J.NAWROCKI, M. ANTCZAK, H. ĆWIEK, W. FROHMBERG, A. HOFFA, M. KIERZYNKA, S. WĄSIK

PROGRAMOWANIE IMPERATYWNE

JĘZYK C

ZAD. 1. Należy napisać w języku *C*, skompilować i wstępnie przetestować następujące programy:

a) Program generujący napis *„Bitwa pod Akcjum”*. Poeksperymentuj z tym programem i zobacz, jak będzie reagował kompilator, gdy takich znaków jak cudzysłów, średnik, nawias, czy *„\n”* będzie brakować lub gdy będą w nadmiarze w stosunku do wersji ze slajdu nr 9.

b) Program generujący sumę liczb 18 i 2.

c) Program generujący sumę dowolnych dwóch liczb całkowitych.

d) Program generujący liczbę równą minimum z dwóch liczb całkowitych.

e) Program generujący wartość bezwzględną podanej liczby całkowitej.

f) Program obliczający liczbę cyfr dziesiętnych podanej liczby naturalnej *(0, 1, 2, ...)*.

Program F napisz metodą *“przyrostową”*, tzn. zacznij od wprowadzenia i skompilowania pnia programu (ang. *stub*) w wersji ze slajdu wykładowego nr 145 i dokładaj kolejne fragmenty pokazane na slajdach nr 148, 150, 152.

ZAD. 2. Symboliczne wykonanie algorytmu określonego schematem blokowym polega na przechodzeniu przez bloki tego schematu i obserwowaniu oddziaływania instrukcji na wartości zmiennych (slajdy wykładowe od 105 do 142). Symboliczne wykonanie algorytmu zapisanego w języku C, polega na:

1. *„Instrumentalizacji”* programu, czyli wstawieniu między oryginalne instrukcje programu dodatkowych instrukcji *printf(...),* które pozwalają obserwować wartości wybranych zmiennych.

2. Wykonaniu tak przygotowanego programu.

Jest to bardzo prosta forma tzw. *debuggingu*.

Załóżmy, że dany jest program:

Nr linii	Instrukcja języka C
1	`int main() {`
2	`int X;`
3	`Scanf("%d", &X);`
4	`If ( X < 0 ) {`
5	`X = -X;`

(*) gwiazdką oznaczone są zadania, które nie są realizowane na ćwiczeniach i są przeznaczone do wykonania jako zadania domowe.

6	}
7	printf("%d \n", X);
8	return;
9	}

Oto jego *„zinstrumentalizowana”* postać pozwalająca obserwować wartość zmiennej X (instrumentalizujące fragmenty kodu zaznaczono kolorem czerwonym):

```
int main() {
          int   X;
          printf("--- 2---: X=%d \n", X);
          scanf("%d", &X);
          printf("--- 3---: X=%d \n", X);
          if ( X < 0 ) {
               X = -X;
               printf("--- 5---: X=%d \n", X);
          }
          printf("%d \n", X);
          return;
}
```

Należy utworzyć zinstrumentalizowaną postać programu z zadania 1F i na tej podstawie (poprzez wykonanie zinstrumentalizowanego programu dla wybranych danych) zweryfikować poprawne działanie programu.

ZAD. 3. Liczba automorficzna to taka, która znajduje się na końcu swojego kwadratu. Na przykład *5, 6, 25* są liczbami automorficznymi, bo $5^2 = 25$, $6^2 = 36$, $25^2 = 625$. Napisz program, który sprawdza, czy dana liczba *N* jest liczbą automorficzną. Zastosuj podejście przyrostowe (patrz komentarz na końcu zadania 1.). Dokonaj instrumentalizacji kodu pozwalającej obserwować wartości zmiennych występujących w programie.

Uwaga 1. W języku C wartością wyrażenia ***a/b***, gdzie *a i b* są typu całkowitego, jest część całkowita z dzielenia, a wartością wyrażenia ***a%b*** jest reszta z dzielenia.

Uwaga 2. Przy realizacji tego zadania dobrym pomysłem może być zaprogramowanie sobie pomocniczej funkcji *Rzad*, która oblicza rząd liczby zdefiniowany następująco:

Rzad(n) jest to najmniejsza liczba 10^k taka, że $n < 10^k$.

