

1. Mój pierwszy program

1.1. Cel zajęć

Naszym celem¹ jest nauczenie się/was:

1. Programowania
2. Programowania w języku C

W pierwszej kolejności należałoby odpowiedzieć na pytanie „Po co!”? Krótka odpowiedź na tak zadane pytanie jest następująca:

— *Bo tak!*

Dłuższa odpowiedź będzie jakoś taka:

— *Rada Wydziału Mechanicznego (Studium Mechatroniki) w swej mądrości zdecydowała, że każdy absolwent AiRu/Mechatroniki, kończący studia I stopnia powinien znać co najmniej jeden język programowania. Jako język podstawowy uznano język C.*

Kolejne, naturalne pytanie będzie jakoś takie:

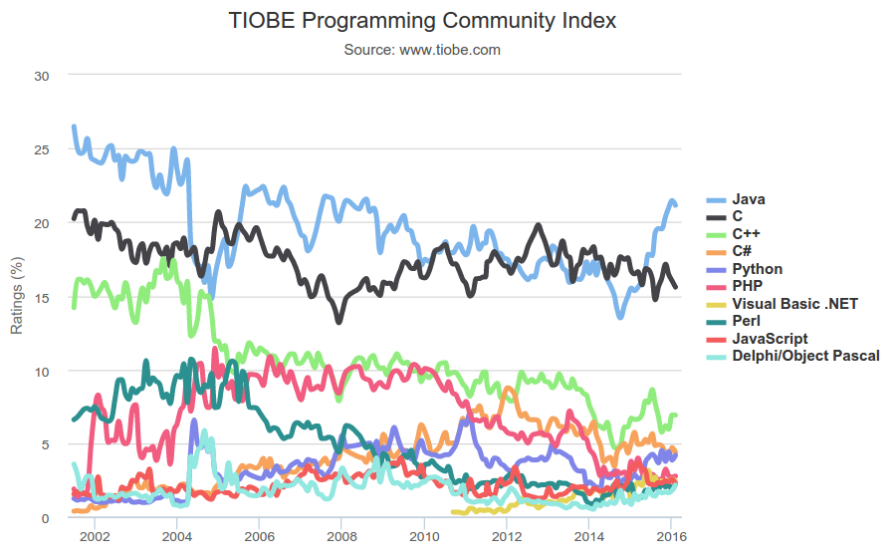
Czemu język C?

Tu odpowiedź jest znacznie bardziej skomplikowana. Odpowiedzi można szukać bądź w historii języków programowania lub w różnych portalach dla programistów. Ja wybrałem TIOBE Company²

Na rysunku 1.1 przedstawiona jest „popularność” różnych języków programowania. Język C od wielu lat znajduje się w czołówce (aktualnie na drugim miejscu).

¹ No dobra. Ja nie wiem co jest Państwa celem. Być może macie, po prostu, sporo czasu. . .

² TIOBE is specialized in assessing and tracking the quality of software. We measure the quality of a software system by applying widely accepted coding standards to it.



Rysunek 1.1. Popularność różnych języków programowania według TIOBE

1.2. Programowanie

Co to jest programowanie? Nie potrafię znaleźć dobrej, oficjalnej, definicji tego słowa (czy procesu). Pozwalam sobie zatem zacytować za (angielską) Wikipedią:

The purpose of programming

is to find a sequence of instructions that will automate performing a specific task or solving a given problem. The process of programming thus often requires expertise in many different subjects, including knowledge of the application domain, specialized algorithms and formal logic.

Co na nasze przetłumaczyć można jakoś tak: *Celem programowania jest znalezienie sekwencji poleceń pozwalających zautomatyzowanie wykonanie zadania lub rozwiązanie zadanego problemu. Dlatego też proces programowania wymaga wiedzy z zakresu wielu dziedzin, w tym dobrej*

znajomości problemu, różnych specjalizowanych algorytmów i logiki formalnej.

