PROGRAMOWANIE OD PIERWSZOKLASISTY DO MATURZYSTY

Grażyna Koba

1. Wprowadzenie

Proces nauczania możemy porównać do działania komputera (rys. 1.). Na wejściu wprowadzamy informacje, które komputer przekształca na dane, czyli zapisuje w postaci zer i jedynek. Następnie, za pomocą odpowiednich programów komputerowych, przetwarza je w celu otrzymania wyników [1].

W szkole na wejściu jest uczeń i podręcznik zawierający treści nauczania oraz ćwiczenia zgodne z podstawą programową. Zaczynamy prowadzić pierwsze zajęcia, stosując odpowiednie metody i formy nauczania – uczniowie nabywają wiedzę i umiejętności („przetwarzamy dane"). Na wyjściu otrzymujemy odpowiednie wyniki kształcenia. Ale czy zawsze takie, jakie planowaliśmy? Niekoniecznie. Oczywiście chcielibyśmy, aby były jak najlepsze. Jak to zrobić?

Aby na wyjściu otrzymać bardzo dobre **wyniki kształcenia**, musimy nie tylko wprowadzić właściwe **informacje**, ale także właściwie je **przetwarzać**, czyli stosować skuteczne metody nauczania. Należy przede wszystkim prawidłowo określić „wyjście", czyli wiedzę i umiejętności, jakie uczeń powinien zdobyć w wyniku kształcenia informatycznego po danym etapie edukacyjnym (etapach edukacyjnych).

Przyjrzymy się przykładowemu „wyjściu" po ukończeniu szkoły podstawowej, gimnazjum i pierwszej klasy szkoły ponadgimnazjalnej. Posłużymy się przykładem z podstawy programowej dla szkoły ponadgimnazjalnej do informatyki realizowanej w zakresie rozszerzonym.

Rysunek 1. Proces nauczania jak działanie komputera

Uczeń powinien znać i stosować m.in. algorytm, np. wydawania reszty metodą zachłanną. Algorytm zaczyna się od sprawdzenia, czy w tabeli nominałów jest wartość mniejsza lub równa reszcie (oznaczmy ją *R*). Jeśli odnaleziony nominał *N* jest równy *R*, to algorytm się kończy. Jeśli nie, to wybieramy największą możliwą liczbę *L* sztuk znalezionego nominału *N* mniejszego od *R* (zakładamy, że tablica nominałów jest posortowana malejąco). Liczbę tę otrzymujemy, dzieląc *R* przez wartość odnalezionego nominału (liczba nominałów to oczywiście część całkowita z dzielenia). Następnie od reszty *R* odejmujemy wartość znalezionego nominału pomnożoną przez liczbę nominałów wynikającą z dzielenia. Szukanie powtarzamy wśród kolejnych nominałów dla pomniejszonej wartości *R* do czasu aż wypłacimy całą sumę [2].

Zastanówmy się, co uczeń powinien umieć, aby zaimplementować ten algorytm w języku wysokiego poziomu, np. C++. Przede wszystkim powinien umieć **myśleć**! Ponadto znać zasady tworzenia programu komputerowego, w tym zasady składni i słowa kluczowe wybranego języka programowania. W przypadku tego algorytmu uczeń będzie musiał jeszcze umieć stosować zmienne (w tym tablice), wprowadzać dane, wyprowadzać wyniki, stosować instrukcje iteracyjne (*for*, *while*), instrukcję warunkową (*if*), definiować podprogramy (tu funkcje) bez parametrów i z parametrami oraz wywoływać je (szczegółowy opis algorytmu i rozwiązanie w języku C++ **–** zob. [2] i [3]).

Jaką wiedzę i umiejętności uczeń może zdobyć wcześniej? W następnych punkach pokażę, czego można nauczyć już na pierwszym etapie edukacyjnym i na kolejnych, aby w liceum uczeń szybciej opanował programowanie w języku wysokiego poziomu.

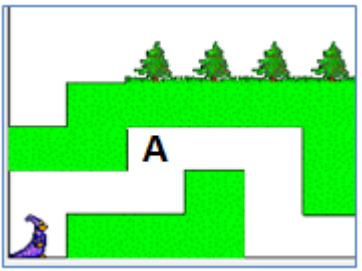
2. Pierwszy etap edukacyjny

Na początku pierwszej klasy szkoły podstawowej uczeń dowiaduje się, że „praca z komputerem to praca z programami komputerowymi i każdy program trzeba uruchomić” [4]. Uzmysławiamy uczniom, czym jest program komputerowy (bez podawania jego definicji).

