Zad. 4: Rotacje 3D

1 Cel ćwiczenia

Wykształcenie umiejętności modelowania kluczowych dla danego problemu pojęć. Generowanie dokumentacji z wykorzystaniem systemu `doxygen`. Praktyczne zweryfikowanie wcześniejszej konstrukcji programu. Jeśli nie jest dobrze napisany, to rozszerzenie wymagane w tym zadaniu będzie trudniejsze.

2 Program zajęć

- *Ocena realizacji zadania z poprzedniego laboratorium* – ocenie podlega poprawność realizacji zadania, styl pisania programu oraz dokumentacja wygenerowana za pomocą programu *doxygen*.
- *Ocena przygotowania do zajęć* – ocenie podlega przykładowa implementacja szablonu funkcji zgodnie z wytycznymi przedstawionymi w rozdziale 4.
- *Modyfikacja programu wg wskazań osoby prowadzącej* – ocenie będzie podlegała poprawność realizacji modyfikacji. Pracę nad modyfikacją programu (wszystkie operacje należy wykonywać na kopii) należy rozpocząć już w trakcie pierwszej fazy laboratorium, gdyż prowadzący nie będzie w stanie ocenić wcześniejszego programu wszystkim jednocześnie.
- *Realizacja wstępnej fazy prac nad nowym zadaniem* – w ramach wstępnej realizacji zadania należy przetestować skalowaność stworzonych dotychczasowych struktur danych takich jak wetor i macierz. Operacja na nich powinny poprawnie działać przy zwiększeniu rozmiaru np. do 5. Jednym z takich testów powinno być ustawienie obrotu dla jednej z maciern np. o 50°, zaś dla drugiej obrotu o -50°. Ich przemnożenie powinno dać macierz jednostkową.
- *Ocena realizacji wstępnej fazy zadania*

3 Opis zadania programowego

Niniejsze zadanie jest kontynuacją wcześniejszego zadania. Obiektem, który ma być poddawany rotacjom jest prostopadłościan. Zasadnicza część menu programu będzie taka sama. Dochodzą jednak nowe pozycje związane z tym, że program powinien być w stanie składać obroty wokół kolejnych osi zgodnie z żądanym przez użytkownika porządkiem.

- obrót bryły o zadane kąty względem wybranych osi z zadaną ilością powtórzeń wraz ze sprawdzenie długości dwóch przeciwległych boków,
- powtórzenie poprzedniego obrotu,
- wyświetlenie macierzy rotacji,
- przesunięcie bryły o zadany wektor,
- wyświetlenie współrzędnych wierzchołków,

- wyświetlenie menu,
- zakończenie działania programu.

W tym zadaniu użytkownik powinien mieć możliwość zadania kąta obrotu, jak też osi układu współrzędnych (tzn. *OX*, *OY* lub *OZ*), względem której ma być dokonany obrót. Program powinien umożliwić wprowadzenie dowolnej sekwencji obrotów względem poszczególnych osi. Wspomnianą sekwencję kończy wprowadzenie znaku kropki zamiast oznaczenia osi obrotu. Natomiast oznaczeniami poszczególnych osi układu współrzędnych są znaki, odpowiednio: *x*, *y* oraz *z*. Dalsze wyjaśnia realizacji tej operacji znajdują się w przykładzie działania programu.

4 Przygotowanie do zajęć (nieobowiązkowe)

Dany jest przykładowy program:

```
void Wymnoz( double Arg1,  double (&Arg2_Tab)[4])
{
  for (double &Mnoznik : Arg2_Tab) Mnoznik = Arg1 * Mnoznik;
}

int main()
{
  double  Arg1 = 2;
  double  Tab[4] = {1, 2, 3, 4};

  Wymnoz(Arg1,Tab);
}
```

Wymnaża ona elementy zawarte w tablicy i wynik składuje ponownie do tej tablicy. Należy zwrócić uwagę, że drugim argumentem jest referencja do tablicy. Dlatego też rozmiar tablicy jest *widoczny* wewnątrz funkcji `Wymnoz`. Tak nie będzie, gdy tablica zostanie przekazana przez wskaźnik. Dzięki temu, że rozmiar tablicy jest *widoczny* wewnątrz funkcji `Wymnoz`, w jej definicji można posłużyć się *pętlą zakresu*.