Warto też popatrzeć na motto tego bryku, sformułowane przez Lesliego Lamporta:

Programming

People seem to equate programming with coding, and that's a problem. Before you code, you should understand what you're doing. If you don't write down what you're doing, you don't know whether you understand it, and you probably don't if the first thing you write down is code. If you're trying to build a bridge or house without a blueprint—what we call a specification—it's not going to be very pretty or reliable. That's how most code is written. Every time you've cursed your computer, you're cursing someone who wrote a program without thinking about it in advance.

Co na nasze tłumaczy się jakoś tak: *Ludzie zwykli utożsamiać programowanie z kodowaniem i to jest problem. Zanim zaczniesz kodować powinieneś rozumieć co robisz. Jeżeli nie zapiszesz sobie co zamierzasz osiągnąć, nie wiesz nawet czy to rozumiesz i, najprawdopodobniej, nie rozumiesz, jeżeli zaczynasz od pisania kodu. Jeżeli rozpoczniesz budowę mostu albo domu bez szczegółowego planu (tu nazywamy to specyfikacją) to nie ma szans żeby był on jakoś specjalnie ładny czy solidny. Za każdym razem kiedy klniesz na komputer — przeklinasz kogoś, kto napisał program bez wcześniejszego przemyślenia co on ma robić.*

Można powiedzieć, że programowanie przypomina trochę budowanie czegoś z klocków. Jak się chce osiągnąć oszałamiający efekt — trzeba mieć dokładny plan. W przeciwnym razie powstające dzieło trzeba wielokrotnie przerabiać i wielokrotnie burzyć to co już osiągnięto, żeby poprawić jeden klocek, zmienić jego kolor, usytuowanie czy orientację.

1.3. Na czym polega programowanie?

— Mamy do rozwiązania problem

- Decydujemy, że użyjemy komputera Zastanawiamy się, jak komputer może nam pomóc
 - jest jakaś gotowa „aplikacja” która się nada? jeżeli tak — użyjemy jej
 - jeżeli nie — musimy ją stworzyć.
- ??? Jak się do tego zabrać?

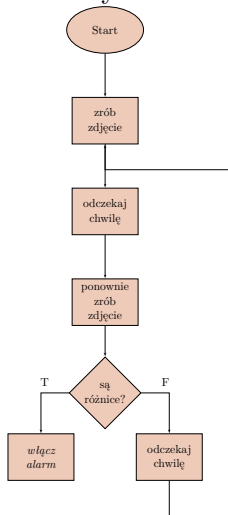
1.3.1. Wykrywanie intruza

Weźmy jakiś prosty przykład. Na początek założmy, że mamy komputer z kamerą i chcemy użyć go do nadzoru jakiegoś pomieszczenia. Trzeba obmyślić jakiś sposób³ postępowania.

Przyjąć można, że kamera co jakiś czas wykonuje fotografię nadzorowanego obszaru i porównuje ją z poprzednim zdjęciem. Jeżeli zauważy jakieś różnice — zgłasza alarm. Jeżeli nie — kontynuuje pracę.

Zaproponowany algorytm jest bardzo naiwny. Jako zadanie domowe mogę polecić zastanowienie się nad jakimś lepszym, uwzględniającym zmiany oświetlenia, cienie i inne mogące pojawić się problemy.

Zacznijmy od schematu blokowego.



³ Ten „sposób“ to będzie nasz algorytm. Zakładam, że to pojęcie nie jest Państwu obce.

Kwestie do rozstrzygnięcia:

1. Jak robić fotografię?
2. Jak „odczekać chwilę” i ile to „chwila”?
3. Jak stwierdzić czy są różnice?
4. Co zrobić po zasygnalizowaniu alarmu?

1.3.2. Największy wspólny dzielnik

Kolejny problem: Mamy znaleźć największy wspólny dzielnik dwu dodatnich liczb całkowitych m i n .