Uczniowie od pierwszych zajęć zapoznają się z dydaktycznym środowiskiem programowania (Baltie) – systematycznie przygotowują się do nauki programowania. Jeszcze nie programują, ale rozwiązują zadania logiczne, w których układają obiekty w serie rosnące i malejące, szukają najkrótszej drogi, znajdują element najmniejszy lub największy. Wprowadzamy mimochodem myślenie algorytmiczne.

W klasie drugiej uczniowie uczą się wydawania poleceń za pomocą listy kroków, sterując postacią na ekranie. Na przykład, aby czarodziej przeszedł z pozycji początkowej (z dolnego lewego rogu sceny) do punktu A po ścieżce pokazanej na rysunku 2., powinien wykonać następujące czynności:

1. Obrócić się w lewo.
2. Przejść krok.
3. Obrócić się w prawo.
4. Przejść dwa kroki.
5. Obrócić się w lewo.
6. Przejść krok.



Rysunek 2. Sterowanie postacią na ekranie

Na tym przykładzie uczniowie powinni zauważyć, że czarodziej dojdzie do punktu A tylko w przypadku, jeśli wykona czynności podane w liście kroków i w takiej kolejności. Na przykład zmiana w kroku 3. obrotu w prawo na obrót w lewo spowoduje, że czarodziej będzie próbował wyjść poza scenę (co i tak się nie uda ☺). Również inna kolejność czynności spowoduje, że Baltie nie dojdzie do celu.

Takie i podobne ćwiczenia można wykonywać z uczniami na początkowych zajęciach nawet bez użycia komputera, chodząc po podłodze po wyznaczonej ścieżce, zbudowanej z… patyczków, sznurków itp.

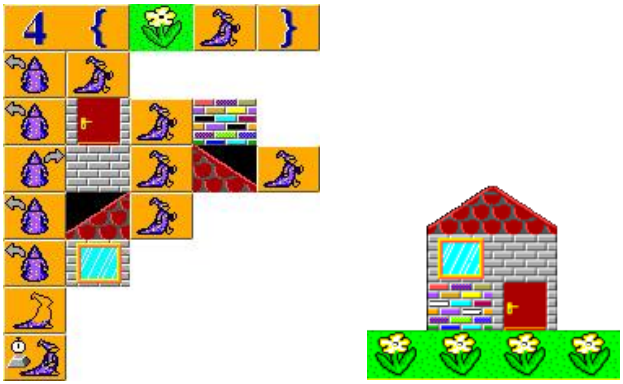
Po dobrym przygotowaniu uczniowie w trzeciej klasie zaczynają „poważnie” programować. Układają polecenia dla czarodzieja w obszarze

roboczym okna programu (tworzą program), a następnie uruchamiają program i obserwują efekt jego wykonania. W przystępny sposób dowiadują się, na czym polega tworzenie programu komputerowego [6].

Można zwrócić uczniom uwagę, że polecenia programu wykonywane są według kolejności wierszy, od lewej strony do prawej, podobnie do odczytywania słów w książce. Po wykonaniu wielu ćwiczeń uczniowie sami zauważą, że wstawienie poleceń w złej kolejności lub wstawienie błędnego polecenia może spowodować, że efekt wykonania programu będzie odmienny od oczekiwanego, np. czarodziej wybuduje okno na dachu domu obok komina.

Pokazujemy również, na czym polega powtarzanie poleceń – jeśli pewne operacje powtarzają się, można je ująć w nawias, a przed nawiasem umieścić liczbę powtórzeń.

Uczniowie powinni na tym etapie wykonać jak najwięcej prostych zadań, utrwalających te zasady. Przykładowe zadanie: *Utwórz program, w którym Baltie wyczaruje łączkę z czterema żółtymi kwiatkami i na niej wybuduje dom jak na rysunku 3.* [5].