Należy przerobić funkcję `Wymnoz` na szablon funkcji, tak aby poniższa definicja funkcji `main` poprawnie kompilowała i konsolidowała się.

```
int main()
{
  double  Arg1 = 2;
  double  Tab[4] = {1, 2, 3, 4};

  Wymnoz(Arg1,Tab);

  Macierz2x2  Mac;
  Wektor2D    TabWek[4];

  Wymnoz(Mac,TabWek);
}
```

Dla tych osób, dla których to zadanie okaże się za trudne, a chciałyby otrzymać dodatkową punktację, można zrealizować wariant łatwiejszy (niżej oceniany). Wystarczy przygotować własny dowolny przykład szablonu funkcji, którego użycie zapewni poprawną kompilację i konsolidację programu.

5 Przykład działania programu

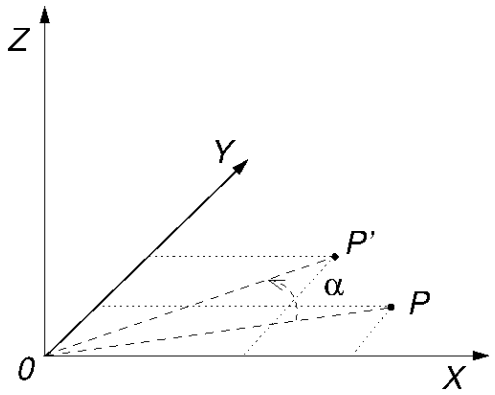
Poniżej przedstawiony przykład wyznacza formę komunikatów i ich forma jest obligatoryjny dla programu tworzonego w ramach niniejszego zadania.

```
panamint> ./program_obroty_3D

:)  Dluzsze przeciwlegle boki sa sobie rowne.
 Dlugosc pierwszego boku: 25.00000000000000000000
   Dlugosc drugiego boku: 25.00000000000000000000
  Dlugosc trzeciego boku: 25.00000000000000000000
  Dlugosc czwartego boku: 25.00000000000000000000

:)  Krotsze przeciwlegle boki sa sobie rowne.
 Dlugosc pierwszego boku: 15.00000000000000000000
   Dlugosc drugiego boku: 15.00000000000000000000
  Dlugosc trzeciego boku: 15.00000000000000000000
  Dlugosc czwartego boku: 15.00000000000000000000

:)  Poprzeczne przeciwlegle boki sa sobie rowne.
 Dlugosc pierwszego boku: 20.00000000000000000000
   Dlugosc drugiego boku: 20.00000000000000000000
  Dlugosc trzeciego boku: 20.00000000000000000000
  Dlugosc czwartego boku: 20.00000000000000000000


 o - obrot bryly o zadana sekwencje katow
 t - powtorzenie poprzedniego obrotu
 r - wyswietlenie macierzy rotacji
 p - przesuniecie prostokata o zadany wektor
 w - wyswietlenie wspolrzednych wierzcholkow
 s - sprawdzenie dlugosci przeciwleglych bokow
 m - wyswietl menu
 k - koniec dzialania programu

 Twoj wybor? (m - menu) > o
 Podaj sekwencje oznaczen osi oraz katy obrotu w stopniach
x 20
y 30
x 30
a 50
:( Bledne oznaczenie osi. Dopuszczalne znaki to: x y z .
:( Sprobuj jeszcze raz.
z 50
.

 Ile razy operacja obrotu ma byc powtorzona?
1


:)  Dluzsze przeciwlegle boki sa sobie rowne.
 Dlugosc pierwszego boku: 25.00000000000000355271
```

```
   Dlugosc drugiego boku: 25.00000000000000355271
  Dlugosc trzeciego boku: 25.00000000000000355271
  Dlugosc czwartego boku: 25.00000000000000355271

:O  Krotsze przeciwlegle boki nie sa sobie rowne!!!
 Dlugosc pierwszego boku: 14.99999999999999822364
   Dlugosc drugiego boku: 14.99999999999999822364
  Dlugosc trzeciego boku: 14.99999999999999467093
  Dlugosc czwartego boku: 14.99999999999999822364

:O  Poprzeczne przeciwlegle boki nie sa sobie rowne!!!
 Dlugosc pierwszego boku: 20.00000000000000710543
   Dlugosc drugiego boku: 20.00000000000000355271
  Dlugosc trzeciego boku: 20.00000000000000355271
  Dlugosc czwartego boku: 20.00000000000000355271


 Twoj wybor? (m - menu) > k

 Koniec dzialania program

panamint> _
```