Z definicji sposób postępowania może być następujący:

1. Znaleźć wszystkie dzielniki liczby m ; tworzą one zbiór M .
2. Znaleźć wszystkie dzielniki liczby n ; tworzą one zbiór N .
3. Znaleźć część wspólną $A = M \cap N$
4. Największym wspólnym dzielnikiem (nwd) będzie największy (co do wartości) element zbioru A ; czyli $nwd = \max(A)$.

Dla każdego z tych kroków opracować trzeba komputerowy sposób postępowania. Dobra wiadomość jest taka, że punkt pierwszy i drugi realizowane będą identyczny sposób, ale na różnych danych.

Przygotowanie odpowiednich algorytmów (w postaci schematów blokowych) pozostawiam zainteresowanym studentom.

1.3.3. Algorytm Euklidesa

Oprócz działania „z definicji” można zastosować znany od wieków *Algorytm Euklidesa*. Wygląda on jakoś tak:

- E1** [Znajdowanie reszty] Podziel m przez n i niech r oznacza resztę z tego dzielenia.
- E2** [Czy wyszło zero?] Jeśli $r = 0$, zakończ algorytm; odpowiedzią jest n .
- E3** [Upraszczenie] Wykonaj $m \leftarrow n$, $n \leftarrow r$ i wróć do kroku E1.

Odlóżmy teraz na bok zastanawianie się „Czemu to działa?” oraz „Jak to działa?” Przyjmijmy (na wiarę), że działa. I chodzi o realiza-

cję tego algorytmu. Możemy ewentualnie zastosować wersję algorytmu „z odejmowaniem”:

Algorytm Euklidesa — wersja z odejmowaniem

1. Jeżeli m **jest równe** n — koniec, największym wspólnym dzielnikiem jest n .
2. Jeżeli $m > n$ przyjmij $m \leftarrow m - n$; w przeciwnym razie przyjmij $n \leftarrow n - m$
3. Przejdź do kroku 1.

Cechą wspólną obu algorytmów jest to, że powtarzane są „w kółko”. W jaki sposób zrealizować ten algorytm?

1. Za pomocą ołówka i kartki

m	n	różnica
200	15	185
185	15	170
...
20	15	5
5	15	10
5	10	5
5	5	0

2. Za pomocą komputera

Realizacja komputerowa może sprawiać pewne kłopoty. Same posiadanie komputera (tableta, telefonu komórkowego) to zapewne za mało. Można oczywiście wykorzystać gotową aplikację (kalkulator) żeby wesprzeć obliczenia prowadzone na kartce, ale...

1. Ołówek + kartka + kalkulator
2. Arkusz kalkulacyjny Konstrukcję takiego arkusza nie jest łatwo opisać. Stąd zawartość poszczególnych komórek:

	A	B	C
1	m	n	różnica
2	200	15	=A2-B2
3	=JEŻELI(C2>0;C2;A2)	=JEŻELI(C2<0;-C2;B2)	=A2-B2
4	=JEŻELI(C2>0;C2;A2)	=JEŻELI(C2<0;-C2;B2)	=A2-B2

W wierszach, począwszy od czwartego, zawartość jest już identyczna. Zwracam uwagę, że **zawsze** wykonuję operację $m - n$, natomiast decyzje podejmuje w zależności od znaku różnicy.

3. Gotowa aplikacja (jest taka!?) Jest! (Na przykład na telefony i tablety z Androidem)
4. Własna aplikacja

Szansę na znalezienie aplikacji, która rozwiązuje ten problem oceniam jako niewielkie, ale mogę się mylić.

Jak zatem stworzyć własną aplikację?

O tym właśnie będzie ten wykład.

1.4. Zabieramy się do programowania

Tworzenie aplikacji komputerowych (w slangu informatycznym) nazywa się **programowaniem**. Programowanie to *pisanie programu*.