Rysunek 3. Program utworzony w środowisku Baltie i efekt jego wykonania

3. Drugi etap edukacyjny

W obowiązującej podstawie programowej dla drugiego etapu edukacyjnego treści nauczania dotyczące elementów programowania zostały określone następująco: *uczeń za pomocą ciągu poleceń tworzy proste motywy lub steruje obiektem na ekranie*. Od dziesięciu lat proponuję w podręcznikach swojego autorstwa znaczne poszerzenie tych treści [7].

Uczniowie, wykonując wiele ćwiczeń, uczą się rozwiązywania problemów i poznają sposoby stosowania do tego celu programów edukacyjnych – środowisk programowania, tj.: Baltie, Scratch, Logomocja.

Na drugim etapie edukacyjnym więcej programujemy. Definiujemy ważniejsze pojęcia, m.in.: program jako ułożone w odpowiedniej kolejności polecenia dla komputera. Zwracamy uwagę, że każdy język programowania posiada zbiór instrukcji i określone zasady składni. W niektórych środowiskach dydaktycznych (np. Baltie, Scratch) instrukcje reprezentowane są przez elementy graficzne, również stosowane według określonych zasad [8].

Należy uzmysławiać uczniom, że… to komputer wykonuje programy komputerowe, a nie czarodziej (Baltie), żółw (Logomocja) czy duszek (Scratch).

Na tym etapie uczniowie tworzą trudniejsze programy wykorzystujące animacje. Program pokazany na rysunku 4. realizuje zadanie: *Utwórz program, w którym Baltie buduje dom pokazany na rysunku. Następnie staje z lewej strony drzwi zwrócony w prawą stronę, otwiera je, wchodzi do domu i zamyka drzwi za sobą. Do otwierania i zamykania drzwi zastosuj definiowanie przedmiotu animowanego* [7].

Rysunek 4. Program utworzony w środowisku Baltie wykorzystujący animacje i efekt jego wykonania

Uczniowie chętnie tworzą gry komputerowe. Należy maksymalnie wykorzystać ich zaangażowanie. Podczas tworzenia gier, przy okazji zabawy, uczniowie uczą się rozwiązywania problemów i poznają podstawowe zasady programowania.

Przykładowa gra: *Utwórz grę „Dwa duszki", w której dwa duszki będą biegać po scenie (wzór na rysunku 5.). Pierwszym duszkiem będziemy sterować, naciskając klawisze sterujące ze strzałkami. Drugi ma poruszać się losowo po całej scenie. Jeśli duszki się dotkną, otrzymujemy 1 punkt. Gra kończy się, gdy zdobędziemy 5 punktów. Na koniec powinien wyświetlić się komunikat „Wygrałeś!".*

Rysunek 5. Scena do gry „Dwa duszki"

Tworząc grę, uczniowie poznają sytuację warunkową – duszek-mysz powinien poruszać się zależnie od naciśniętego klawisza strzałki. Dzięki temu uczą się stosowania instrukcji warunkowej w języku Scratch.

Rysunek 6. Stosowanie instrukcji warunkowej w języku Scratch

Drugi duszek-piłka powinien poruszać się losowo po całym ekranie. Uczniowie muszą zaplanować sposób zapisu tej sytuacji w języku Scratch. W tym celu stosują odpowiednie polecenie powtarzania i polecenie zmieniające położenie duszka.

Wykonując to zadanie, uczniowie poznają również prostokątny układ współrzędnych – już po kilku zadaniach bardzo dobrze radzą sobie z określeniem współrzędnych *x* i *y* położenia duszka (nie mają również problemu z liczbami ujemnymi). Dodatkowo poznają w praktyce, na czym polega losowe wybieranie liczb – działanie generatora liczb losowych (rys. 7.).

Rysunek 7. Losowe przemieszczanie duszka po scenie

Kolejnym zadaniem jest dodanie do programu zliczania punktów (licznik ma zwiększać się o jeden, gdy duszki się dotykają). Uczeń ponownie spotyka się z sytuacją warunkową, ale zna już sposób jej realizacji w języku Scratch, więc sobie poradzi.

