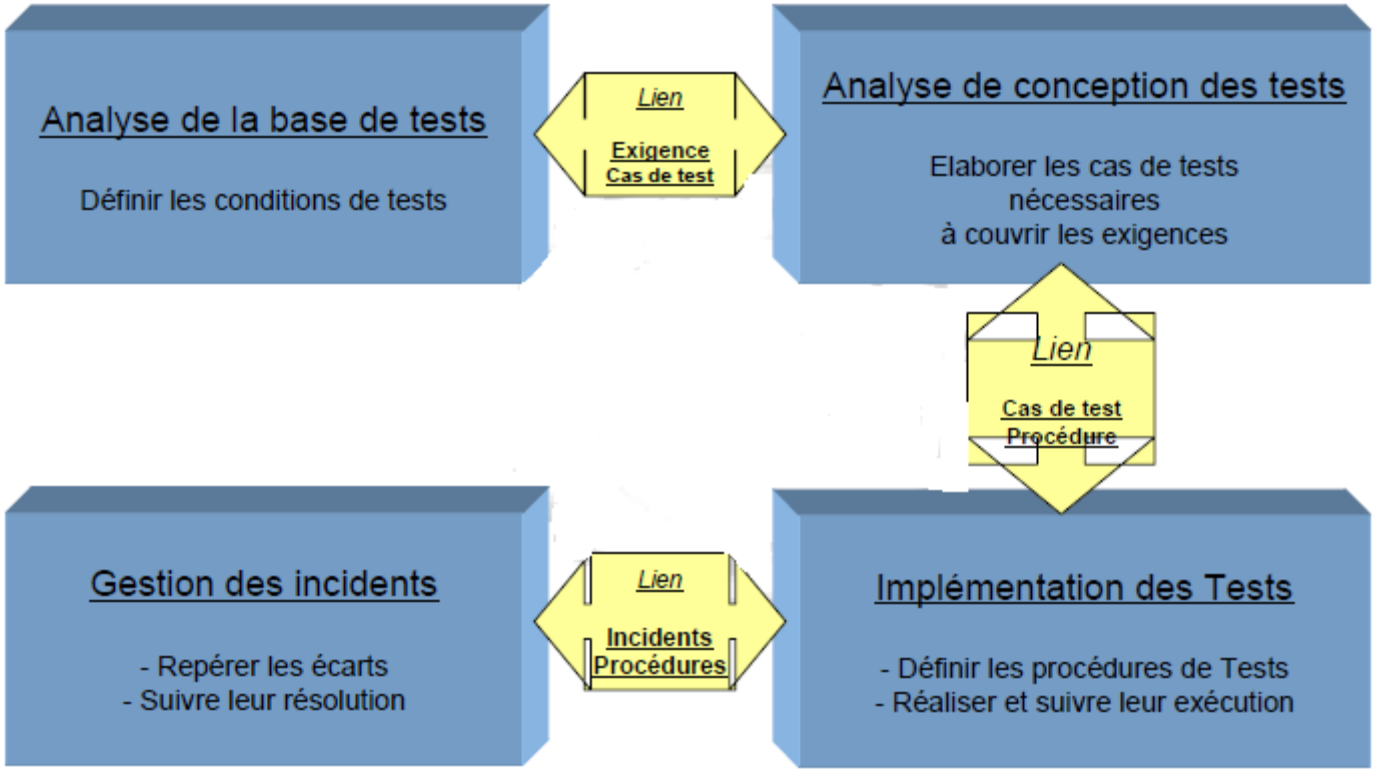
  Analyser la documentation de la base des tests déterminer ce qui est à tester, c'est à dire à identifier les conditions de test.

- Déterminer la cartographie des exigences **(la traçabilité des conditions de tests vers** les spécifications et exigences**)**

- Classer des exigences en fonction du:
  - **Type de test**
  - **Niveau de l'exigence**
  - **Importance de l'exigence**
  - **Fragilité possible du produit**
- Les cas de test , procédures de tests et données de test sont créés et spécifiés dans le plan de tests

Processus de développement de Test(2)

- **Traçabilité entre élément de tests**



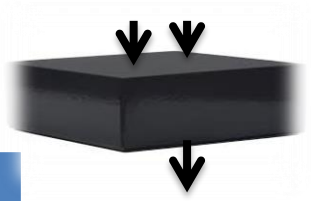
Catégories de techniques de test

- Objectif : identifier les conditions de tests, les cas de test et les données de tests.

Technique Boite Noire	Technique Boite Blanche
Appelée technique basée sur les spécifications) Une façon de dériver et de sélectionner : -Les conditions de tests, les cas de test ou les données de test en se basant sur une analyse de la documentation de la base des tests. -N'utilise aucune information concernant la structure interne d'un composant ou système à tester. -Utilisation des soit formels soit informels, pour la spécification des problèmes à résoudre.	Dites techniques structurelles ou basées sur les structures) Sont basées sur une analyse de la structure du composant ou du système. Basé sur le niveau de couverture du logiciel peut être mesuré à partir de cas de test existants, et des cas de test complémentaires peuvent être dérivés de façon systématique pour augmenter la couverture.

Techniques basées sur les spécifications fonctionnelles(boite noire)

- Tester en considérant le logiciel comme « une boîte noire » et basé sur la modélisation et le comportement .
- Les moyens sont:



1. **Partition d'équivalence**
2. **Analyse des valeurs limites**
3. **Test par table de décision**
4. **Transition d'états pour déterminer les scénarios fonctionnels**
5. **Tests de cas d'utilisation**

Partitions d'équivalence(1/2)

- Partition d'équivalence : Un groupe d'entrées d'un logiciel ou système qui montre un comportement similaire, donc un traitement identique est appelé partitions (ou classes) d'équivalence.
- Il suffit de prendre une des données du groupe pour valider la partition d'équivalence en entier.
- Des tests peuvent être conçus pour couvrir toutes les partitions valides et invalides.
- Les partitions d'équivalence sont applicables à tous les niveaux de tests.

