Université libre de Bruxelles

Faculté des Sciences

Départment d'informatique

Langages de programmation

(INFO-F105)

Première année du programme de bachelier en Science Informatique

Examen — juin 2024

- L'examen comprend quatre questions, chacune notée sur 10 points
- La pondération associée à chaque sous-question est fournie
- Répondez à chaque question sur une feuille séparée fournie à cet effet
- Indiquez vos nom, prénom et numéro d'étudiant sur chaque feuille de réponse
- Toutes les notes de cours, transparents et syllabus sont autorisés
- Calculatrices, ordinateurs, téléphones interdits
- Durée de l'examen : deux heures trente
- Pour rappel, la note de cours est consituée de soit
  - 2/3 examen + 1/3 projet
  - 100% examen

  (la note la plus favorable des deux).
- Ecrivez de façon lisible...

Question 1

Répondez aux questions sur base du code ASM ci-dessous.

```
1  section .data
2  V dw 2, 3, 5, 7, 11, 29, 23, 19, 17, 13
3  n dw 10
4
5  section .text
6  global CMAIN
7  CMAIN:
8      movzx ecx, word [n]
9      dec ecx
10 label1:
11     movzx edx, word [n]
12     dec edx
13 label2:
14     mov ax, [V+2*edx]
15     cmp ax, [V+2*(edx-1)]
16     jnb label3
17     mov bx, [V+2*(edx-1)]
18     xchg bx, [V+2*edx]
19     mov [V+2*(edx-1)], bx
20 label3:
21     dec edx
22     jnz label2
23     loop label1
24     ret
```

1. Expliquez en une phrase ce qu'est l'*endianness* (le boutisme), et donnez un exemple de situation dans laquelle il est important de savoir si le système est *big-endian* ou *little-endian*.

   [2 points]

2. En deux phrases, expliquez ce que fait le code ASM ci-dessous.

   [4 points]

3. Quel est l'état du vecteur `V` à la fin de l'exécution ?

   [1 point]

4. Peut-on placer des nombres négatifs dans `V` ? Motivez votre réponse.

   [2 points]

5. Le mot-clef **word** est nécessaire à la ligne **8** mais pas à la ligne **14**. Pourquoi ?

   [1 point]

6. **(BONUS [1 point])** Si on remplace la ligne **3** par `n dw 0`, comment pouvez-vous adapter le code pour retrouver la valeur de `[n]` ?

Question 2

Étant donné le code C++ suivant, répondez aux questions suivantes. **Toutes vos réponses doivent être justifiées (*brièvement*).**

1. Qu'affiche le programme lors de l'exécution de la ligne **24** ?
   [1 point]
2. Quel concept permet les définitions des fonctions aux lignes **9** et **13** ?
   [1 point]
3. Qu'affiche le programme lors de l'exécution des lignes **26** et **28** ?
   [2 points]
4. Que se passerait-il si les lignes **36** à **39** étaient supprimées ?
   [1 point]
5. Que se passerait-il si la ligne **3** était supprimée ?
   [1 point]
6. Qu'affiche le programme à la ligne **31** ?
   [2 points]
7. Justifiez la présence des mot-clés **static** et **inline** à la ligne 5.
   [2 points]

```cpp
#include <iostream>

void f3(int x);

static inline void afficher_variables(int a, int b, int c){
    std::cout << a << " " << b << " " << c << std::endl;
}

void f1(int* x){
    std::cout << "Hello, World !" << std::endl;
}

void f1(int x){
    std::cout << "World, Hello !" << std::endl;
}

void f2(int* x){
    *x /= 2;
}

int main(){
    int a=3, b=5, c=10, d=3;

    f1(a);

    afficher_variables(b, c, d);
    f2(&b);
    afficher_variables(b, c, d);

    f3(c);
    afficher_variables(a, c, d);

    return 0;
}

void f3(int x){
    int a = 0;
    ++x;
}
```

Question 3

Vous faites partie de l'équipe technique des scénaristes d'Hollywood de la série *The Office*. On vous demande d'écrire un outil permettant d'automatiser les possibilités de scénario de l'épisode "*Dinner Party*". Voici le cahier des charges.

1. Implémentez une structure **Guest** avec :
   — un attribut **name**, qui indiquera le nom de l'invité.
   — un attribut **jobTitle**, qui indiquera le travail de l'invité.

   [1 point]

2. Implémentez une classe **DinnerParty** avec :
   — un attribut *publique* **at**, qui indiquera l'endroit où le dîner aura lieu.
   — un attribut *publique* **guests**, un tableau, qui stockera les invités présents au dîner. Un nombre maximum de 10 invités est possible pour chaque dîner.
   — une méthode *publique* **countGuests** qui retourne le nombre d'invités.