6 Realizacja rotacji

Niech będzie dany punkt $P = (x, y, z)$. Rozważmy najpierw rotację tego punktu wokół osi OZ o kąt α. Otrzymujemy nowy punkt $P' = (x', y', z')$ tak jak to jest pokazane na rysunku rys. 1. Łatwo zauważyć, że współrzędna z-towa tego punktu nie zmienia się. Transformację tę można



Rysunek 1: Rotacja punktu P o kąt α wokół osi OZ

potraktować jako rozszerzenie wcześniejszej rotacji w układzie dwuwymiarowym, która opisana została w poprzednim zadaniu. Transformację Współrzędnych punktu P do współrzędnych punktu P' realizujemy zgodnie z następującym wzorem:

$$\begin{aligned} x' &= x\cos\alpha - y\sin\alpha, \\ y' &= x\sin\alpha + y\cos\alpha, \\ z' &= z. \end{aligned}$$

Możemy go zapisać w postaci macierzowej

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Tak więc mając na uwadze, że $P = (x,y)$ i $P' = (x',y')$ oraz oznaczając macierz rotacji $\mathbf{R}_{z,\alpha}$, powyższą wzór można zapisać w formie

$$P' = \mathbf{R}_{z,\alpha} \cdot P.$$

W programie należy dokonać odpowiednich przeciążeń operatorów, aby tego typu zapis można było stosować bezpośrednio w programie.

Analogicznie obrót wokół osi *OY* o kąt β możemy zapisać jako

$$\begin{bmatrix} x'' \\ y'' \\ z'' \end{bmatrix} = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}.$$

Zmiana znaku przy funkcji sinus jest związana z przyjętą konwencją miary kąt. Kierunek dodatni miary jest od osi *OZ* do osi *OX*. Przedstawiony wyżej typ przekształcenia dalej zapisywać będziemy jako

$$P'' = \mathbf{R}_{y,\beta} \cdot P.$$

Natomiast macierz obrotu wokół osi *OX* o kąt γ ma postać

$$\begin{bmatrix} x''' \\ y''' \\ z''' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}.$$

Analogicznie do wcześniejszej notacji przekształcenie to zapisywać będziemy jako

$$P''' = \mathbf{R}_{x,\gamma} \cdot P.$$

Załóżmy, że mamy przetransformować pewien punkt P_a poprzez złożenie dwóch obrotów. Tak więc najpierw chcemy dokonać obrotu wokół osi *OZ* o kąt α, a później wokół osi *OX* o kąt γ. Tym samym chcemy wykonać następujące operację

$$P_a^{'} = \mathbf{R}_{z,\alpha} P_a,$$

a następnie

$$P_a^{''} = \mathbf{R}_{x,\gamma} P_a^{'}.$$

Jednak te operacje możemy złożyć i zapisać jako

$$P_a^{''} = \mathbf{R}_{x,\gamma}(\mathbf{R}_{z,\alpha} P_a) = (\mathbf{R}_{x,\gamma}\mathbf{R}_{z,\alpha}) P_a.$$

Tak więc składanie obrotów jest równoważne przemnożeniu odpowiednich macierzy obrotów. Tę cechę należy wykorzystać przy oprogramowaniu pozycji menu

```
o - obrot bryly o zadana sekwencje katow
```

Należy zauważyć że obroty wokół różnych osi nie są przemienne, tzn. w ogólnym przypadku mamy

$$\mathbf{R}_{x,\gamma}\mathbf{R}_{z,\alpha} \neq \mathbf{R}_{z,\alpha}\mathbf{R}_{x,\gamma}.$$

7 Wymagania co do konstrukcji programu

Oprócz wymagań sformułowanych w opisie zadania należy uwzględnić uwarunkowania przedstawione poniżej.