Większym problemem dla uczniów jest zastosowanie zmiennej, w której będzie pamiętana wartość licznika. Pojęcie zmiennej jest dla uczniów na tym etapie edukacyjnym trudne. Należy w prosty sposób wyjaśnić, co oznacza, że zmiennej o określonej nazwie przypisana jest konkretna wartość. Natomiast na tym etapie edukacyjnym nie musimy wyjaśniać przypisania *licznik* := *licznik* +1 – wystarczy krótkie wyjaśnienie, że w poleceniu 

wartość licznika zwiększa się o jeden.

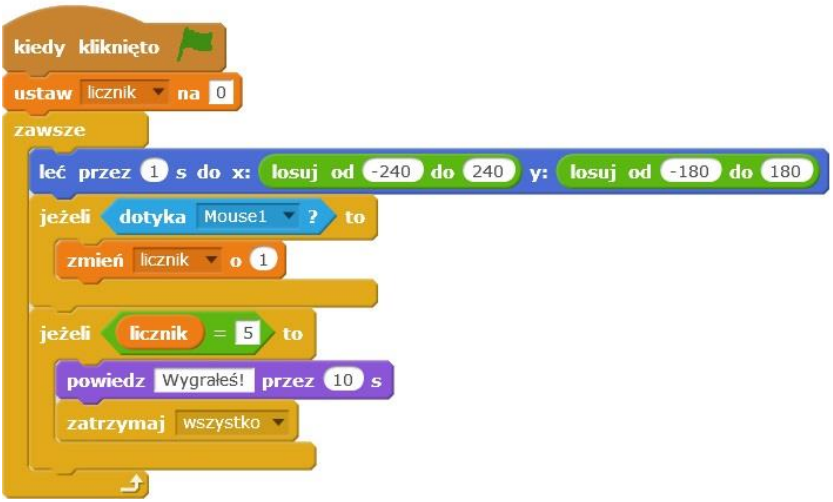
Jeśli chcemy, aby punkty w grze zliczały się zawsze od zera, należy zwrócić uwagę na konieczność wyzerowania licznika na początku programu (ustawienie wartości zmiennej *licznik* na zero).

Na koniec dodajemy warunek zakończenia programu – uczniowie kolejny raz stosują polecenie warunkowe, ale tym razem muszą użyć jako warunku wyrażenia logicznego, w którym występuje zmienna (rys. 8.). Częstym błędem w tym przypadku jest wpisywanie w wyrażeniu warunkowym napisu „licznik” zamiast wstawienia do odpowiedniego pola

nazwy zmiennej *licznik* (tu elementu: licznik). Należy zwracać na takie błędy uwagę od samego początku.

Tworzenie tego rodzaju gry inspiruje do dalszego jej modyfikowania, co uczniowie chętnie robią samodzielnie. Na przykład dodają trzeciego duszka losowo przemieszczającego się po scenie, którego dotknięcie zmniejsza licznik o jeden. Mierzą również czas trwania gry.

Gra jest jeszcze ciekawsza, gdy wprowadzi się kolejnego gracza, czyli duszka, którym sterujemy innymi klawiszami niż strzałki. Wtedy uczniowie mogą grać ze sobą lub z nauczycielem, a w domu – z rodzicami lub rodzeństwem. Gdy uczniowie zaczynają grać w utworzone przez siebie gry, zapominają, że chcieli grać w jakieś inne gry lub korzystać z portali społecznościowych.



Rysunek 8. Skrypt dla duszka-piłki z uwzględnieniem warunków zakończenia gry

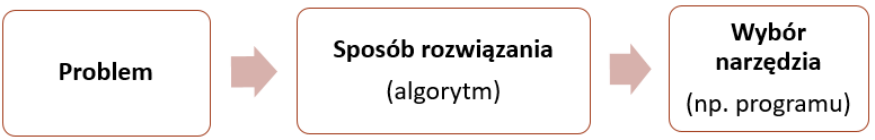
W drugim etapie edukacyjnym należy zwracać uwagę na optymalizowanie programu i na styl programowania – uczniowie często używają zbyt dużej liczby poleceń, zapominają również o stosowaniu powtarzania poleceń i często zapisują polecenia w jednym długim wierszu.

Uczniowie powinni wykonywać jak najwięcej zadań, w których pojawiają się animacje i tworzenie gier komputerowych. Takie programy wyzwalają w uczniach szczególną aktywność twórczą. Rozwiązując tego typu zadania, rozwijają również logiczne myślenie i poznają podstawowe zasady programowania.