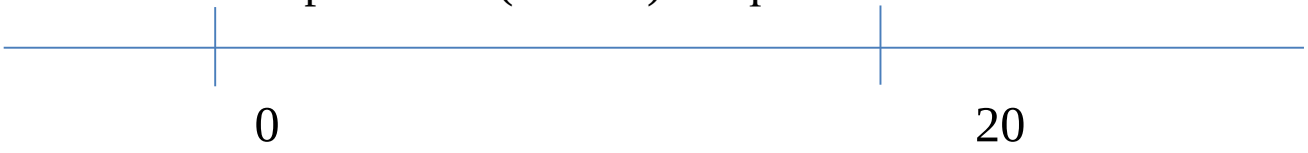
Partitions d'équivalence(2/2)

❑ Exemple:

Un utilitaire permet la saisie des frais de repas pour les salariés d'une entreprise. Les règles définies par la DRH sont:

- ***Seuls les repas inférieurs à 20€ sont remboursés***
- ***Si le montant saisi est inférieur à 0 €, message: « Remboursement impossible »***
- ***Si le montant saisi est égal à 0 €, message: « Remboursement non autorisé »***
- ***Si le montant saisi est supérieur à 20€, message: « Remboursement non autorisé »***

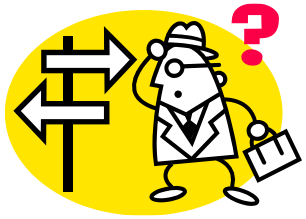
Déterminer le nombre de partitions (classes) d'équivalence



En déduire le nombre de cas de tests minimum pour atteindre 100% de couverture des partitions d'équivalences.

Analyse des valeurs limites

- **Valeur limite :** une valeur d'entrée ou de sortie qui est au bord d'une partition, ou à la distance minimale d'un incrément de chaque côté de cette limite, par exemple le minimum ou le maximum d'une plage de valeurs.

- **Il y a 3 types de valeurs:**
  - Valeurs valides 
  - Valeurs invalides 
  - Valeurs aux limites  

- **Exemple :** le mois de février,

Années non bissextiles : 27 = valeur valide, 28 = valeur aux limites, 29 = valeur invalide

Années bissextiles : 28= valeur valide, 29 = valeur aux limites, 30= valeur invalide.

Table de décision(1/2)

- **Table de décision :**

  Table montrant la combinaison des entrées et/ou stimuli (causes) et de leurs sorties et/ou actions (effets) associées, qui peut être utilisée pour concevoir des cas de tests.

- **Pour appliquer la technique, on doit** :

  - Identifier les différentes conditions influant sur la décision
  - Identifier toutes les actions qui peuvent découler des combinaisons
  - Identifier toutes les combinaisons possibles de conditions
  - Construire la table
  - Vérifier la table avec les experts du processus
  - Simplifier la table de décision en éliminant les règles impossibles et en combinant celles qui ont trait à des conditions qui n'affectent pas vraiment la décision.

Table de décision(2/2)

❑ **Exemple**:

Produit assurance décès souscrit par des personnes salariées de 21 à 60 ans d'un capital de plus de 10 K€ et n'excédant pas 200 K€.

Conditions	R	E	G	L	E	S			
C1:Capital < 10k€	n	n	o	n	n	o	n	n	o
C2:Capital > 200k€	n	o	n	n	o	n	n	o	n
10k€= <C3:Capital =< 200k€	o	n	n	o	n	n	o	n	n
21 ans < C4: Age =< 60 ans	n	n	n	o	o	o	n	n	n
C5: Age< 21 ans	o	o	o	n	n	n	n	n	n
C6: Age> 60 ans	n	n	n	n	n	n	o	o	o
Actions									
A1: souscription autorisée				x					
A2: refus de la demande	x	x	x		x	x	x	x	x
N° cas de tests	1	2	3	4	5	6	7	8	9

Table de décision(2/2)

La notation courante dans les tables de décision est la suivante :

- Pour les conditions :
  - V signifie que la condition est vraie (peut aussi être représentée par O (Oui) ou 1)
  - F signifie que la condition est fausse (peut aussi être affiché comme N (Non) ou 0)
  - — signifie que la valeur de la condition n'a pas d'importance (peut également être affichée en

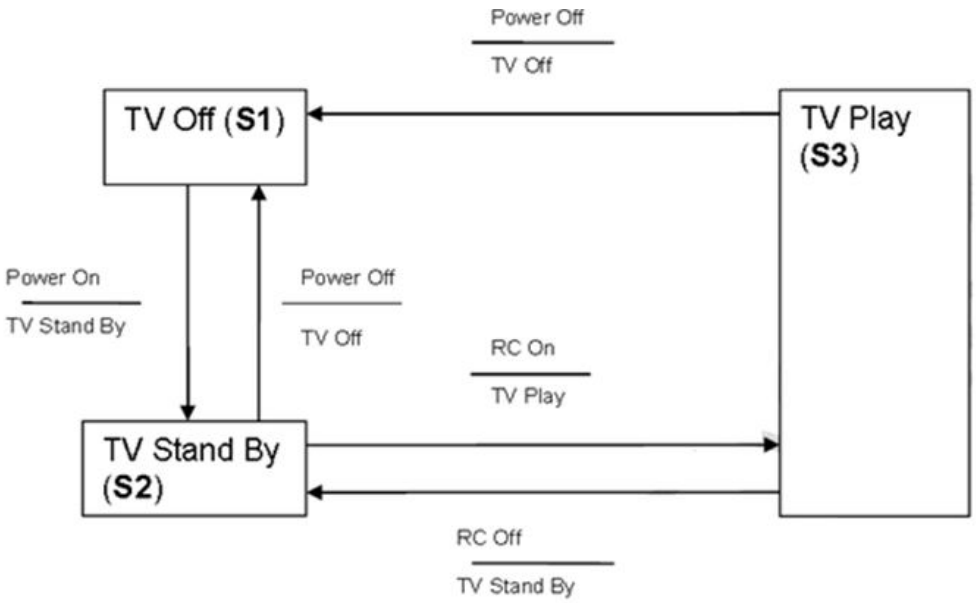
    tant que N/A)
- Pour les actions :
  - X signifie que l'action doit avoir lieu (peut aussi être représenté par V, O ou 1)
  - Vide signifie que l'action ne doit pas se produire (peut également être affiché sous la forme -

    ou F, N ou 0)

Tests de transition d'Etats

- Si on est capable de définir dans l'application les différents états que peut prendre une fonction ou un système
- Dans ce cas on définit :
  - les transitions
  - les prédicats
  - les actions
- On trace le graphe de ces critères et on applique certains résultats de la "Théorie des graphes" pour déterminer:
  - les chemins minimum
  - Les circuits etc..
- Ces résultats permettent de réaliser les procédures de tests, de mesurer des couvertures

Tests de transition d'Etats



Test case	1	2	3	4	5
Start state	S1	S2	S2	S3	S3
Input	Power On	Power Off	RC On	RC Off	Power Off
Expected output	TV Stand By	TV Off	TV Play	TV Stand By	TV Off
Finish State	S2	S1	S3	S2	S1

18

Tests de cas d'utilisation

Conception des cas de tests à partir des cas d'utilisation

- Tests spécifiés à partir des cas d'utilisation.
- Un cas d'utilisation peut inclure des variations possibles de son comportement de base
- Les tests sont conçus pour exercer les comportements définis (comportement de base, exceptionnel ou alternatif, et traitement des erreurs).
- La couverture peut être mesurée par le pourcentage des comportements de cas d'utilisation testés divisé par le nombre total des comportements du cas d'utilisation, généralement exprimé en pourcentage.