- Należy przekształcić klasy `Wektor2D`, `Macierz2x2` do postaci szablonów `Wektor<>` i `Macierz<>`. Klasy `Wektor3D`, `Macierz3x3`, niezbędne w tym zadaniu, należy zdefiniować jako instancje wspomnianych wcześniej szablonów. W szablonach `Wektor<>`, `Macierz<>` powinny być zdefiniowane analogiczne przeciążenia operatorów jak w zadaniu poprzednim. Należy zwrócić uwagę, że w szablonie `Macierz<>` będzie dodatkowo potrzebna operacja mnożenia macierzy.

  Natomiast na bazie klasy `Prostokąt` należy zdefiniować klasę `Prostopadloscian`. W tym przypadku nie tworzymy szablonu. Definiowane klasy muszą mieć **tylko i wyłącznie niezbędne pola reprezentujące atrybuty danego pojęcia**.

- Należy również zwrócić uwagę, że ze względu na konieczność pamiętania wcześniejszego obrotu, niezbędne jest przechowywanie macierzy obrotu. Dobrze jest więc stworzyć dodatkową strukturę danych, której polami są odpowiednio bryła i macierz obrotu. Struktura ta będzie reprezentowała scenę, na której znajduje się obracana bryła.

- Program musi zachować strukturę modułową i odpowiednią strukturę kartotek. O ile będzie to konieczne, należy zmodyfikować plik `Makefile` (np. gdy dodany zostanie nowy moduł).

- Każda z klas powinna zostać zdefiniowana w oddzielnym pliku nagłówkowym. Metody tej klasy powinny być natomiast definiowane w osobnym module związanym z daną klasą, np. definicja klasy `Prostopadloscian` powinna znaleźć się w pliku nagłówkowym `Prostopadloscian.hh`, zaś metody w pliku `Prostopadloscian.cpp`. Proste metody można definiować bezpośrednio w ciele klasy.

- Dla poszczególnych klas należy przeciążyć niezbędne operatory działające na strumieniach. Nie wszystkie przeciążenia są w tym zadaniu potrzebne. Na pewno będą potrzebne przeciążenia operatorów wczytywania i zapisu dla klasy `Wektor3D` oraz operatora wyświetlania dla klasy `Prostopadloscian`.

- Wszystkie metody, które nie zmieniają stanu obiektu, na którym działają, powinny być metodami typu **const**.

- Program powinien umożliwiać graficzną wizualizację wyników działania. Pozwala na to dołączony w załążku moduł łącza do programu `gnuplot`.

- Wszystkie klasy i metody oraz funkcje powinny zostać opisane. Opis ten powinien być zgodnie z wymogami systemu `doxygen`. Ponadto należy wygenerować dokumentację w formacie HTML za pomocą programu `doxygen`.

Oprócz tego pozostają w mocy wszystkie wcześniejsze wymagania dotyczące struktury katalogów, pliku Makefile, modułowej struktury programu, jak też opisów.

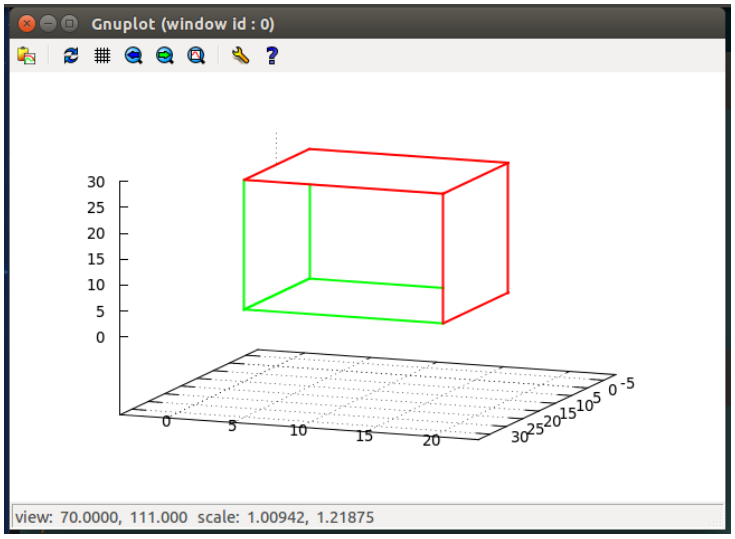
8 Materiały pomocnicze

Zalążek programu znajduje się w katalogu ~bk/edu/kpo/zad/z4. Zawiera on przykład wykorzystania modułu złącza do programu gnuplot, który pozwala zwizualizować wyniki obliczeń. Podstawowe objaśnienia znajdują się w kodzie dostarczonego zalążka.