Na przykład *Rzad(1) = Rzad (9) = 10, Rzad(10) = Rzad(99) = 100* itd.

ZAD. 4. Na podstawie rozwiązania zadania 3. napisz program, który generuje wszystkie liczby automorficzne w podanym przedziale *A, B*. Napisz i wykorzystaj funkcję *Automorf(n)*, która sprawdza, czy *n* jest liczbą automorficzną.

(*) gwiazdką oznaczone są zadania, które nie są realizowane na ćwiczeniach i są przeznaczone do wykonania jako zadania domowe.

We wszystkich poniższych zadaniach domowych:

1. **Opracuj zbiór przypadków testowych, które doprecyzują interfejs użytkownika.**
2. **Narysuj schemat blokowy dla proponowanego rozwiązania.**
3. **Przelicz (metodą ręcznej symulacji, która była prezentowana na wykładzie) wybrane przypadki testowe na schemacie, by sprawdzić jego poprawność.**
4. **Napisz program w C.**
5. **Przelicz na nim wybrane przypadki testowe metodą ręcznej symulacji.**

ZAD. 5(*). Napisz program czytania trzech liczb naturalnych i drukowania ich **średniej arytmetycznej**.

ZAD. 6(*). Napisz program czytania trzech liczb i sprawdzania, czy mogą być one **długościami boków jakiegoś trójkąta**. **Wskazówka:** $|b-c|<a<b+c$.

ZAD. 7(*). Napisz program obliczania wartości **funkcji n!** (silnia) dla *n* będących liczbami naturalnymi (0, 1, 2, .. 8).

ZAD. 8(*). Napisz program obliczania **sumy cyfr** dziesiętnych podanej liczby naturalnej n (np.: suma cyfr liczby 0 jest równa 0, a suma cyfr dziesiętnych liczby 99 jest równa 18).

ZAD. 9(*). Napisz program obliczania wartość **funkcji potęgowej** a^b dla *a, b* będących liczbami naturalnymi (0,1, 2,..).

ZAD. 10(*). Napisz program czytania skończonego ciągu liczb całkowitych i drukowania tego ciągu w **odwrotnej kolejności** (np.: 10 liczb).

ZAD. 11(*). Napisz program znajdowania **najmniejszej liczby** w skończonym ciągu liczb naturalnych (np.: 10 liczb).

ZAD. 12(*). Napisz program obliczania **sumy liczb** podanego, skończonego ciągu liczb całkowitych ((*)rzeczywistych) (np.: 10 liczb).

ZAD. 13(*). Napisz program obliczania **wartości średniej** podanego, skończonego ciągu liczb naturalnych (rzeczywistych) (np.: 10 liczb). Należy zmodyfikować odpowiednio rozwiązanie, które zostało zaproponowane dla zadania 11.

ZAD. 14(*). Napisz program czytania rozmiaru *n* (skończona liczba naturalna) i rysowania macierzy o rozmiarach *n* x *n* typu *lewa-dolna*. Macierz *lewa-dolna* zawiera na głównej przekątnej i pod tą przekątną znaki *X*, natomiast powyżej głównej przekątnej są znaki spacji. Oto przykład macierzy lewa-dolna dla *n=4*:

```
X
XX
XXX
XXXX
```

ZAD. 15(*). Napisz program czytania rozmiaru *n* i rysowania macierzy o rozmiarach *n* x *n* typu *prawa-górna*. Macierz *prawa-górna* zawiera na głównej przekątnej i nad tą przekątną znaki *X*, natomiast poniżej głównej przekątnej są znaki spacji. Oto przykład takiej macierzy dla *n=4*:

(*) gwiazdką oznaczone są zadania, które nie są realizowane na ćwiczeniach i są przeznaczone do wykonania jako zadania domowe.

XXXX
 XXX
  XX
   X

ZAD. 16(*). Napisz program rozwiązywania **równania kwadratowego** $ax^2 + bx + c = 0$. Uwzględnij wszystkie możliwe przypadki wartości współczynników *a*, *b*, *c*.