Technique de test basée sur la structure
Technique de test boîte blanche

- **Basé sur la structure du logiciel**
  - Test de couverture des instructions
  - Test de couverture des branche / de décision
  - Test de couverture des conditions

Graphe de flux d'un module

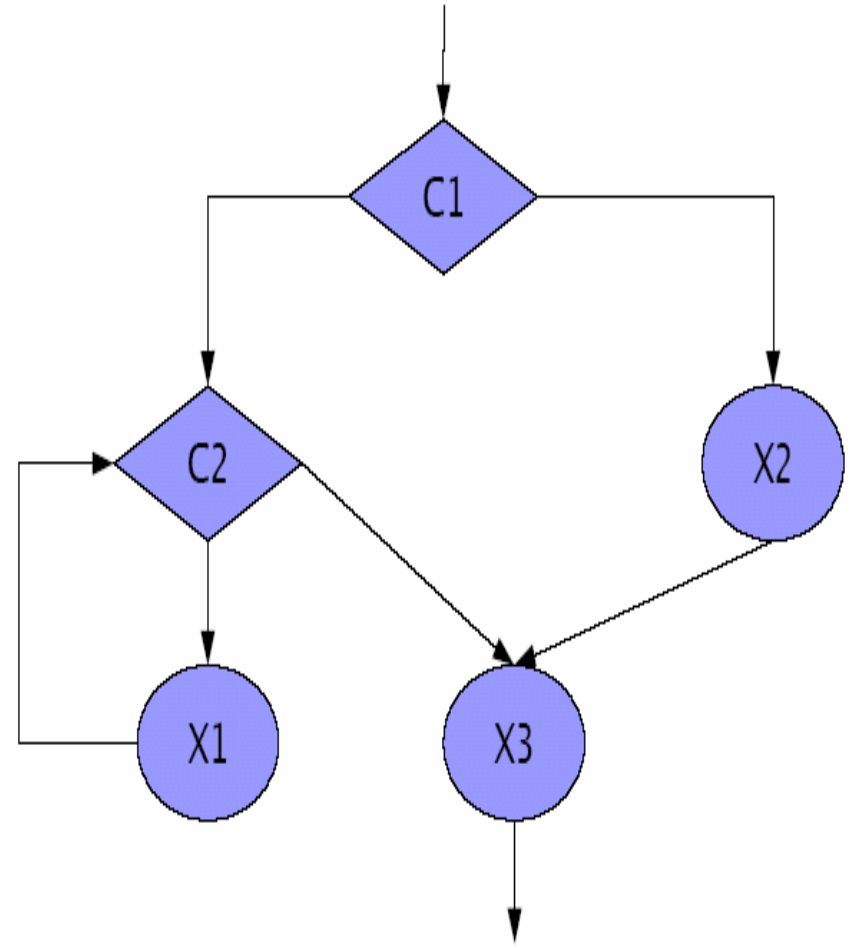
- Un nœud corresponds à:
  - Point de départ (initial).
  - Une instruction simple (affectation, lecture, écriture, …),
  - Le prédicat d'une instruction conditionnelle ou répétitive (if, while, for, …).
  - Point d'arrivée (sortie)
- Un arc entre deux nœuds correspond à un flux de contrôle possible.

Théorie des graphes, nombre cyclomatique

- $V(G) = e - n + NbS + NbE$
  - e = nombre d'arcs
  - n = nombre de nœuds
  - NbE / NbS = nombre d'entrées / nombre de sorties

V(G) = 6 – 5 +1 +1= 3

V(G): donne le nombre de chemins indépendants dans un algorithme



Couverture des tests structurels

- **Critère de couverture orienté flux de contrôle**
  - Instructions

    Entité dans un langage de programmation, qui est typiquement la plus petite unité indivisible d'exécution.
  - *Branches*

    Un bloc de base qui peut être sélectionné pour exécution, basé sur une construction programmatique dans laquelle un chemin, parmi deux ou plus, est disponible par exemple : case, jump, go to, if-then-else.
  - *Décisions*

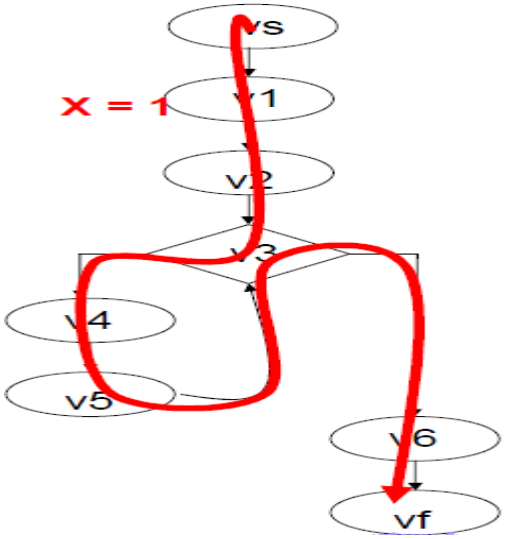
    Un nœud avec deux ou plus liens vers des branches séparées.
  - *Conditions*

    Expression logique qui peut être évaluée à Vrai ou Faux.