We wspomnianym katalogu znajduje się również przykład realizacji programu.

9 Zapis danych dla programu gnuplot

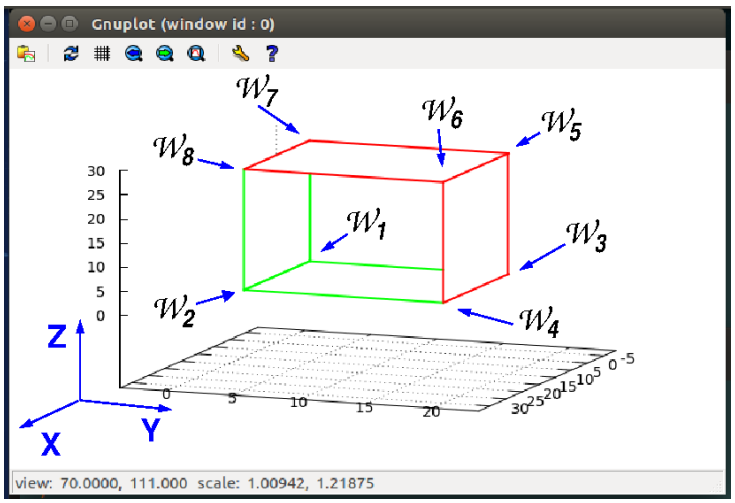
Program gnuplot pozwala na rysowanie powierzchni. Można w ten sposób narysować wszystkie ścianki bryły. Jednak jest to trochę uciążliwe. Dlatego w tym zadaniu proponuje się narysowanie powierzchni prostopadłościanu bez dwóch przeciwległych ścianek (patrz rys. 2). Z tego powodu prostopadłościan przedstawiony na rys. 2 widoczny jest z *prześwitem*. Brzeg tych ścianek, które widoczne są od zewnątrz rysowany jest kolorem czerwonym. Natomiast brzeg ścianek widoczny od wewnątrz rysowany jest kolorem zielonym. Aby otrzymać rysunek w przedstawionej postaci, współrzędne wierzchołków należy zapisać do pliku tekstowego, który będzie przekazany gnuplotowi, w odpowiedni sposób.



Rysunek 2: Rysunek prostopadłościanu w okienku programu gnuplot

Chcąc lepiej przedstawić sposób zapisu na rys. 3 przedstawiony jest rysunek prostopadłościanu z oznaczeniem wszystkich wierzchołków. Schemat zapisu współrzędnych przedstawiony jest poniżej. Uwaga: wolne linie są istotne, należy również zwrócić uwagę na to, że na końcu należy powtórzyć zapis współrzędnych pierwszego i ostatniego wierzchołka.

```
x1  y1  z1      # <-- Współrzędne wierzchołka W1
x2  y2  z2      # <-- Współrzędne wierzchołka W2

x3  y3  z3      # <-- Współrzędne wierzchołka W3
x4  y4  z4      # <-- Współrzędne wierzchołka W4

x5  y5  z5      # <-- Współrzędne wierzchołka W5
x6  y6  z6      # <-- Współrzędne wierzchołka W6

x7  y7  z7      # <-- Współrzędne wierzchołka W7
x8  y8  z8      # <-- Współrzędne wierzchołka W8

x1  y1  z1      # <-- Współrzędne wierzchołka W1
x2  y2  z2      # <-- Współrzędne wierzchołka W2
```



Rysunek 3: Rysunek prostopadłościanu w okienku z zaznaczonymi kolejnymi wierzchołkami