4. Trzeci etap edukacyjny

W trzecim etapie edukacyjnym rozwijamy i rozszerzamy podejście algorytmiczne do rozwiązywania problemów. Uczniowie dowiadują się, co trzeba zrobić, aby rozwiązać dowolny problem (zadanie) – należy go wcześniej poprawnie sformułować oraz ustalić dane i określić cel, czyli wyniki. Następnie należy zastanowić się nad sposobem rozwiązania, czyli algorytmem, a także wyborem odpowiedniego narzędzia, np. programu komputerowego, który to umożliwi (rys. 9.).

Uczniowie dowiadują się, czym jest algorytm, poznają określenie specyfikacji zadania oraz sposoby przedstawiania algorytmów. Poznają i opisują wybrane algorytmy, np. algorytm znajdowania wybranego elementu w zbiorze uporządkowanym i nieuporządkowanym, algorytm porządkowania elementów [9].



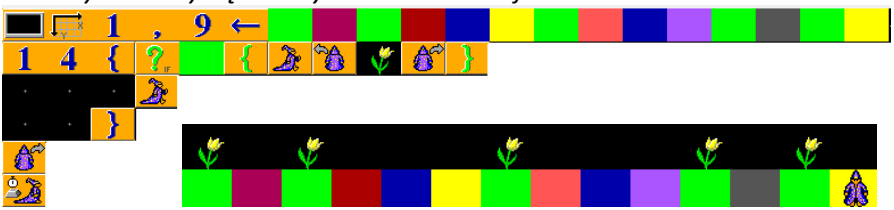
Rysunek 9. Schemat postępowania przy rozwiązywaniu problemów za pomocą komputera

W celu ułatwienia zrozumienia wybranych algorytmów można niektóre z nich wykonać z uczniami praktycznie, posługując się przygotowanymi wcześniej rekwizytami [10].

Dopiero po zrozumieniu danego algorytmu (często po wykonaniu go praktycznie), uczniowie przedstawiają go w postaci listy kroków i schematu blokowego, a także zapisują w postaci programu komputerowego. Korzystają z poznanych na wcześniejszych etapach edukacyjnych środowisk programistycznych (Baltie, Scratch i Logomocja). Dzięki temu mogą skupić się na rozwiązywaniu problemów, a nie na zapoznawaniu się z nowym środowiskiem. Zależnie od grupy uczniów, można w ostatniej klasie gimnazjum wprowadzić wybrany język wysokiego poziomu (dla uczniów zainteresowanych).

W gimnazjum większą uwagę zwracamy na analizę rozwiązania. Na przykład po wykonaniu zadania, w którym Baltie ma umieścić kwiatki na zielonych przedmiotach (rys. 10.), uczniowie odpowiadają na pytania: *Jakie*

polecenie jest wykonywane, gdy warunek jest spełniony, a jakie, gdy nie jest spełniony? Ile razy będzie wykonana instrukcja warunkowa?



Rysunek 10. Przykład zastosowania instrukcji warunkowej w środowisku Baltie

Jeszcze więcej programujemy – uczniowie stosują zmienne, wykonują na nich obliczenia, wprowadzają dane z klawiatury, wyprowadzają wyniki na ekran, stosują instrukcje powtarzania i warunkową oraz podprogramy (bez parametrów i z parametrami).

W przypadku używania zmiennych należy dokładnie wyjaśnić uczniom, na czym polega wprowadzanie wartości zmiennej z klawiatury. Na przykład po napisaniu i uruchomieniu programu, który ma umożliwić wprowadzenie dwóch różnych liczb z klawiatury i wyprowadzenie większej z nich (rys. 11.), uczeń czeka i dziwi się, że nic się nie dzieje… Nie rozumie, że to on jest użytkownikiem i powinien wprowadzić z klawiatury dwie liczby do porównania, ponieważ tak to jest zap

Rysunek 11. Stosowanie zmiennych w języku Scratch

W trzecim etapie edukacyjnym wprowadzamy definiowanie i wywoływanie podprogramów (procedur). Należy dokładnie wyjaśnić

uczniom, że kroki, które powtarzają się w algorytmie, można potraktować jako problem cząstkowy i przedstawić w postaci podprogramu (procedury lub funkcji). Umożliwia to opracowanie każdego problemu cząstkowego oddzielnie. Aby napisać cały program, wystarczy zestawić podprogramy i odpowiednio je połączyć. Połączenia podprogramu z programem głównym realizuje się poprzez instrukcję wywołania podprogramu [9].