ZAD. 17(*). Napisz program sprawdzania, czy podana liczba naturalna jest **liczbą pierwszą**. **Wskazówka:** wykorzystaj metodę sita Eratostenesa.

ZAD. 18(*). Napisz program wyszukiwania **wszystkich liczb pierwszych** w podanym przedziale <A, B>. Zastosuj funkcję sprawdzającą, czy dana liczba jest liczbą pierwszą z poprzedniego zadania.

ZAD. 19(*). Napisz program obliczania **największego wspólnego podzielnika** (NWP) dwóch liczb naturalnych.

ZAD. 20(*). Napisz program obliczania **NWP dla skończonego zbioru liczb** naturalnych (np.: 10 liczb).

ZAD. 21(*). Napisz program obliczania **najmniejszej wspólnej wielokrotności** (NWW) dwóch liczb naturalnych.

ZAD. 22(*). Napisz program obliczania **NWW dla skończonego zbioru liczb** naturalnych (np.: 10 liczb).

ZAD. 23(*). Napisz program wyszukiwania w podanym przedziale *<D, G>* wszystkich liczb naturalnych *a*, *b*, dla których istnieje liczba naturalna *c* spełniająca warunek $\boldsymbol{a^2 + b^2 = c^2}$ (np.: $3^2 + 4^2 = 5^2$).

ZAD. 24(*). Napisz program wyznaczania *n*-tego wyrazu **ciągu Fibonacciego**.

ZAD. 25(*). Napisz program sortujący ciąg liczb naturalnych metodą przez **wymianę/wybór** (selectionsort).

ZAD. 26(*). Napisz program **scalania ciągów**, który polega na łączeniu kilku (co najmniej dwóch) posortowanych ciągów w jeden ciąg posortowany.

ZAD. 27(*). Napisz program **mnożenia macierzy kwadratowych**.

ZAD. 28(*). Napisz program sprawdzający **współliniowość trzech punktów**.

ZADANIA NIEOBOWIĄZKOWE DLA ZAAWANSOWANYCH

ZAD. 29. Programy, które na ekran wypisują dokładną kopię swojego własnego kodu, to tzw. **„quine programs”**, nazwane od nazwiska logika Willarda van Orman Quine’a. Przeanalizuj znajdujące się poniżej quine’y i postaraj zrozumieć jak działają. A Ty umiesz napisać jakiś inny samodzielnie?

Uwaga – w poniższym programie druga i trzecia linijka powinny być wpisane w jednej linii.

```
#include<stdio.h>
char *program="#include<stdio.h>%cchar *program=%c%s%c;%cvoid
main()%c{%cprintf(program,10,34,program,34,10, 10,10,10);%c}";

void main()
```

(*) gwiazdką oznaczone są zadania, które nie są realizowane na ćwiczeniach i są przeznaczone do wykonania jako zadania domowe.

```
{
printf(program,10,34,program,34,10,10,10,10);
}
```

A tu trochę krótsza wersja:

```
p="p=%c%s%c;main(){printf(p,34,p,34);}";main(){printf(p,34,p,34);}
```

ZAD. 30. Napisz **jak najkrótszy** program, który scala trzy posortowane ciągi o długości **n** w posortowany ciąg o długości **3*n**. Program powinien wczytać z wejścia liczbę **n**, a następnie trzy ciągi wejściowe. Wynik powinien być wypisany na ekran. Program powinien działać w **czasie liniowym**, to znaczy przeglądać każdy ciąg tylko raz. Długość kodu powinna być zmierzona **liczbą bajtów** użytą do zapisania pliku (podawaną we właściwościach pliku).

Przykładowe wejście:

```
4
1 4 7 8
3 3 5 10
2 7 9 11
```

Oczekiwane wyjście:

```
1 2 3 3 4 5 7 7 8 9 11
```

Podpowiedź: Twój program w czasie kompilacji może generować ostrzeżenia. Lepiej zastosuj język C niż C++, bo jest on zdecydowanie mniej restrykcyjny.

(*) gwiazdką oznaczone są zadania, które nie są realizowane na ćwiczeniach i są przeznaczone do wykonania jako zadania domowe.