Podstawy Programowania 1

Wstęp

Arkadiusz Chrobot

Katedra Systemów Informatycznych

3 października 2024

Plan

1. Informacje organizacyjne
2. Bibliografia
3. Wprowadzenie
4. Algorytm
5. System komputerowy
6. Języki programowania
7. Język C
   - Podstawy
   - Komentarze
   - Stałe
   - Zmienne i typy zmiennych

Informacje organizacyjne

Wykładowca: dr inż. Arkadiusz Chrobot

Numer pokoju: 3.23, budynek D

Termin konsultacji: czwartek, 12:00 – 14:00

Numer telefonu: 41 34-24-185

Adres e-mail: a.chrobot@tu.kielce.pl

Strona WWW: https://achilles.tu.kielce.pl/portal/Members/84df831b59534bdc88bef09b15e73c99

Karta przedmiotu

Karta przedmiotu

Bibliografia

Literatura Podstawowa

1. **Brian W. Kernighan, Dennis M. Ritchie, „Język ANSI C. Programowanie", Wydanie 2, Wydawnictwo Helion, Gliwice 2010**
2. **Stephen Prata, „Język C. Szkoła programowania", Wydanie 6, Helion, Gliwice 2016**
3. Zed A. Shaw, „Programowanie w C. Sprytne podejście do trudnych zagadnień, których wolałbyś unikać (takich jak język C)", Wydanie 1, Helion, Gliwice 2016
4. Paul Deitel, Harvey Deitel, „Język C. Solidna wiedza w praktyce", Wydanie 8, Helion, Gliwice 2020
5. Piotr Wróblewski, „Algorytmy, struktury danych i techniki programowania", Helion, Gliwice 1997
6. Jon Bentley, „Perełki Oprogramowania", WNT, Warszawa 1992
7. Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman „Algorytmy i struktury danych", Helion, Gliwice 2003

Bibliografia

Literatura uzupełniająca

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, „Wprowadzenie do algorytmów”, WNT, Warszawa 1998
2. Donald E. Knuth, „Sztuka programowania”, WNT, Warszawa 2002
3. Robert Sedgewick, Kevin Wayne, „Algorytmy“, Wydawnictwo Helion, Gliwice, 2017
4. Steven S. Skiena, “The Algorithm Design Manual”, Springer-Verlang, London 2010 (w języku angielskim)
5. Jarosław Stańczyk „Nowoczesny C: Przegląd C23 z przykładami”, Helion, Gliwice, 2023
6. Arkadiusz Chrobot, Adam Krechowicz, „Wprowadzenie do podstaw programowania”, Wydawnictwo Politechniki Świętokrzyskiej, Kielce, 2023

Bibliografia

Zasoby internetowe

1. Wikibooks: Język C
2. The GNU C programming tutorial
3. Learning GNU C
4. The GNU C Library

Programowanie

Definicja programowania

Programowanie jest to zespół czynności, którego celem jest przygotowanie programu dla *systemu komputerowego*. W skład tych czynności wchodzą: opracowanie modelu problemu, który chcemy rozwiązać za pomocą systemu komputerowego, opracowanie *algorytmu* (lub algorytmów) rozwiązania, zapisanie tego algorytmu za pomocą wybranego języka programowania oraz usunięcie błędów składniowych i logicznych.

Program komputerowy

Definicja programu komputerowego

Program komputerowy jest to zapis w określonym języku programowania *algorytmu* lub algorytmów rozwiązywania pewnych problemów.

Algorytm

Definicja algorytmu

Algorytm jest to skończony zbiór precyzyjnie zdefiniowanych czynności koniecznych do wykonania określonego zadania.

Własności algorytmu

Skończoność: wykonanie algorytmu zawsze musi się zatrzymać (dać wynik) po skończonej liczbie kroków. Procedura, która ma wszystkie własności algorytmu, poza skończonością nazywana jest procedurą obliczeniową.

Dobre zdefiniowanie: każdy krok algorytmu musi być precyzyjnie opisany, dla każdego przypadku akcje muszą być opisane ściśle i jednoznacznie.

Dane wejściowe: algorytm ma zero lub więcej danych wejściowych.

Dane wyjściowe: algorytm generuje jedną lub więcej danych wyjściowych, czyli wartości w określony sposób powiązanych z danymi wejściowymi.

Efektywność: z praktycznego punktu widzenia, algorytm nie tylko powinien dać się wykonać w skończonym czasie, ale również ten czas powinien być możliwie krótki.