   [2 points]

3. Supposons que la variable `guest` est une instance de la structure **Guest** et que `dinnerParty` est une instance de la classe **DinnerParty**.

   Implémentez les surcharges d'opérateur » et « pour supporter les 2 opérations ci-dessous :

   (a) `guest >> dinnerParty`, qui ajoutera `guest` au tableau de `guests` de l'instance `dinnerParty`.

   (b) `dinnerParty << guest`, qui ajoutera `guest` au tableau de `guests` de l'instance `dinnerParty`.

   Pour cette question 3, vous êtes libre de choisir les types de retour de vos surcharges, tout en gardant une certaine cohérence.

   [3 points]

4. Implémentez les surcharges d'opérateur « pour permettre l'affichage d'instances de `Guest` et `DinnerParty` (via `cout` par exemple) :

   (a) `cout << guest`, qui affichera une instance de **Guest** sous le format suivant, dans le cas où `name` de guest correspond à Jimothy et `jobTitle` à Salesman : `Jimothy (Salesman)`. Voir également l'exemple d'utilisation.

   [1 point]

   (b) `cout << dinnerParty`, qui affichera une instance de **DinnerParty**. Le format d'affichage est donné dans l'exemple d'utilisation ci-dessous.

   [2 points]

5. L'implémentation de la classe **DinnerParty** respecte-elle le principe d'encapsulation ? Justifiez votre réponse, et si pas suggérez un changement.

   [1 point]

Passez à la page suivante pour voir l'exemple d'utilisation !

Exemple d'utilisation de votre code :

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

// votre code pour la question 3 se trouve ici

int main() {
    Guest dwight = {
      .name="Dwight",
      .jobTitle="Assitant (to the) Regional Manager"
    };
    Guest angela = {
    .name="Angela",
    .jobTitle="Small Accountant"
    };
    Guest pam = {.name="Pam", .jobTitle="Receptionist"};
    Guest jimothy = {.name="Jimothy", .jobTitle="Salesman"};
    DinnerParty partyAtTheOffice = {
      .at="Dunder Mifflin", .guests={}
    };
    DinnerParty partyAtMichael = {
      .at="Michael's Condo", .guests={}
    };
    jimothy >> partyAtMichael;
    dwight >> partyAtTheOffice;
    partyAtMichael << pam;
    partyAtTheOffice << angela;
    cout << partyAtMichael << endl;
    cout << partyAtTheOffice << endl;
    return 0;
}
```

Affichera ceci (2 lignes affichées) :

Dinner party at Michael's Condo with 2 guests : Jimothy (Salesman) Pam (Receptionist)

Dinner party at Dunder Mifflin with 2 guests : Dwight (Assitant (to the) Regional Manager) Angela (Accountant)

Question 4

Considérez le programme C++ suivant.