Przyjrzyjmy się naszemu problemowi:

m, n

Mamy dwie dodatnie liczby całkowite m i n

- Takie obiekty (posiadające własną nazwę) występujące w programach nazywane są **zmiennymi**.
- Służą do przechowywania różnych wartości.
- Można ich używać jako argumentów w różnych operacjach matematycznych, mogą przechowywać ich wynik.

$m \leftarrow m - n$

Jest to operacja wykonywana na dwu zmiennych.

- Najpierw wykonywana jest operacja po prawej stronie strzałki (odejmowanie).
- W drugim kroku wynik operacji wstawiany jest jako nowa wartość zmiennej m .

Operacja warunkowa

Jeżeli m jest równe n

Operacja warunkowa („rozgałęzienie algorytmu”)

- Jeżeli wartość zmiennej m jest taka sama jak zmiennej n to algorytm kończy działanie.
- W algorytmie występują jeszcze wariant takiej operacji: **Jeżeli $m > n$** (wartość m jest większa niż n) **zrób coś w przeciwnym razie zrób coś innego.**

Przejdź do kroku 1.

Instrukcja skoku bezwarunkowego.

- Instrukcja ta wymusza „przejsie” na początek programu.
- Wykonywana jest zawsze.
- Instrukcja ta organizuje powtarzanie czynności **w pętli**:
 1. Jeżeli $m = n$ — koniec, największym wspólnym dzielnikiem jest n .
 2. Jeżeli $m > n$ przyjmij $m \leftarrow m - n$; w przeciwnym razie przyjmij $n \leftarrow n - m$
 3. Przejdź do kroku 1.

Można te dwie instrukcje (pierwsze **jeżeli** i ostatni skok zastąpić instrukcją zastępczą, która (po zanegowaniu warunku) będzie wyglądała tak:

1. Tak długo jak $m \neq n$ powtarzaj:
 - a) Jeżeli $m > n$ przyjmij $m \leftarrow m - n$; w przeciwnym razie przyjmij $n \leftarrow n - m$
2. Największym wspólnym dzielnikiem jest n .

1.4.1. Realizacja komputerowa

Lak te wszystkie działania zrealizować na komputerze? Pewno trzeba instalować jakieś dziwne oprogramowanie? Jak z niego korzystać?

1.5. Blockly

1. Do

- nauki programowania,
 - tworzenia prostych aplikacji,
 - sprawdzania koncepcji
- można użyć języka programowania Blockly.
2. Aplikację można łatwo przetestować.
 3. Dostępne w Internecie, ale również można sobie zainstalować na desktopie.
 4. Gotowe aplikacje przygotowane z jego użyciem: <https://blockly-games.appspot.com/>.

Instalacja Google Blockly...

...na lokalnym komputerze jest bardzo prosta:

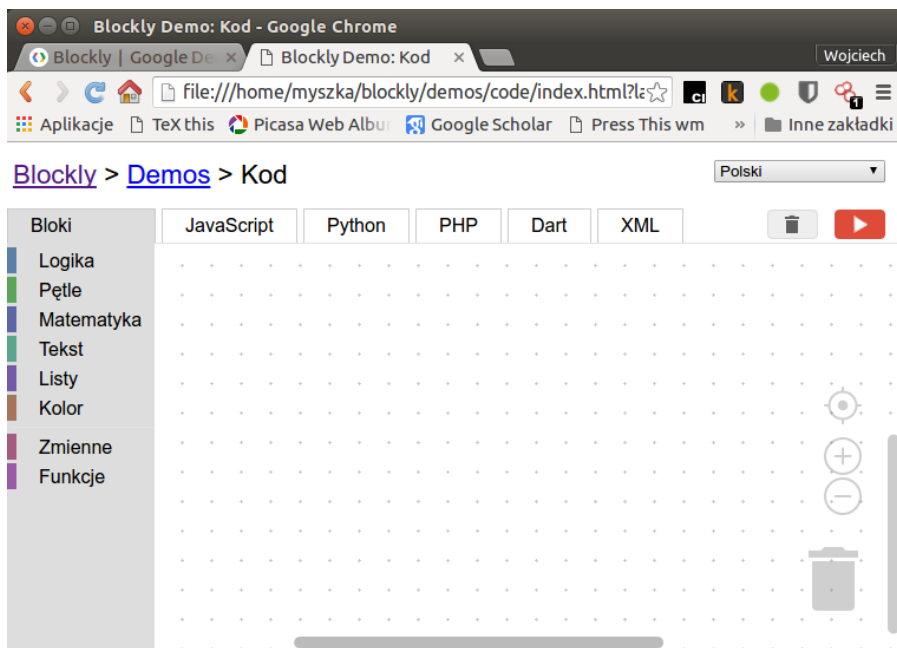
1. Ściągnąć musimy plik:
 - zip z adresu <https://github.com/google/blockly/archive/master.zip>
 - lub użyć programu git w formie: `git clone https://github.com/google/blockly.git`
2. Ściągnięte pliki (gdymy nie korzystamy z programu git) należy „rozpakować” (powinna zostać utworzona kartoteka o nazwie `blockly-master`)
3. Przechodzimy do „podkartoteki” `/demos/code/` i oglądamy w przeglądarce plik `index.html`

Tak na marginesie — Na serwerach github dostępna jest również wersja „gier” zrobionych w Blockly: <https://github.com/google/blockly-games>

1.5.1. Nasz pierwszy program w Blockly

Po uruchomieniu Blockly (treba wejść na tę stronę) zobaczymy coś przedstawionego na rysunku 1.2.

Po lewej stronie jest główne menu programu zawierające bibliotekę bloków dostępnych do programowania. Po prawej stronie, u góry jest przełącznik języka. Jak dla początkujących wszystko będzie teraz po polsku, ale już później (oraz na stronie z przykładami) — wyłącznie po angielsku. Zakładki nad „kropkowanym polem” pozwalają przetłumaczyć nasze dzieło do jednego z wybranych języków programowania:



Rysunek 1.2. Przestrzeń robocza Blockly

JavaScript, Python, PHP oraz DART. Zakładką XML pozwala zapisać nasze dzieło na później i ewentualnie odtworzyć.

W prawym dolnym rogu kosz na śmiecie (wrzucamy tam niepotrzebne bloczki) oraz trzy przyciski pozwalające wycentrować nasz projekt na stronie oraz powiększyć lub zmniejszyć jego widok.

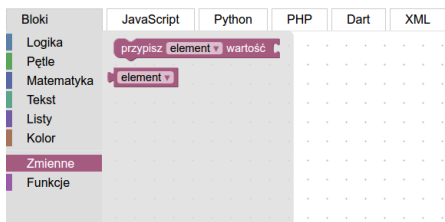
Kropkowane pole to przestrzeń robocza.

Omówię teraz pokrótce zawartość biblioteki, ale bez wdawania się w szczegóły — po pierwsze nikt tego nie przeczyta, a nawet jak przeczyta nie spróbuje i się nie nauczy.

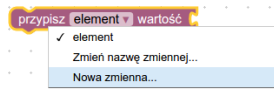
1.5.2. Zmienne

Pierwotnie w tej części biblioteki bloków nie jest wiele (rys.1.3). Jedyną zmienną nazywa się „element” (ang. *item*) i są dwa bloczki. Pierwszy z nich to odpowiednik operacji przypisania (nadania warto-

Blockly > Demos > Kod




Rysunek 1.3. Zawartość biblioteki Zmienne




Rysunek 1.4. Tworzenie nowej zmiennej

ści: \rightarrow), drugi to blok pozwalający użyć zmiennej po prawej stronie operatora przypisania.