Test des instructions et couverture

- La couverture des instructions est l'évaluation du pourcentage d'instructions exécutables qui ont été exercées par une suite de cas de test. Le test des instructions dérive des cas de test pour exécuter des instructions spécifiques, pour accroître la couverture des instructions.
- Une suite de tests satisfait le critère de **Couverture des instructions** si chaque instruction est couverte par l'un des chemins de la suite de tests.
- La couverture des instructions est déterminée par le nombre d'instructions exécutables couvertes par des cas de test (conçus ou exécutés) divisé par le nombre de toutes les instructions exécutables dans le code testé.

```
vs:     int main(int x) {
        int sum, I;
v1:     sum = 0;
v2:     i = 1;
v3:     while (i <= x) {
v4:     sum = sum + i;
v5:      i = i + 1;
         }
v6:     printf("%d",sum);
vf: }
```



Test des décisions et couverture

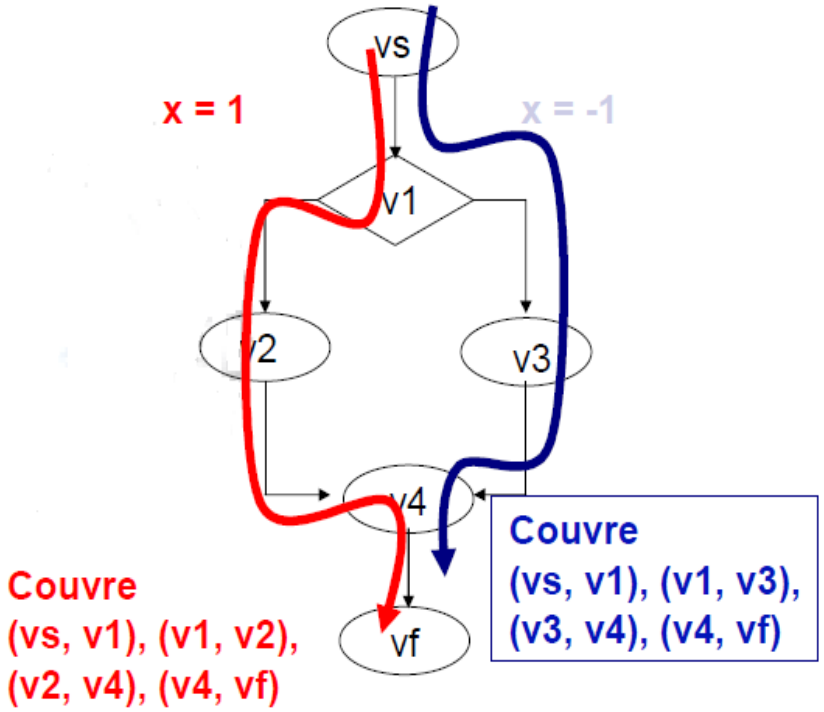
- La couverture des décisions, liées aux tests de branches, est l'évaluation des pourcentages de résultats de décisions (p.ex. les options Vrai et Faux d'une instruction IF) qui ont été traitées par une suite de cas de test. Les branches trouvent leur origine dans les points de décision du code et montrent le transfert du contrôle vers différents endroits du code.

Une suite de tests satisfait le critère de couverture des branches si chaque branche est couverte par l'un des chemins de la suite de tests.

Branches: **(vs, v1), (v1, v2), (v1, v3),**

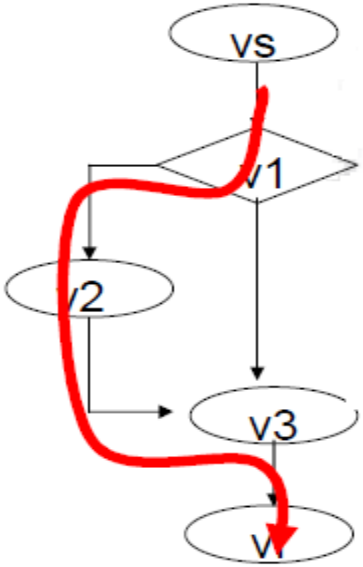
- **(v2, v4), (v3, v4), (v4, vf)**

```
vs:   int main(int x) {
        int result;
v1:   if (x > 0)
v2:   result = x;
else
v3:   result = 1 / x;
v4:   printf("%d", result);
vf: }
```

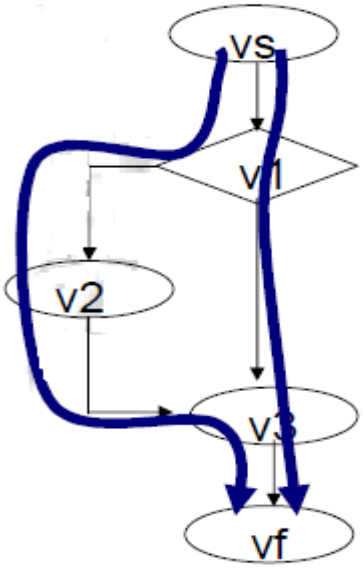


Couverture d'instructions et de branches

- La couverture des Branches inclut la couverture des instructions.
- Si un ensemble de tests satisfait au critère de couverture des branches, **alors il satisfait aussi** au critère de couverture des instructions.

