Podstawy Programowania 2014/15

Zadania projektowe – ogólne wytyczne

I. Organizacja zajęć

Poniżej znajduje się orientacyjny schemat obrazujący realizację poszczególnych typów zajęć. W szczególności, odbiory pierwszego zadania projektowego nastąpią w 10 i 11 tygodniu zajęć, natomiast odbiory drugiego zadania projektowego nastąpią w dwóch ostatnich tygodniach zajęć. Terminy te są orientacyjne i dokładne daty zostaną podane później (każdy student będzie miał przydzielony indywidualny termin i miejsce odbioru).

<table>
<tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td></tr>
<tr><td colspan="8">Wykład</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>L0</td><td>L1</td><td>L2</td><td>L3</td><td>L4</td><td>L5</td><td>L6</td><td>L7</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td colspan="6">LO – laboratorium otwarte</td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>Odbiór Projektu 1</td><td></td><td></td><td></td><td colspan="2">Odbiór Projektu 2</td></tr>
</table>

II. Przebieg realizacji każdego z zadań projektowych

1. Student pisze program realizujący zadanie projektowe.
   - Program należy pisać samodzielnie.
   - Pomoc można uzyskać u wyznaczonych do tego celu osób, które są do dyspozycji studenta w podanych terminach i we wskazanych salach.
   - Program musi spełniać szczegółowe wytyczne podane w instrukcji projektowej (do każdego projektu przygotowana jest osobna instrukcja).
2. Student wysyła ukończony program korzystając z serwisu przeznaczonego do tego celu. Instrukcja postępowania zostanie podana w momencie rozpoczęcia realizacji części projektowej przedmiotu. Przesłanie programu musi nastąpić przed wyznaczoną datą (spóźnienie oznacza 0 punktów za dany projekt).
3. W wyznaczonym terminie (indywidualnym dla każdego studenta) następuje ocena rozwiązania (tzw. odbiór projektu). Odbiór projektu ma charakter rozmowy z nauczycielem, która przebiega następująco:
   - Student przybywa do wskazanej sali wcześniej w stosunku do wyznaczonego terminu, i przygotowuje się do rozmowy. Przygotowanie do rozmowy obejmuje:
     - zajęcie miejsca przy dowolnie wybranym (wolnym) komputerze,
     - otworzenie napisanego kodu w odpowiednim edytorze (jeśli program jest umieszczony w wielu plikach, wszystkie pliki należy otworzyć w kolejnych „zakładkach”)
     - skompilowanie kodu i upewnienie się, że wszystko działa prawidłowo.
   - Nauczyciel zadaje pytania mające na celu sprawdzenie wiedzy, samodzielności

wykonania programu i umiejętności studenta. Dokładne wytyczne podane są w punkcie III poniżej.
    - Nauczyciel ocenia program informując studenta o liczbie przyznanych punktów.
4. Po zakończeniu semestru następuje porównanie wszystkich programów przesłanych przez studentów w celu wykrycia ewentualnych plagiatów. Student, który dokonał plagiatu, musi liczyć się z poważnymi konsekwencjami z tego tytułu.

III. Wymagania stawiane przed studentem podczas odbioru zadań projektowych

1. Student prezentuje program rozwiązujący zadanie projektowe (sprawdzanie ogólnego sensu wypowiedzi i użycie określonego słownictwa).
2. Student potrafi wyjaśnić znaczenie i sposób użycia poszczególnych zmiennych i konstrukcji programistycznych wykorzystanych w aplikacji. Student odpowiada na pytania dotyczące funkcjonalności, oraz potrafi wskazać fragmenty kodu odpowiedzialne za uzyskanie określonego (wybranego przez nauczyciela) efektu w ocenianej aplikacji.
3. Student zna alternatywne (omówione na wykładzie) konstrukcje programistyczne pozwalające na uzyskanie wskazanych efektów w aplikacji.
4. Student potrafi (czas około 5 minut) wprowadzać modyfikacje w kodzie prowadzące do uzyskania efektu wskazanego przez prowadzącego.
5. Program jest wykonany przez studenta samodzielnie.

IV. Wytyczne dotyczące stylu programowania przy realizacji projektów

1. Należy używać kwalifikatorów const lub dyrektyw #define w stosunku do wszystkich wartości, które nie ulegają zmianie w trakcie wykonywania programu.
2. Należy unikać umieszczania stałych liczbowych w kodzie. Zamiast tego, warto użyć dyrektywy #define i posługiwać się identyfikatorami (nazwami) zamiast stałymi liczbowymi. Poprawi to czytelność programu i pozwoli na łatwą jego modyfikację.
3. Należy stosować wcięcia w kodzie w sposób konsekwentny. Typowo przyjęte rozmiary wcięć to 3, 4 lub 8 spacji. Wcięcia powinny być robione zawsze tym samym znakiem: spacją albo tabulacją (nie należy mieszać tych znaków).
4. Unikamy umieszczania wielu instrukcji w jednej linii.
5. Stosujemy komentarze, zarówno nagłówkowe, jak i wyjaśniające wybrane (mniej intuicyjne) miejsca w kodzie. Komentarze nie powinny być parafrazą instrukcji, tzn. zamiast opisywać co robimy, powinny dawać sugestię dlaczego tak robimy, np. zamiast:

   ```
   if ( wiek >= 18 ) // jeśli wiek >= 18
   ```

   piszemy:

   ```
   if ( wiek >= 18 ) // jeśli osoba jest pełnoletnia
   ```

6. Używamy białych znaków do oddzielenia fragmentów kodu. Liczba białych znaków powinna być proporcjonalna do „siły” rozdzielenia. Na przykład, odstępy (liczba linii) pomiędzy funkcjami nie powinny być mniejsze niż odstępy pomiędzy fragmentami kodu tej samej funkcji, np.:

```
int funkcja1() {
  etap1
        // oddzielamy przerwą kolejne etapy funkcji
  etap2

  etap3
}
        // oddzielamy funkcje - „silniejszy” separator niż powyżej

int funkcja2() {
  etap1

  etap2
}
```

Separatory w wyrażeniach sugeruje się stosować następująco. Jeśli stosujemy różne odległości pomiędzy znakami w wyrażeniach, także należy pamiętać "sile" podziału - im więcej znaków tym większa, czyli nie piszemy:

```
a = a+b * c-d
```

tylko

```
a = a + b*c – d.
```

Pierwszy przypadek sugeruje wyrażenie (a+b)*(c-d), co nie jest prawdą. Podobnie:

```
a=b + c // wygląda jak (a=b)+c
```

sugeruje się zastąpić:

```
a = b+c // wygląda jak a=(b+c)
```

7. Program dzielimy na funkcje. Funkcja powinna wykonywać dobrze określoną czynność i mieć ograniczoną liczbę linii kodu (np. funkcja mieści się na stronie ekranowej komputera).
8. Funkcje powinny komunikować się z otoczeniem poprzez swoje parametry. Nie należy używać zmiennych globalnych, chyba że instrukcja projektowa zawiera informację o możliwości zakwalifikowania konkretnie wskazanych zmiennych jako globalne.
9. Nie należy stosować instrukcji goto.