Zapis algorytmu

Algorytmy mogą być zapisane w formie, która jest bardziej zrozumiała dla człowieka, lub w formie bardziej zrozumiałej dla komputera. Granice między tymi dwoma kategoriami nie są ostre.

Zapis algorytmu

Opis problemu

Algorytm Eukildesa wyznaczania największego wspólnego dzielnika liczb całkowitych (NWD).

Problem

Dane są dwie liczby całkowite M i N, należy znaleźć ich największy wspólny dzielnik, tj. największą dodatnią liczbę całkowitą, która dzieli całkowicie zarówno M, jak i N.

Zapis algorytmu

Pseudokod

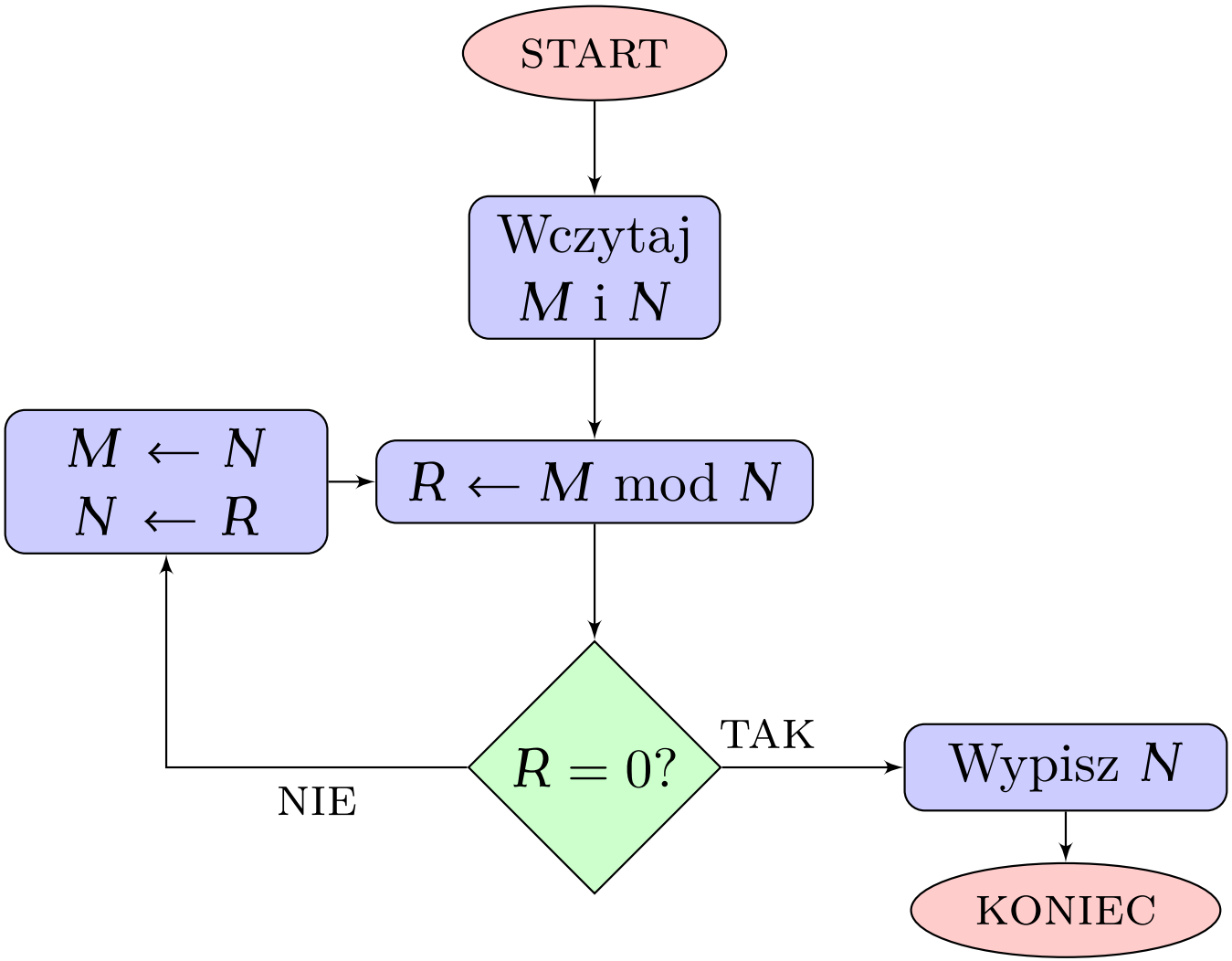
E1.[Znajdowanie reszty] Podziel M przez N i niech R oznacza resztę z tego dzielenia. ($\boldsymbol{Mamy}$ $0 \leqslant R < N$).