Nowe zmienne tworzyć możemy używając bloczku przypisz (ang. *set*), klikając w trójkącik po prawej stronie nazwy zmiennej (rys. 1.4) i wybierając z menu „Nowa zmienna...” (otworzy się okno pozwalające nadać jej nazwę). Każda nowa zmienna automatycznie dodawana jest do biblioteki. Jeżeli zajdzie taka potrzeba można również zmienić nazwę zmiennej.

Początkową wartość jakiejś zmiennej nadajemy tak:  (ten niebieski klocek trzeba znaleźć w bibliotece matematyka i zamiast zera wpisać potrzebną wartość). Formalny zapis wyglądać będzie tak: $m \leftarrow 200$.

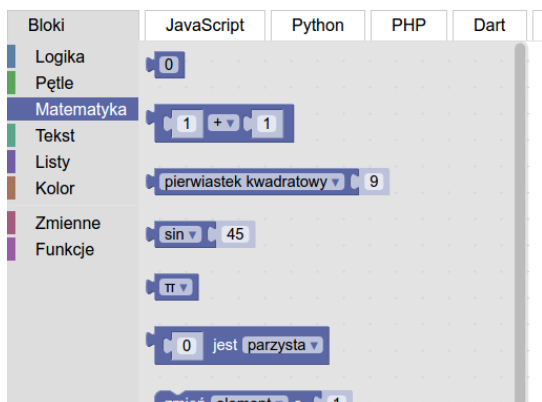
1.6. Operacja $m \leftarrow m - n$

Żeby wykonać tę operację musimy zajrzeć do biblioteki Matematyka (rys. 1.5) i wybrać bloczek odpowiedzialny za operacje arytmetyczne (). Ma on dwa pola, w które wstawić można albo wartość albo nazwę zdefiniowanej zmiennej. Rodzaj operacji matematycznej realizowanej przez blok określamy klikając w trójkącik koło znaczku plus i wybierając z menu.

Ostatecznie operacja $m \leftarrow m - n$ przyjmie postać



[Blockly](#) > [Demos](#) > Kod



Rysunek 1.5. Fragment zawartości biblioteki Matematyka

[Blockly](#) > [Demos](#) > Kod

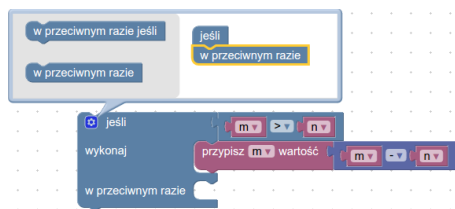


Rysunek 1.6. Fragment zawartości biblioteki Logika

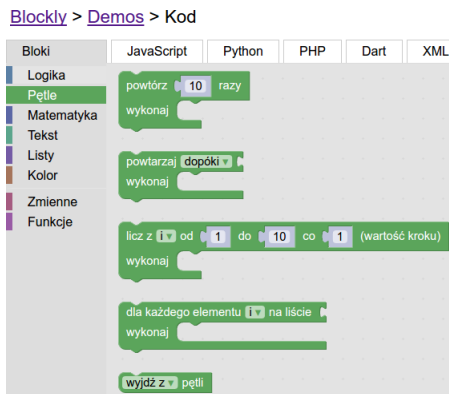
1.6.1. Operacja jeśli

Potrzebna nam operacja Jeżeli („Jeżeli m jest większe od n ”) znajduje się w bibliotece Logika (rys 1.6). Na samej górze jest bloczek Jeżeli — wykonaj (rys. 1.7).


Przed użyciem trzeba blok skonfigurować. Normalnie występuje on w postaci **jeśli** (tu jakiś warunek) *zrób coś*. Może występować również w wersji: **jeśli** (tu jakiś warunek) *zrób coś w przeciwnym razie zrób coś innego*. Wystarczy kliknąć w koło zębate i w dymku przesunąć „w przeciwnym razie” na prawą stronę (jeżeli taka akcja jest nam potrzebna).