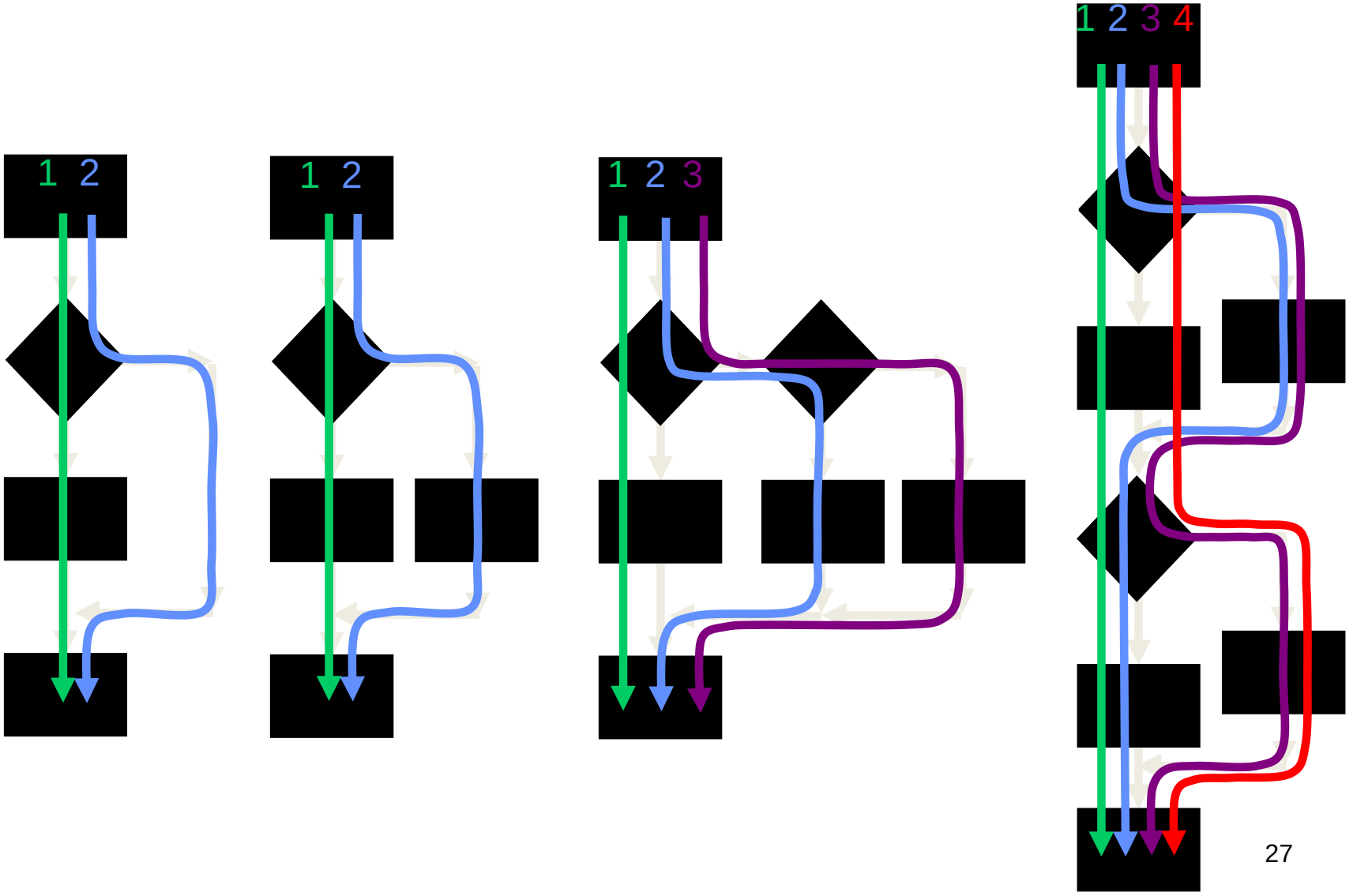


Couvertures des Instructions



Couverture des Branches

Chemins dans le code



Décisions & Conditions(1)

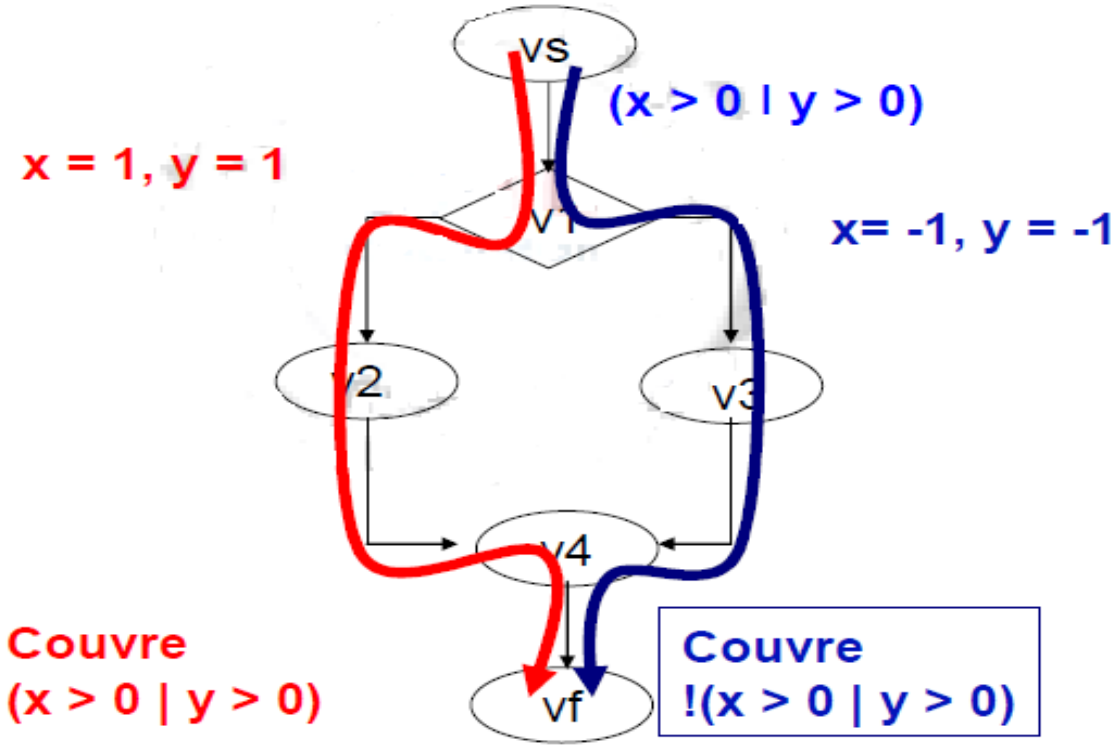
- **Décision**
  - Le prédicat d'instruction conditionnelle ou répétitive
  - Exemple : (x > 0 | y > 0)
- **Condition**
  - Expression booléenne inclue dans une décision
  - Exemple : (x > 0), (y > 0)

```
vs:        int main(int x,
               y) {
           int result;
v1:        if (x > 0 | y > 0)
v2:            result = x;
           else
v3:            result = y;
                   v4:
    printf("%d", result);
           vf: }
```

Décisions & Conditions(1)

Couverture des décisions:

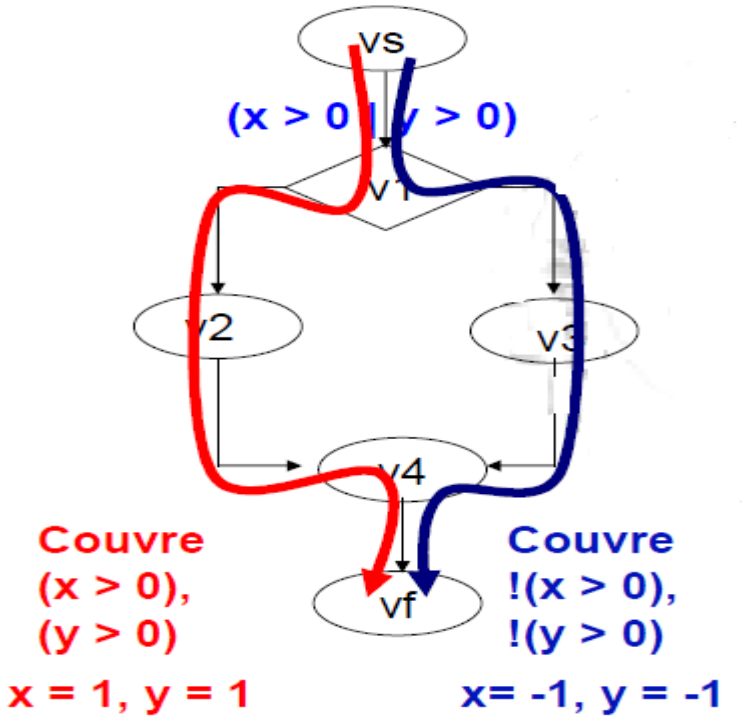
- Couvre toutes les décisions et leur contraire.
- Est équivalent au critère de couverture des branches.