W dalszej część przedstawiony jest zapis faktycznych współrzędnych wierzchołków prostopadłościanu przedstawionego na obu rysunkach.

```
 2.0000000000  3.0000000000  3.0000000000
22.0000000000  3.0000000000  3.0000000000

 2.0000000000 18.0000000000  3.0000000000
22.0000000000 18.0000000000  3.0000000000

 2.0000000000 18.0000000000 28.0000000000
22.0000000000 18.0000000000 28.0000000000

 2.0000000000  3.0000000000 28.0000000000
22.0000000000  3.0000000000 28.0000000000

 2.0000000000  3.0000000000  3.0000000000
22.0000000000  3.0000000000  3.0000000000
```

10 Uproszczenie

Jeżeli zadanie sprawia zbyt duże problemy, to można je zrealizować w uproszczonej formie. W dalszej części przedstawione są proponowane zmiany.

10.1 Pominięcie sekwencji

W uproszczonej wersji przy realizacji opcji:

```
o - obrot bryly o zadana sekwencje katow
```

można przyjąć, że obrót zadawany jest tylko względem jednej wybranej osi. Tak więc użytkownik zamiast wprowadzać sekwencję oznaczeń: znak osi, kąt obrotu; którą kończy kropką, wprowadzi tylko jedną taką parę. Skorzystanie z tego uproszczenia powoduje obniżenie oceny programu o $1,0$ (tzn. jedną ocenę).

10.2 Pominięcie powtórzeń

W kolejnym uproszczeniu można zrezygnować z opcji powtórzeń. Powoduje to jednak, że ocena programu będzie obniżona o $0,5$.

11 Rozszerzenia

Dla osób, które niniejsze zadanie nie sprawi problemu, proponuje się rozszerzenia przedstawione poniżej.

11.1 Zbiór brył

Zamiast pojedynczej bryły, należy stworzyć scenę na której jest co najmniej 5 brył i każdą z nich z osobna można obracać niezależnie od innych.

11.2 Rysowanie bryły

Należy tak zmodyfikować układ punktów zapisywanych w pliku, aby rysowała się pełna bryła, np. tak jak jest to przedstawione na stronie:
http://sequoia.iiar.pwr.wroc.pl/∼kreczmer/kpo/zadania/zad-manipulator/pomoc/zasoby/laczedognuplota-doc/index.html

12 Wymagania i zarys programu zajęć w okresie realizacji zadania

12.1 Tydzień 0

Należy przetestować skalowaność stworzonych dotychczasowych struktur danych takich jak wetor i macierz. Operacja na nich powinny poprawnie działać przy zwiększeniu rozmiaru np. do 5. Jednym z takich testów powinno być ustawienie obrotu dla jednej z maciern np. o 50°, zaś dla drugiej obrotu o -50°. Ich przemnożenie powinno dać macierz jednostkową.

12.2 Tydzień 1

Przed zajęciami muszą zostać stworzone szablony klas `Wektor<>` oraz `Macierz<>`. Ich parametrem powinien być wymiar. Szablony muszą być tak napisane, aby można było je użyć dla dowolnego wymiaru np. 1 lub 1000. Poprawność definicji szablonów powinna być zweryfikowana poprzez ich użycie w poprzedniej wersji programu. Aby to było możliwe klasy `Wektor2D` oraz `Macierz2x2` muszą zostać zdefiniowane jako instancje szablonów `Wektor<>` oraz `Macierz<>`, np.

```
typedef Wektor<2>  Wektor2D;

typedef Macierz<2> Macierz2x2;
```

Pozytywną weryfikacją definicji szablonów będzie uruchomienie programu z zadania nr 3 w wersji z szablonami.

Dokumentacja szablonów powinna być stworzona na bazie wcześniejszej dokumentacji klas `Wektor<>` oraz `Macierz<>`. W większości przypadków nie powinna ona w ogóle wymagać (o ile wcześniej została dobrze napisna) tworzenia żadnego dodatkowego opisu. To co jest konieczne, to jedynie umieszczenie w opisach odpowiednich tagów, które są niezbędne do tworzenia dokumentacji przez program `doxygen`. Generowanie dokumentacji zostanie zaprezentowane na zajęciach.

12.3 Tydzień 2

Rozliczenie się z gotowego programu i rozpoczęcie następnego zadania.