Procedury można definiować w każdym ze znanych im już środowisk dydaktycznych: Baltie, Scratch i Logomocja).

Uczniowie powinni również dobrze zrozumieć, czym są parametry w procedurze, a także rozróżniać parametry formalne od aktualnych (w definicji procedury określamy parametry formalne, które w momencie wywołania procedury są zastępowane przez parametry aktualne).

Na rysunku 12. pokazana jest przykładowa definicja procedury *wielokąt* z parametrem określającym liczbę boków wielokąta (*liczba_boków*). Procedura jest wywoływana z parametrem aktualnym *n*, którego wartość jest wprowadzana z klawiatury w momencie wywołania procedury.

Należy też uczniom wyjaśnić, że wywołanie procedury powoduje wykonanie składających się na jej treść instrukcji.

Rysunek 12. Stosowanie procedur z parametrami w języku Scratch

Procedury można wprowadzić również w drugim etapie edukacyjnym, np. w klasie VI, ale bez parametrów. Stosowanie parametrów jest dla uczniów ze szkoły podstawowej zbyt trudne i może być niezrozumiałe.

5. Czwarty etap edukacyjny

Nadszedł czas odpowiedzi na pytania: Czy otrzymaliśmy zadowalający wynik kształcenia po dziewięciu latach? Czy udało nam się zapoznać uczniów z programowaniem tak, aby poradzili sobie na czwartym etapie edukacyjnym z programami w języku wysokiego poziomu, np. C++? Czy uczeń zdobył wystarczającą wiedzę—i umiejętności na wcześniejszych etapach, aby na czwartym etapie nie zaczynać wszystkiego od początku, tylko korzystać z wcześniej zdobytej wiedzy?

Odpowiedzi powinny być twierdzące, jeśli uczniowie zdobywali wiedzę i umiejętności z programowania systematycznie i poprawnie pod względem merytorycznym oraz dydaktycznym.

Uczniowie na poprzednich etapach edukacyjnych rozwijali myślenie algorytmiczne, poznawali podstawowe zasady programowania, stosując zmienne oraz podstawowe instrukcje (powtarzania, warunkową) i procedury. Nauczyciel prowadzący zajęcia na czwartym etapie powinien nawiązywać do tej wiedzy i umiejętności uczniów, pokazując podobieństwa.

Uczniowie spotykali się na przykład w rozwiązywanych zadaniach z sytuacją warunkową. Wiedzą zatem, że sytuację warunkową zapisuje się, stosując instrukcję warunkową. Na lekcji informatyki w szkole ponadgimnazjalnej uzupełniamy wiedzę uczniów. Wyjaśniamy, że działanie instrukcji warunkowej jest w większości języków programowania podobne (rys. 13.). Sprawdzany jest warunek logiczny – jeśli jest prawdziwy, to wykonywana jest *instrukcja1*; jeśli fałszywy – wykonywana jest *instrukcja2* (po słowie **else**), a jeśli brak tej części instrukcji – wykonywana jest od razu kolejna instrukcja, zapisana po instrukcji warunkowej [11].

Rysunek 13. Porównanie instrukcji warunkowych: w języku Scratch i C++

Uczniowie sami powinni zauważyć, że sytuacje warunkowe realizuje się tak samo w języku Scratch, jak i w języku C++ i działanie instrukcji warunkowej jest takie samo.

Należy polecić uczniom wykonanie w języku C++ takiego samego zadania, które wykonywali w gimnazjum w języku Scratch: *Napisz program, który ma umożliwić wprowadzenie dwóch różnych liczb z klawiatury i wyprowadzenie większej z nich* (rys. 14.).

Jeśli uczniowie dobrze rozumieli to zadanie, bez problemu (oczywiście po zapoznaniu się z podstawami pisania programu w języku C++, niezbędnymi do zapisania tego zadania) zapiszą ten algorytm w języku wysokiego poziomu.

