I Kółko informatyczne w V LO w B-B

Jacek Migdał (jacek at migdal.pl)

2006-09-11

1 Streszczenie

Pierwsze zajęcia kółka. Wyjaśnienie co to jest język programowania i kompilowanie. Będzie też parę rad dla kogoś kto rozpoczyna przygodę z informatyką. Gwoździem programu będzie początek kursu C++.

2 Sprawy organizacyjne

2.1 Czas

Kółko będzie się odbywało w każdy czwartek począwszy od 2006-09-22. Planuje że będzie trwało 1h +/- 15minut. Wiem, że kółko informatyczne nakłada się z dodatkowymi zajęciami z matematyki dla klas pierwszych. Jeśli bedą osoby, które chcą uczęszczać na oba kółka poszukamy innego terminu.

2.2 Tematyka

Planuję poświęcić 4 zajęcia na naukę C++. Potem będziemy przerabiać algorytmikę. Będzie też parę tematów-przerywników takich jak rysowanie po ekranie czy napisanie jakiejś prostej gierki. Po zakończeniu Olimpiady Informatycznej zajmiemy się bardziej praktycznymi tematami.

2.3 Jak będą wyglądały zajęcia?

Na początku omawiamy zadania z poprzednich zajęć. Prezentujemy rozwiązania i/lub wskazówki. Potem przerabiamy trochę teorii i/lub wspólnie rozwiązujemy jakiś problem. Na pożegnanie zadaje zadania + ewentualne wskazówki do nich. Nie trzeba notować, do każdych zajęć będzie pdf (w miarę mojego wolnego czasu).

2.4 Ustalenia techniczne

Korzystamy z kompilatora "gcc" i powłoki bash. Edytor tekstu wedle uznania, polecam "vim", a będę pokazywał "xemacs". Linux'owcy zapewne już te programy mają, użytkownicy windows będą korzystać z "cygwin". Oczywiście można używać innych kompilatorów/edytorów. Z swojej strony polecam tandem "linux" + "gcc" + "vim".

2.5 A jak Ja mam to sobie wszystko zainstalować?

Jest płytka zawierający "cygwin.zip". Wystarczy rozpakować ten plik i uruchomić "cygwin.bat". W razie problemów zgłosić się do mnie.

2.6 Cel

Kółko będzie optymalizowane pod przygotowanie do udziału w Olimpiadzie Informatycznej. Poza tym będzie uczyło myśleć "informatycznie".

3 Cytaty

Jest parę sentencji dotyczących programowania godnych przytoczenia:

- Ucz się przez pisanie, nie spamiętywanie.
- Komputer zrobi dokładnie to co mu napiszesz, niekoniecznie to co chcesz.
- Trening czyni mistrza.
- Co mnie nie zabije, to mnie wzmocni.

4 Parę pojeć

Definicje zaczerpnięte z wikipedii.

- **Język programowania** to usystematyzowany sposób przekazywania komputerowi poleceń do wykonania.
- **Kompilacja** to proces automatycznego tłumaczenia kodu napisanego w jednym języku programowania na drugi. Dane wejściowe najczęściej nazywa się kodem źródłowym. Program wykonujący tłumaczenie to kompilator.
- **Kod źródłowy** to program komputerowy w postaci takiej, jaką tworzy ją człowiek w pewnym języku programowania, zazwyczaj jako tekst, ale też jako dane dla translatora, przeznaczony do analizowania i modyfikacji przez człowieka.
- **Kod maszynowy** to język programowania, w którym zapis programu wymaga instrukcji bezpośrednio jako liczb, które są rozkazami i danymi bezpośrednio pobieranymi przez procesor wykonujący ten program.
- **Biblioteka** w informatyce to zbiór klas, funkcji (i ew. innych konstrukcji programistycznych), z których korzystają inne programy.

5 C++ pierwszy program

Niepisanym zwyczajem jest zaczynanie nauki programowania od napisania programu wyświetlającego "Hello World!". Oto on:

Listing 1: Hello World

```
1 #include <cstdio>
2 using namespace std;
3
4 int main()
5 {
6         printf("Hello World!\n");
7         return 0;
8 }
```

Tworzymy nowy katalog("`mkdir helloworld`"), wchodzimy do niego ("`cd helloworld`") i uruchamiamy edytor tekstu "xemacs" ("`xemacs hello.cpp &`"). Następnie przepisujemy program, i zapisujemy zmiany. W terminalu kompilujemy program ("`g++ -o hello hello.cpp`") i uruchamiamy ("`./hello.exe`").