Rysunek 1.7. Konfiguracja bloku „Jeśli — wykonaj”



Rysunek 1.8. Zawartość biblioteki Pętla

Warunek logiczny określający działanie bloku jeśli też znajdziemy w Bibliotece Logika. Do porównania dwu wartości służy blok: .

1.6.2. Pętla

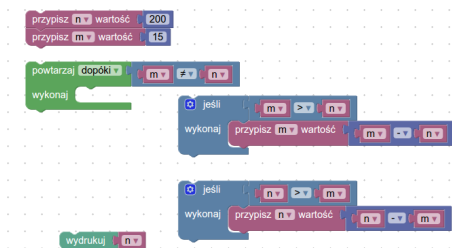
Na sam koniec zostaje pętla. To może być najtrudniejsze. W bibliotece Petle (rys. 1.8) znaleźć można kilka bloczków.

Patrząc na zmodyfikowany warunek („**tak długo jak m różne od n powtarzaj**”) wybieramy bloczek oznaczony opisem „powtarzaj dopóki (coś jest prawdą)”. W miejsce oznaczone „wykonaj” wstawić trzeba czynność (czynności) która ma być wykonywana...

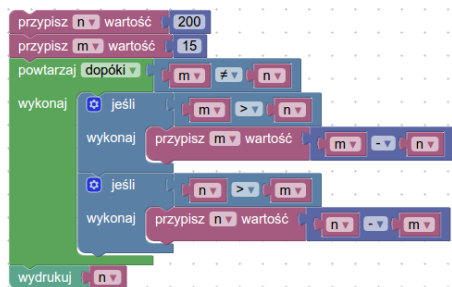
1.6.3. Prezentacja wyniku

W ogólnych zarysach opisałem wszystkie elementy składowe algorytmu (rys. 1.9). Pozostaje tylko wyprowadzenie wyniku działania programu. Wartość zmiennej n jest rozwiązaniem.

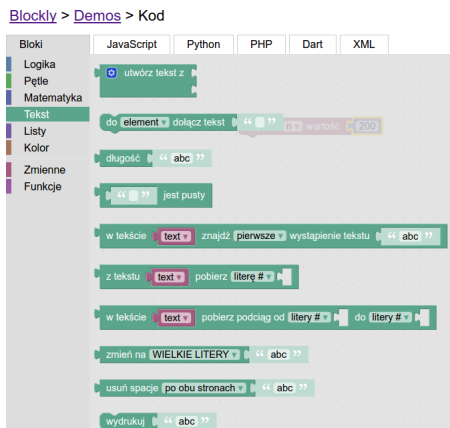
Wystarczy z biblioteki Tekst (rys. 1.10) wybrać Wydrukuj i dodać zmienną n . Gotowy program wygląda tak jak na rysunku 1.11. Dwie niezależne instrukcje **jeżeli $m > n$ zrób coś** i **jeżeli $n > m$ zrób coś**



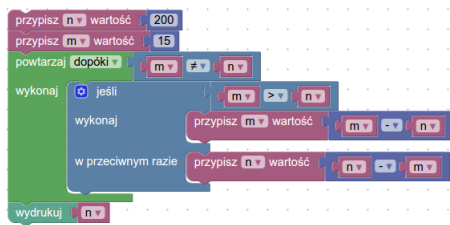
Rysunek 1.9. Elementy algorytmu



Rysunek 1.11. Pierwsza wersja programu



Rysunek 1.10. Fragment zawartości biblioteki Tekst



Rysunek 1.12. Wersja zmodyfikowana programu

innego można zastąpić jedną: **jeżeli** $m > n$ **zrób coś w przeciwnym razie** *zrób coś innego*, a program będzie taki jak na rysunku 1.12.

Proces tworzenia programu można prześledzić na stronie.