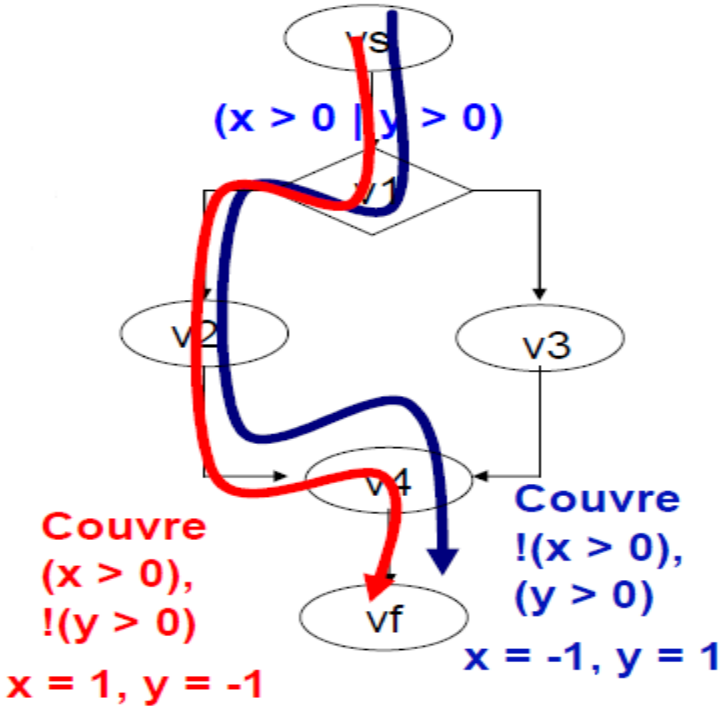


Décisions & Conditions(2)

Couverture des conditions:

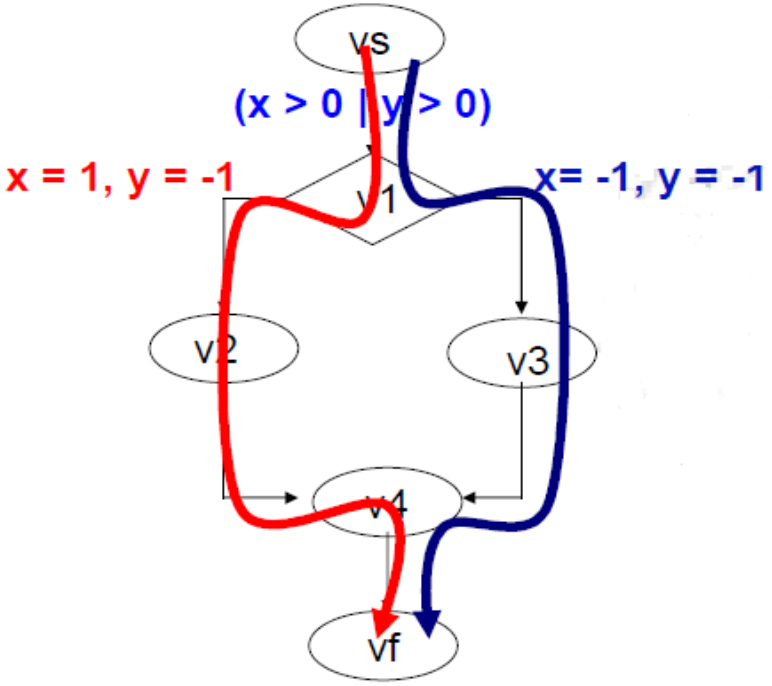
- Couvre chaque condition et son contraire



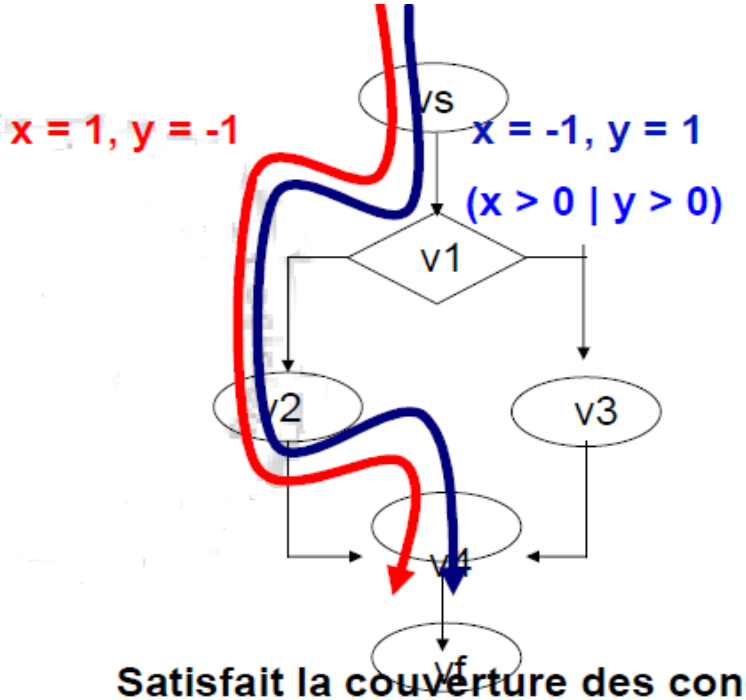


Décisions & Conditions (3)

Les critères de couverture des Décisions et des conditions ne sont pas comparables:



Satisfait la couverture des décisions
Mais pas la couverture des conditions

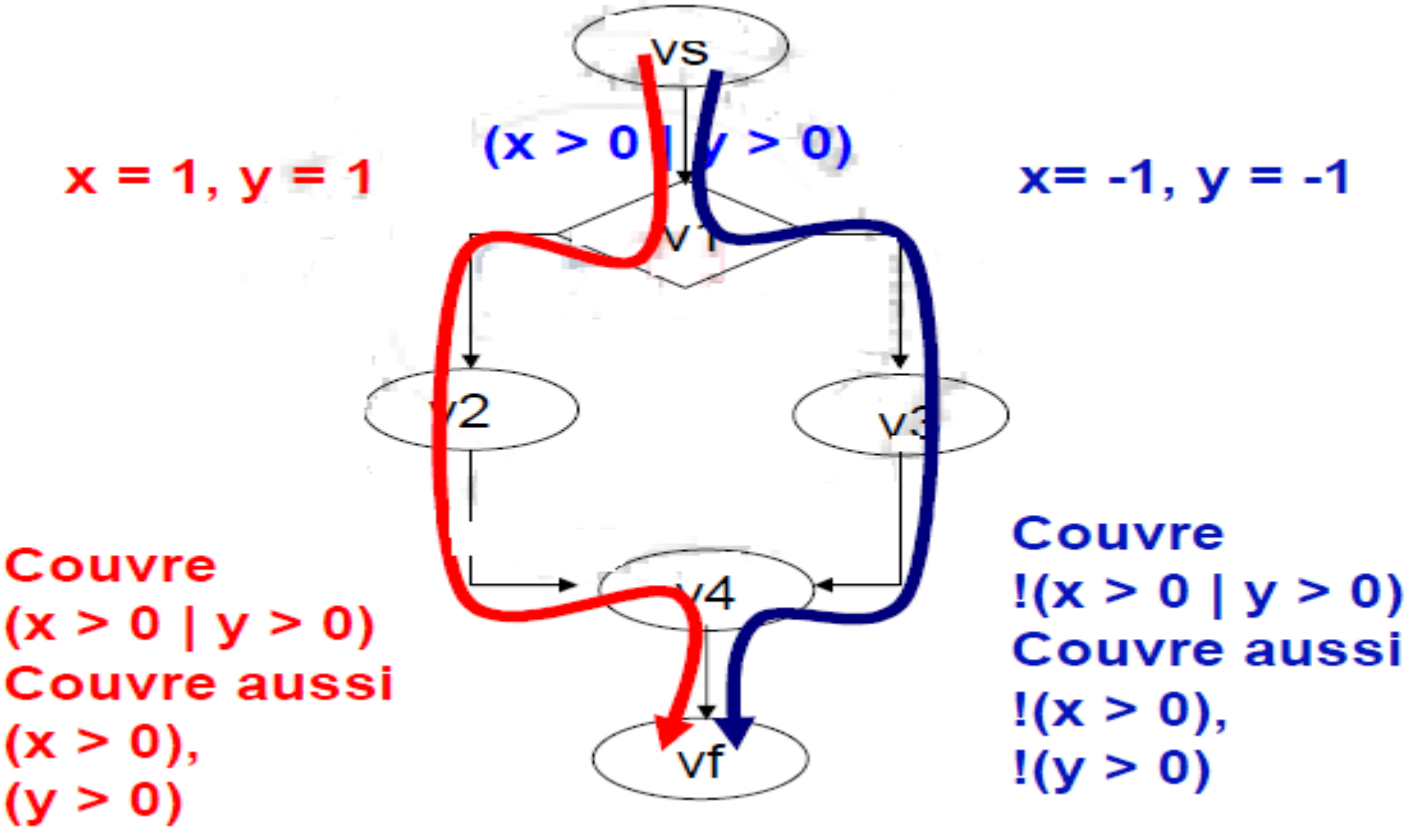


Satisfait la couverture des conditions
Mais pas la couverture des décisions

Décisions & Conditions (4)

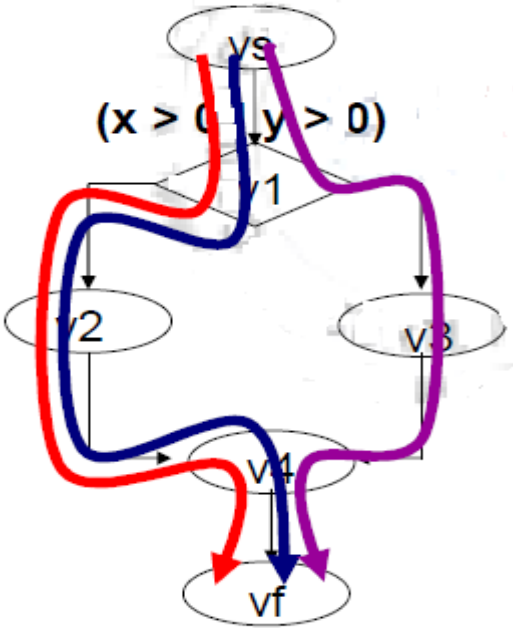
Couverture des Conditions/Décisions

- Combine couverture des conditions & des Décisions



Décisions & Conditions (5)

- **Couverture de Conditions/Décisions Modifiées** (Modified condition/decision coverage ou MC/DC):
  - Chaque condition influe indépendamment la décision, on varie les conditions une à une pour une décision avec 'n' conditions.



Pour (x >0 | y>0) :

x = -1, y = 1

x = 1, y = -1

x = -1, y = -1

Techniques basées sur l'expérience

- Estimation d'erreurs
  - *Basé sur l'expérience personnelle de chaque testeur*
    - ✓ Utilisées en plus des techniques systématiques
    - ✓ Utilisation de listes de typologie d'erreurs (souvent incomplètes)

  - Une approche méthodique de la technique par estimation d'erreur consiste à créer une liste d'erreurs, de défauts et de défaillances possibles, et à concevoir des tests qui exposeront ces défaillances et les défauts qui les ont causées.

  - L'estimation d'erreur est une technique utilisée pour anticiper les erreurs, les défauts et les défaillances, sur la base des connaissances du testeur
    - Comment l'application a fonctionné antérieurement.?
    - Quels types d'erreurs les développeurs ont tendance à faire?
    - Les défaillances qui se sont produites dans d'autres applications

Techniques basées sur l'expérience

Les Tests Exploratoires (Exploratory Testing)

- Les tests exploratoires sont le plus utile lorsqu'il y a peu de spécifications ou des spécifications inadéquates ou des contraintes de temps importantes sur les tests.
- Les résultats des tests sont utilisés pour en apprendre davantage sur le composant ou le système
- les tests exploratoires sont des tests informels (non prédéfinis) et sont conçus, exécutés, enregistrés et évalués dynamiquement pendant l'exécution des tests
- les tests exploratoires sont effectués parfois dans un temps défini, et le testeur utilise une charte de test comprenant les objectifs du test pour guider les tests.
- Les tests exploratoires peuvent inclure l'utilisation d'autres techniques boîte-noire, boîte-blanche et basées sur l'expérience.