Teraz demistyfikujemy działanie programu:

```
1 #include <cstdio>
```

Załączamy bibliotekę do obsługi standardowego wejścia/wyjścia. Ponieważ korzystamy z biblioteki z języka C nazwę poprzedzamy literą "c".

```
2 using namespace std;
```

Używamy standartowego zakresu nazw, w tym programie ta instrukcja jest zbędna. Została umieszczona, żeby wyrobić nawyk jej pisania.

```
4 int main()
```

Rozpoczynamy główny program, "int" to nazwa typu zmiennej jaki będziemy zwracali. Choć teoretycznie można zamiast tego napisać:

```
4 void main()
```

lub nawet

```
4 main()
```

to obie powyższe instrukcje są niepoprawne. Zgodnie ze standardami konieczne jest zwrócenie rezultatu działania programu.

```
5 {
8 }
```

Klamrami zaznaczamy części kodu, w tym przypadku kodu należącego do main.

```
6         printf("Hello World!\n");
```

Wypisujemy na standardowe wyjście napis "Hello World!", "\n" to znak sterujący, tworzy nową linię. "printf" jest funkcją z biblioteki "cstdio". Niektórzy napisaliby tą instrukcję tak (cout << "Hello World!\n";) korzystając z biblioteki "iostream". Choć ta metoda jest odrobinę wygodniejsza, jest 2-4 razy wolniejsza co czyni ją niepraktyczną w zadaniach z Olimpiady Informatycznej.

```
7         return 0;
```

Kończymy program, zwracając, że zakończył się sukcesem. Inne liczby niż "0" informują system operacyjny i użytkownika o błędzie. Np. odwołanie się do nie swojej pamięci.

6 Definicja, deklaracja i inicjalizajca

Bardzo ważne pojęcia w informatyce, w szczególności C++. Mogą dotyczyć klas, funkcji i zmiennych. Omówimy na przykładzie kota.

- **Definicja** – Jeśli spotkasz coś o nazwie mruczek to wiedz, że będzie to kot.
- **Deklaracja** – Niech będzie kot o nazwie mruczek.
- **Inicjalizacja** – Niech mruczek ma czarną sierść i określone cechy.

Łatwo zauważyć że deklaracja jest jednocześnie definicją, ale nie odwrotnie. Inicjalizacji możemy dokonać po deklaracji. Inicjalizujemu zmienne tylko za pierwszym razem.

7 Składnia C++

Czyli sposób zapisu programów. Nie przejmuj się jeśli czegoś nie będziesz rozumiał. To są podstawowe zasady, których znaczenie będzie widać dopiero w użyciu.

- **Komentarze** coś co widzi tylko człowiek

```
1  //to jest komentarz
2  int a = /*a to tez*/ 5;
3  /*
4  A to jest komentarz wielolinijkowy
5  */
6  /*Jeden komntarz /* BLAD - nie mozna zagniezdzac komentarzy  */ */
```

- **Polecenia preprocesora** coś co rządzi się swoimi prawami, służy m.in. do kompilacji warunkowej i załączania bibliotek. Każda linia do preprocesora zaczyna się od #. Każda instrukcja preprocesora kończy się znakiem nowej linii. Jest to wyjątek.

```
1  #include <cstdio>
2  #include "moj_lokalny_plik.h"
3  #define JEDEN 1
4  #ifdef JEDEN
5  int a = 1; // to jest co innego
6  #else
7  int a = 0; // i to tez
8  #endif
```

- **Wrażliwość na duże-małe litery** Wielkość znaków ma znaczenie.

```
1  int a = 0;
2  A = 1; // blad, zadeklarowalismy inna zmienna
```

- **Niewrażliwość na białe znaki** Czyli, poza wyjątkami, więcej niż jedna spacja i enter nie robi różnicy

```
1  int main(){printf("Hello world!\n");return 0;}
2  int main()
3  {
4          printf
5          ("Hello world!")
6          ;
7          return
8          0
9          ;
10 } // to znaczy dla komputera to samo co w pierwszej linijce
```

Tu macie władzę, możecie pisać jak chcecie. Mimo wszystko doradzam umiar, bo przegięcia w każdą stronę są niezdrowe (i uciążliwe dla siebie i środowiska).