```
1  #include <iostream>
2  using namespace std;
3
4  int cnt = 0;
5
6  int f1(int a, int& b, int* c, long int d){
7      cnt++ ; a++ ; b++ ; c++ ; d++ ;
8      return b ;
9  }
10
11 int f2(int a, int b){
12     return a-b ;
13 }
14
15 int main() {
16     int a = 0 ;
17     int b = 1 ;
18     int* c= new int (a) ;
19     int& d = a ;
20     int* e = &b ;
21     a++ ; b++ ; (*c)++ ; d--; (*e)++;
22     int f= (a==d) && (c == e);
23     cout<<"a="<<a<<" b="<<b<<" *c="<<*c;
24     cout<<" d="<<d<<" *e="<<*e<< "f="<<f<<endl;
25
26     long int powT [] = {1,2,4,8,16,32,64,128,256,512,1024};
27     long int g = (&powT[3]+2)[-4] - ((powT[3]+2)-4);
28     cout<<"g="<<g<<endl;
29
30     int h= powT[5]*powT[6]*powT[7]*powT[8]*powT[9];
31     cout<<"h="<<h<<endl;
32
33     f1(a,a,&a,a);
34     cout<<"cnt="<<cnt<<" a="<<a<<endl;
35
36     int i = (a==a) || (f1(a,a,&a,a)) ;
37     cout<<"cnt="<<cnt<<" a="<<a<<endl;
38
39     int j = f2(f1(a,b,c,d), f1(a,b,c,d));
40     cout<<"j="<<j<<endl;
41
42     return 0;
43 }
```

Répondez aux questions suivantes en supposant que le programme est exécuté sur une architecture 32-bit.

1. Qu'est-ce que le programme affichera en lignes 23-24 ? Justifiez votre réponse.
   [2 points]
2. Qu'est-ce que le programme affichera en ligne 28 ? Justifiez votre réponse.
   [1 point]
3. Qu'est-ce que le programme affichera en ligne 31 ? Justifiez votre réponse.
   [1 point]
4. Quelles sont les variables de *main* dont les valeurs seront modifiées par la ligne 33, et qu'est-ce que le programme affichera en ligne 34 ? Justifiez vos réponses.
   [2 points]
5. Quelles sont les variables de *main* dont les valeurs seront modifiées par la ligne 36, et qu'est-ce que le programme affichera en ligne 37 ? Justifiez vos réponses.
   [2 points]
6. Quelles sont les variables de *main* dont les valeurs seront modifiées par la ligne 39, et qu'est-ce que le programme affichera en ligne 40 ? Justifiez vos réponses.
   [2 points]

Examen INFO-F105 Langages de Programmation 1 ; examen – juin 2024

NOM, PRENOM (en majuscules) :
Numéro d'étudiant :

Numéro de question : 1

---

1. L'endianness correspond à l'ordre dans lequel les bytes individuels d'un objet sont stockés. En little-endian les bytes de poids faible correspondent aux plus petites adresses alors qu'en big-endian, les bytes de poids faible sont mis aux plus grandes adresses. Cette information est importante à savoir dès qu'il va être question de lire certains bytes individuellement (par exemple lors de l'implémentation d'opérations arithmétiques sur des entiers trop grands pour rentrer dans un registre, c.f. le TP 9).
2. Le code donné trie le vecteur `V` par l'algorithme *bubble sort* (tri bulle) : on parcourt $n-1$ fois le vecteur `V` (de n entiers) de droite à gauche et on compare chaque élément avec son prédécesseur. À chaque fois, s'il est plus petit que ce dernier, on les échange.
3. Le vecteur `V` sera trié à la fin et vaudra donc :
   `2, 3, 5, 7, 11, 13, 17, 19, 23, 29`.
4. Non : l'instruction **`jnb`** (pour *jump if not below*) correspond à une comparaison non signée. Pour une comparaison signée, il aurait fallu utiliser l'instruction **`jnl`** (pour *jump if not less*).
5. À la ligne 8, la taille de sortie est connue puisqu'il s'agit des 32 bits de **`ecx`**, mais la taille d'entrée ne peut pas en être déduite pour l'instruction **`movzx`**. L'instruction **`mov`** quand à elle ne peut fonctionner que sur des opérandes de même taille, et donc les 16 bits de **`ax`** déterminent automatiquement le nombre de bits à aller lire en mémoire.
6. Si nous savons que les données mises dans la section **`.data`** sont stockées de manière contiguë et en little-endian (comme en IA-32), alors nous pouvons retrouver la valeur de `n` comme ceci :

```
1  section .text
2       ...
3       mov esi, n
4       sub esi, V
5       shr esi, 1
6       mov [n], si  ; si = les 16 LSB de esi
7       ...
```

Examen INFO-F105 Langages de Programmation 1 ; examen – juin 2024

NOM, PRENOM (en majuscules) :
Numéro d'étudiant :

Numéro de question : 2

---

1. Le programme affiche "World, Hello!". En effet, le paramètre passé lors de l'appel à **f1** est de type **`int`**, ce qui correspond à la signature de la fonction définie à la ligne 13.
2. Les deux définitions aux lignes 9 et 14 sont permises par la notion de **surcharge**.
3. A la ligne 26, le programme affiche "5 10 3". Ensuite, la fonction **f2** divise (en division **entière**, impliquée par les types de `*x` et du diviseur) la valeur pointée par `x`. La ligne 28 affiche alors "2 10 3".
4. Une erreur serait levée à la compilation pour la production d'un exécutable : la fonction **f3** ne serait pas définie alors qu'elle est appelée dans la fonction main.
5. Si cette ligne est supprimée, l'appel à la fonction **f3** dans la fonction `main` ne pourra pas être réalisé, comme cette fonction n'est pas encore connue du compilateur. La compilation ne pourra pas donc être effectuée.
6. L'appel à la fonction **f3** ne modifie aucune des variables définies dans la fonction `main`. En effet, il y définit une variable **locale** `a` (qui ne modifie donc pas la valeur de la variable `a` définie dans la fonction `main`), et l'opération d'incrémentation est effectuée sur une **copie** de la valeur passée en paramètre lors de l'appel.
7. En déclarant la fonction comme **statique**, on s'assure que la fonction ne sera appelable que dans l'unité de compilation correspondant au fichier d'exemple. Le mot-clé `inline` permet de demander au compilateur de substituer les appels à la fonction `afficher_variables` avec le corps de cette fonction.

Examen INFO-F105 Langages de Programmation 1 ; examen – juin 2024

NOM, PRENOM (en majuscules) :
Numéro d'étudiant :

Numéro de question : 3