Techniques basées sur l'expérience

- Tests basés sur des checklists
  - Dans les tests basés sur des checklists, les testeurs conçoivent, implémentent et exécutent des tests pour couvrir les conditions de test figurant dans une checklist.
  - Les checklists peuvent être construites sur la base de l'expérience, de la connaissance de ce qui est important pour l'utilisateur, ou de la compréhension du pourquoi et du comment des défaillances logicielles
  - Au cours de l'analyse, les testeurs peuvent céer une nouvelle checklist ou complètent une checklist existante, mais ils peuvent également utiliser une checklist existante sans modification.

Conclusion

- Le choix des techniques de tests à utiliser dépend de différents facteurs incluant :

  Le type de système les standards réglementaires,

  les exigences client ou contractuelles, le niveau de risque, le type de risque, les objectifs de test, la documentation disponible, la connaissance des testeurs, le temps disponible et le budget, le cycle de vie de développement utilisé, les modèles de cas d'utilisation et l'expérience sur les défauts découverts précédemment.

- Le test exhaustif est en général impossible à réaliser:

  En test fonctionnel, l'ensemble des données d'entrée est en général infini ou de très grande taille.

esprit
Ecole Supérieure Privée d'Ingénierie et de Technologies

Exercice

1. Le bout de code suivant contient une erreur potentielle de division par 0

```
J=50
K=1
N=3
while (J>=10) and (N<=10) loop
M [K] = J/N
K = K + 1
N = N - 1
end loop
```

Laquelle de ces méthodes celle qui sera la plus efficace pour détecter cette erreur ?

A. Test aux limites
B. Conditions de test
C. Compilation du code source
D. Inspection du code source

2. Pour un site Internet d'un grand magasin, on utilise des symboles différents pour chaque saison. On veut tester si cette fonction marche correctement. Quel type de test est effectué avec les valeurs « 1er janvier 2009 », « 31 avril 2010 », « 30 juin 2009 » et « 1er octobre 2008 » ?

A. Le test basé sur l'expérience
B. Une analyse des classes d'équivalences
C. Une analyse statique
D. Une analyse des valeurs limites

esprit
Ecole Supérieure Privée
d'Ingénierie et de Technologies

Exercice

3. Combien de cas de test faut-il pour atteindre 100% de couverture des décisions dans le fragment de programme ci-après ? Les conditions sont indépendantes les unes des autres.

```
IF(condition1)
THEN
        statement1
ENDIF
IF (condition2)
THEN
        statement2
ENDIF
```

A. 1 cas de test
B. Plus de 3 cas de test
C. 2 cas de test
D. 3 cas de test

4. Lequel de ces tests celui qui n'appartient pas aux techniques boîtes noires

A. Test de décision
B. Partition d'équivalence
C. Table de décision
D. Test de transition d'état

esprit
Ecole Supérieure Privée
d'Ingénierie et de Technologies

Exercice

5. Quelle affirmation concernant les techniques de test white box et black box est exacte ?

A. La partition d'équivalence, la table de décision et le test flux de données sont des techniques de test de type Black Box
B. La partition d'équivalence, les valeurs aux limites et les tests d'instruction sont des techniques de test de type Black Box
C. La partition d'équivalence, les transitions d'états et les tests des cas d'utilisation sont des techniques de test de type Black Box
D. La partition d'équivalence, les transitions d'états, les tests des cas d'utilisation et la table de décision sont des techniques de test de type White Box

6. Si la température descend sous les 18° C, le chauffage se met en marche, si elle atteint 21°C, le chauffage s'éteint. Quel est l'ensemble minimum de valeurs couvrant toutes les partitions d'équivalence

A. 15, 19 et 25 °C
B. 17, 18, 20 et 21 °C
C. 18, 20 et 22 °C
D. 16 et 26 °C

Exercice

6. L'analyse des valeurs limites fournit des cas de test intéressants, car ...

A. Des classes d'équivalence identiques d'un point de vue fonctionnel sont prises en considération dans les cas de test.
B. Lors de la programmation de cas distincts, des erreurs sont souvent commises aux « bords » des domaines de définition.
C. Cette analyse est un standard industriel.
D. Le fonctionnement d un système peut s'avérer dangereux au-delà de ses limites.

7. Pour le fragment de code suivant, laquelle des réponses celle qui représente le minimum de tests requis respectivement pour la couverture de branches et des instructions

```
Discount rate=1;
Fare = 1000;
Si ((person == 'senior citizen') ET (travel month = 'January'))
        Bonuspoints = 100+Bonuspoints
FINSI
SI(class=='first')
        discountRate = 5;
        Fare = fare * discountRate;
FINSI
```

A. Couverture d'instructions : 1 , couverture de branches : 2
B. Couverture d'instructions : 2 , couverture de branches : 2
C. Couverture d'instructions : 1 , couverture de branches : 3
D. Couverture d'instructions : 2, couverture de branches : 4

esprit
Ecole Supérieure Privée
d'Ingénierie et de Technologies

Exercice

8. La couverture d'instructions ne pourra pas contrôler :

A. Les instructions manquantes
B. Les branches non utilisés
C. Le code mort
D. Les instructions non utilisées

9. Laquelle des énumérations suivantes ne contient que des tests boîte noire ?

A. Couverture des instructions; couverture des branches; analyse du flux de données.
B. Couverture des branches; formation des classes d'équivalence; analyse des valeurs limites.
C. Couverture des instructions; couverture des branches; test d état et de transitions d états.
D. Formation des classes d'équivalence; analyse des valeurs limites; test d'état et de transitions d'états

esprit
Ecole Supérieure Privée
d'Ingénierie et de Technologies

Exercice

10. La mesure de couverture :

A. N'a aucun rapport avec les tests.
B. Est une mesure partielle du test.
C. Est mesuré par la couverture des branches, qui devrait être obligatoire pour tous les logiciels.
D. Ne peut être appliqué ni à l'unité ou aux tests de modules, ni aux tests du système.

11. Une entrée qui présente l'année de naissance entre 1900 et 2004. les valeurs des tests aux limites sont :

A. 0, 1900, 2004, 2005
B. 1900, 2004
C. 1899, 1900, 2004, 2005
D. 1899, 1900, 1901, 2003, 2004, 2005

12. La partition d'équivalence est :

A. Une technique de test de catégorie black box utilisée uniquement par les développeurs
B. Une technique de test de catégorie black box appropriée à tous les niveaux de test
C. Une technique de test de catégorie black box utilisée uniquement pendant les tests systèmes
D. Une technique de test de catégorie White box utilisée lors du test composant