- **Fragmenty kodu zaznaczamy { i }** - dotyczy to pętli, warunków, deklaracji funkcji i definicji klas. W przypadku pętli i warunków po których jest co najwyżej jedna instrukcja jest to opcjonalne.

```
1  for( int i = 1 ; i<10 ; i++ ) //abrakadabra
2  {
3          a *= i; // hocus pocus
4  }
5
6  {
7          // Ot tak sobie zaznaczony kod
8  }
9
10 for( int i = 1 ; i<10 ; i++ )
11         a *= i; // takie samo zaklecie
```

- **Każda instrukcję kończymy ";"** poza deklaracjami funkcji, poleceniami preprocesora, instrukcjami warunkowymi i pętlami (* wyjątek od wyjątku, nie dotyczy pętli i warunków pustych).

```
1         int a;
2         int b; int c;
3         int mojaUkochanaFunkcja( int argument ); // kosmos
```

8 Ja chce juz coś napisać!

Spokojnie, teraz przedstawię parę potrzebnych rzeczy niezbędnych do napisanie programu. Sygnalizuję tu pewne zagadnienia, które omówię później dokładnie.

- Deklaracja zmiennych stało-przecinkowych (całkowitych):

  TYP NAZWA; lub TYP ZMIENNA1, ZMIENNA2, ... ;

  int to typ 32 bitowy stałoprzecinkowy.

```
1         int a; // zmienna globalna,
2         // mozna ja uzywac wszedzie ponizej od definicji
3         // na poczatku ma wartosc 0
4
5         int main()
6         {
7                 int b; // zmienna lokalna,
8                 //ma wartosc smiec, dowolna
9                 //mozna ja uzywac ponizej deklaracji
10                //w obrebie zaznaczonego kodu
11                int c, d; //dwie zmienne za jednym zamachem
12                int e = 1, f = 2; //definicja i inicjalizacja w jednym
13                int return; //BLAD nazwa nie moze byc slowem kluczowym
14                int 11e; //BLAD pierwszy znak musi byc litera lub "_"
15                return 0;
16        }
```

- Wczytywanie i wypisywanie zmiennych

```
1  #include <cstdio>
2  using namespace std;
3
4  int main()
5  {
6         int a;
7         scanf("%d", &a); //wczytywanie zmiennej a, zwrocic uwage na &
8         printf("A oto zmienna %d\n", 2*a); // wypisywanie zmiennej
9         return 0;
10 }
```

- **Wykonywanie działań** działają podstawowe operatory + - * /

```
1  #include <cstdio>
2  using namespace std;
3
4  int main()
5  {
6         int a, b;
7         scanf("%d %d", &a, &b); //mozna wczytywac 2 za jednym zamachem
8         int suma = a*b;
9         int roznica = a-b;
```

```
10          int iloczyn = a*b;
11          int iloraz = i/b;
12          printf("Suma %d + %d = %d\n", a, b, suma);
13          printf("Suma %d - %d = %d\n", a, b, roznica);
14          printf("Suma %d * %d = %d\n", a, b, iloczyn);
15          printf("Suma %d / %d = %d\n", a, b, iloraz);
16          // sproboj wpisac 3 i 2, dlaczego ostatni wynik to 1?
17          return 0;
18  }
```

9 Sterowanie kodem programu

9.1 Warunki

Porównywanie liczb za pomocą operatorów "<" ">" "<=" ">=" "==". "==" to co innego niż "=". Ten pierwszy porównuje, drugi przypisuje.

9.2 Instrukcje warunkowe

Pierwszą z nich jest **if**. Po prostu warunek jeśli. Można stosować jeden po drugim:

```
1  {       // gdzies wczesniej jest zdeklarowana zmienna int a
2          if( a<2 )
3          {
4                  printf("Zmienna a jest mniejsza niz dwa\n");
5                  // cos tam robie
6          }
7          else if( a>4 )
8          {
9                  printf("Zmienna a jest wieksza od 4\n");
10                 // cos tam robie
11         }
12         else
13         {
14                 //zmienna a nalezy do przedzialu <2, 4>
15         }
16 }
```

Podobnie działa instrukcja **switch**.

```
1  {       // gdzies wczesniej jest zdeklarowana zmienna int a
2          switch( a )
3          {
4                  case 1:
5                  // a ma wartosc 1
6                  break; // konczymy zabawe
7
8                  case 2:
9                  case 3:
10                 // a ma wartosc 2 lub 3
11                 break;
12
13                 default:
```

```
14                      //a ma wartosc inna niz 1, 2 i 3
15                      break;
16              }
17      }
```

Obie metody działają tak samo szybko, częściej stosuję się **if**.