```
1  #include <iostream>
2
3  using namespace std;
4
5  const int N_GUESTS = 10;
6
7  struct Guest {
8      string name;
9      string jobTitle;
10 };
11
12 class DinnerParty {
13 public:
14     string at;
15     Guest guests[N_GUESTS];
16     int guestCounter = 0;
17     int countGuests() const {
18         return guestCounter;
19     }
20 };
21
22 ostream& operator <<(ostream &s, const Guest& guest) {
23     s << guest.name << " (" << guest.jobTitle << ")";
24     return s;
25 }
26
27 ostream& operator<<(ostream &s, const DinnerParty& dp) {
28     s << "Dinner party at ";
29     s << dp.at << " with " ;
30     s << dp.countGuests() << " guests: ";
31     for (int i=0; i < dp.guestCounter; i++) {
32         cout << dp.guests[i] << " ";
33     }
34     return s;
35 }
36
37 DinnerParty& operator<<(DinnerParty &dp, const Guest& g) {
38     dp.guests[dp.guestCounter] = g;
```

```
39      dp.guestCounter++;
40      return dp;
41  }
42
43  void operator>>(const Guest& guest, DinnerParty &dp) {
44      dp << guest;
45  }
```

Examen INFO-F105 Langages de Programmation 1 ; examen – juin 2024

NOM, PRENOM (en majuscules) :
Numéro d'étudiant :

Numéro de question : 4

---

1. Le programme affiche `a=0 b=3 *c=1 d=0 *e=3 f=0`. Justification
   — `d` est un alias de `a` (référence), les opérations `a++` et `d--` affectent les deux variables ;
   — `c` est un pointeur vers une copie de `a` (en ligne 18), en ligne 20 cette copie est incrémentée ;
   — `e` est un pointeur vers l'adresse de `b`, la variable `b` est incrémentée deux fois en ligne 20 par les instructions `b++` et `(*e)++`.
2. Le programme affichera `g=-4`. On a `(&powT[3]+2)[-4]=powT[1]=2` et `((powT[3]+2)-4)=powT[3]-2=6`.
3. Le programme affichera `h=0`. La valeur de `powT[5]*powT[6]*powT[7]*powT[8]*powT[9]` est 2^{35}, qui est ensuite convertie en **`int`**. Lors de la conversion, les bits de poids fort sont perdus pour ne garder que les $n = 16$ ou $n = 32$ bits de poids faible et il reste $0 = 2^{35} \bmod 2^n$.
4. Le programme affichera `cnt=1 a=1`. Seule `a` est modifiée via le deuxième paramètre de `f1` (`cnt` est également modifié car variable globale).

   Notons que le troisième paramètre de `f1` est l'adresse de `a` mais cette adresse est passée en copie ; l'incrément de `c` en ligne 7 (un incrément de l'adresse, pas de la valeur pointée) n'a aucun effet au sein de la fonction `main`.

   Notons aussi que `d` est une référence, pointeur constant dont la valeur est l'adresse de `a`, donc non modifié (même si en apparence comme il est implicitement déréférencé il pourrait sembler modifié, dans le sens où afficher `cout << d` avant et après l'opération donneraient des valeurs différentes).
5. Le programme affichera `cnt=1 a=1`. Seule `i` est modifiée : le premier opérande de `||` étant vrai, le deuxième opérande n'est pas évalué
6. Les variables modifiées sont `j` (évidemment), `cnt` (modifiée par chaque instance de f1) et `b` (car passée par référence pour chaque instance de f1). Les autres variables passées en paramètres le sont par copie, et donc ne changent pas. Notons que `e` contient l'adresse de `b` et bien que `b` change, `e` lui-même ne change pas.

   Le résultat est *INDÉFINI* car il dépend de l'ordre d'évaluation des opérandes de `f2`, qui n'est pas défini par le standard. De nombreux compilateurs évalueront ces opérandes de droite à gauche, et dans ce cas le programme affiche `j=1` (après évaluation du deuxième opérande celui-ci et `b` valent 2 ; après évaluation du premier opérande celui-ci et `b` valent 3).

Remarque : de façon générale, j'ai accordé plus de points à l'explication que à la valeur affichée.