Ucząc programowania na czwartym etapie edukacyjnym w zakresie podstawowym, należy koniecznie omówić rozwiązywanie problemów algorytmicznych i programowanie w języku wysokiego poziomu tak, aby możliwe było znaczne rozszerzenie tych zagadnień podczas realizacji informatyki w zakresie rozszerzonym. Należy jednak pamiętać, że nawet na informatyce realizowanej w zakresie rozszerzonym, uczniowie powinni zaczynać od prostych przykładów. Podawanie im od razu trudnych zadań zniechęci ich do dalszej nauki programowania.

```
1   #include <iostream>
2
3   using namespace std;
4
5   int liczba1, liczba2;
6
7   int main()
8   {
9       cout << "Podaj pierwsza liczbe: ";
10      cin >> liczba1;
11      cout << "Podaj druga liczbe: ";
12      cin >> liczba2;
13      if(liczba1>liczba2)
14          cout << "wieksza jest: " << liczba1;
15      else cout << "wieksza jest: "  << liczba2;
16
17
18      return 0;
19  }
```

Rysunek 14. Program w języku C++ realizujący zadanie wyboru większej z dwóch liczb

6. Podsumowanie

Zgodnie z teorią konstruktywizmu w nauczaniu, uczący się aktywnie konstruuje własną wiedzę, gdy wykorzystuje wiedzę już posiadaną. Aby uczeń mógł świadomie wykorzystywać nabywaną wiedzę, musi być ona przekazywana prawidłowo pod względem merytorycznym i metodycznym oraz podawana w sposób uniwersalny, systematyczny oraz uporządkowany.

Na kolejnych etapach edukacyjnych powinniśmy zapewniać uczniom właściwe rozumienie pojęć, metod i procesów związanych z programowaniem. Podstawowe treści nauczania (pojęcia, metody, w tym zasady programowania) powinny być niezależne od wykorzystywanego środowiska programistycznego.

Szczególną uwagę powinniśmy zwracać na:

- umiejętność myślenia algorytmicznego;
- rozumienie, że postać instrukcji musi być zgodna z zasadami składni danego języka – dotyczy to również wyboru odpowiedniego elementu graficznego;
- rozumienie, że kolejność występowania instrukcji w programie (także prezentowanych przez elementy graficzne) powinna odpowiadać kolejności operacji realizujących dany algorytm;
- rozumienie, że programy wykonuje komputer, a nie czarodziej, duszek czy żółw;
- wykonywanie wielu podobnych zadań z zastosowaniem poszczególnych instrukcji i zasad programowania;
- umiejętność zastosowania poznanej zasady programowania w innym zadaniu;

i na koniec – na samodzielną pracę uczniów.

Literatura

1. Koba G., *Z nowym bitem. Informatyka dla gimnazjum. Część I,* MIGRA, Wrocław 2015.
2. Koba G., *Informatyka dla szkół ponadgimnazjalnych – zakres rozszerzony,* MIGRA, Wrocław 2013.
3. Koba G., *Poradnik metodyczny. Informatyka dla szkół ponadgimnazjalnych. Zakres rozszerzony,* MIGRA, Wrocław 2013.
4. Koba G., *Zajęcia komputerowe dla szkoły podstawowej. Klasa I*, MIGRA, Wrocław 2009.
5. Koba G., *Zajęcia komputerowe dla szkoły podstawowej. Klasa III*, MIGRA, Wrocław 2011.
6. Koba G., *Poradnik metodyczny. Zajęcia komputerowe dla szkoły podstawowej. Klasy I-III*, MIGRA, Wrocław 2011.
7. Koba G., *Zajęcia komputerowe dla szkoły podstawowej. Klasy IV-VI*, MIGRA, Wrocław 2012.
8. Koba G., *Poradnik metodyczny. Zajęcia komputerowe dla szkoły podstawowej. Klasy IV-VI*, MIGRA, Wrocław 2012.
9. Koba G., *Z nowym bitem, Informatyka dla gimnazjum. Część II*, MIGRA, Wrocław 2016.
10. Koba G., *Poradnik metodyczny. Z nowym bitem, Informatyka dla gimnazjum*, MIGRA, Wrocław 2014.
11. Koba G., *Z nowym bitem, Informatyka dla szkół ponadgimnazjalnych. Zakres podstawowy*, MIGRA, Wrocław 2015.