9.3 Pętle

Proszę się nie bać. Nikogo nie zamierzam wieszać... na razie.

Typu **while**:

```
1               while( a>2 ) // dopoki a jest wieksze od 2
2               {
3                       // wykonaj cos
4                       a--; // i zmniejsz a o jeden
5               }
```

Typu **do while**:

```
1               do //ta petla wykona sie conajmiej raz
2               {
3                       // wykonaj cos
4                       a--; // i zmniejsz a o jeden
5               }
6               while( a>2 ) // dopoki a jest wieksze od 2
```

Typu **for**, najpotężniejsza i najbardziej przydatna, jest podobna do pętli **while**. Składnia:

```
1       for( instrukcja_poczatkowa ; warunek ; instrukcja_kroku )
2       {
3               // cos sie tu wykonuje
4       }
```

instrukcja_poczatkowa - wykona się zawsze na początku, tylko jeden raz

warunek - musi być spełniony za każdym razem zanim sie wykonają instrukcje pomiędzy { }

instrukcja_kroku - wykona sie po każdym przebiegu funkcji

Przykład:

```
1       {
2               for( int i = 0 ; i<10 ; i=i+1 )
3               {
4                       printf("%d ", i);
5               }
6               printf("\n");
7               // na ekranie wyswietli sie 0 1 2 3 4 5 6 7 8 9
8       }
```

9.4 Instrukcje break i continue

Służą do kierowanie pętlą z jej wnętrza.

break powoduje zakończenie ostatniej pętli, jest też używana w instrukcji warunkowej **switch**.

```
1       {
2               for( int i = 0 ; i<10 ; i=i+1 )
3               {
4                       if( i==6 )
5                               break;
```

```
6                       printf("%d ", i);
7               }
8               // to da wyjscie 0 1 2 3 4 5
9       }
```

continue powoduje przejście do końca pętli, ale nie jej zakończenie

```
1       {
2               for( int i = 0 ; i<10 ; i=i+1 )
3               {
4                       if( i==6 )
5                               continue;
6                       printf("%d ", i);
7               }
8               // to da wyjscie 0 1 2 3 4 5 7 8 9
9       }
```

10 Zadania

Czyli to co wszystkie dzieci lubią najbardziej.

10.1 Liczby

Zadanie: Zamień liczbę z postaci cyfrowej na zapis naturalny (bez polskich liter).

Wejście: (n) liczba całkowita z przedziału <1, 100>liczba zamian do wykonania. Kolejne n wierszy zawiera liczbę do zamiany.

Wyjście: n słów opisujących liczby

Przykładowe wejście:

3

11

0

3

Przykładowe wyjście:

trzy

zero

jedenascie

10.2 Tabliczka mnożenia

Zadanie: Mały Jaś chce mieć tabliczkę mnożenia. Chce być oryginalny dlatego codziennie prosi Mamę o jakaś o x kolumnach i y wierszach. Pomóż Mamie i napisz program, który będzie generował tabliczki.

Wejście: dwie liczby całkowite 0¡x, y¡=20

Wyjście: y+1 wierszy, w każdym x+1 cyfr. W pierwszym wierszu liczby od 0 do x. W kolejnych numer wiersza i x liczb całkowitych będących wynikiem mnożenia numeru wiersza i kolumny w której dana liczba się znajduję. Wiersze i kolumny liczymy od zera.

Przykładowe wejście: 4 3

Przykładowe wyjście: 0 1 2 3 4 1 1 2 3 4 2 2 4 6 8 3 3 6 9 12

10.3 Program z błędem

Co jest nie tak z tym programem?

```
1  #include <cstdio>
2  using namespace std;
3
4  int main()
5  {
6          int a, b;
7          scanf("%d %d", &a, &b);
8          if( a = b )
9                  printf("a jest rowne b\n");
10         else
11                 printf("a jest rozne od b\n");
12         printf("Na poczatku podales a=%d i b=%d\n", a, b);
13         return 0;
14 }
```

Dla dociekliwych: podaj takie wejście dla których zostanie wyświetlone "a jest rozne od b".

11 Linki

- http:\\www.oi.edu.pl\ Strona Olimpiady Informatycznej.
- http:\\www.tocoder.com\ Taki konkurs informatyczny.
- http:\\ag.bocznica.org\ Sprawdź tu zanim zadasz mi jakieś pytanie.
- http:\\www.intercon.pl\šektor\cbx\ W miarę dobry kurs C++, na OI musimy jednak używać "printf" "scanf" zamiast "cin" i "cout".