E2.[Czy wyszło zero?] Jeśli $R = 0$, to zakończ algorytm, odpowiedzią jest N.

E3.[Upraszczanie] Wykonaj $N \to M$, $R \to N$ i wróć do kroku E1.

Zapis algorytmu

Schemat blokowy



System komputerowy

Definicja systemu komputerowego

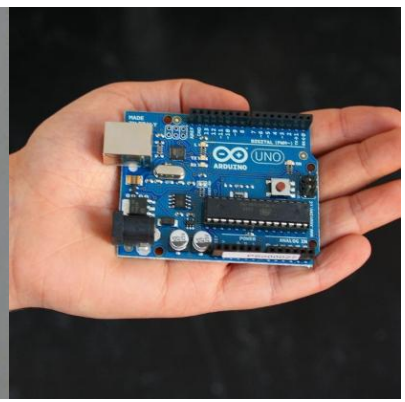
System komputerowy to urządzenie lub zespół współpracujących urządzeń, które mogą zrealizować określony, przygotowany dla nich program.

Rodzaje systemów komputerowych

(a) Klaster

(b) Komputer osobisty

(c) Urządzenia mobilne



(d) Mikrokontroler

...i jeszcze kilka innych kategorii.

Rodzaje systemów komputerowych - elementy wspólne

Każdy system komputerowy zawiera co najmniej dwa elementy:

1. procesor,
2. pamięć.

Każdy komputer „rozumie” zapis dwójkowy (binarny).

Zapis algorytmu - program dla komputera

Pierwotnie programy były zapisywane w kodzie maszynowym, czyli ciągu liczb binarnych (lub pokrewnych). Następnie pojawiły się assemblery, czyli języki, w których pojedyncze instrukcje maszynowe oznaczono łatwymi do zapamiętania ciągami znaków. Następnie pojawiły się języki wysokiego poziomu, w których zapis programu jest zbliżony do zapisu w języku naturalnym (najczęściej angielskim), a jedna instrukcja może odpowiadać kilku instrukcjom maszynowym. Tak zapisane programy są tłumaczone na język maszynowy za pomocą programów nazywanych translatorami, które dzielą się na kompilatory i interpretery.
Czynność zapisywania algorytmu w postaci programu komputerowego lub zapisany algorytm nazywamy *implementacją*. Program zapisany w assemblerze lub języku wysokiego poziomu nazywamy *kodem źródłowym*. Powstały na jego podstawie program wykonywalny nazywamy *kodem wynikowym*.

Abstrakcja

Rozwój języków programowania jest przykładem zastosowania abstrakcji - metody upraszania problemu polegającej na uwypukleniu jego najważniejszych atrybutów, a ukryciu tych, które nie są istotne dla jego rozwiązania. Programowanie sprowadza się do sprawnego posługiwania się abstrakcją.

Język C

Kilka własności

- opracowany w latach 70 ubiegłego wieku,
- język wysokopoziomowy, ale zawierający dużo niskopoziomowych elementów,
- język imperatywny, wspierający proceduralny model programowania,
- ma prostą składnię, która została zapożyczona przez wiele innych języków programowania (Java, C++, C#, itd.),
- jeden z najpopularniejszych języków programowania według rankingu TIOBE,
- jest kompilowany, istnieje wiele kompilatorów dostępnych dla różnych systemów komputerowych,
- język jest ustandaryzowany (najnowsza wersja standardu to ISO C23, na wykładzie będziemy używać przeważnie ISO C99 i kilku niestandardowych rozszerzeń GNU C).

Najprostszy program

Listing 1: Najprostszy program w C

```
1  int main(void)
2  {
3          return 0;
4  }
```

Program “Hello World!”

Listing 2: Program ”Hello World!” w C

```
1  #include<stdio.h>
2
3  int main(void)
4  {
5          puts("Hello World!");
6          return 0;
7  }
```

Komentarze

Komentarze w kodzie źródłowym służą do umieszczania przez programistę notatek odnoszących się do konkretnych fragmentów programu. Komentarze są ignorowane przez kompilator. Czasem wykorzystuje się tę własność komentarzy, aby chwilowo wyłączyć fragmenty kodu, które nie działają. Język C udostępnia trzy sposoby wprowadzania komentarzy do kodu źródłowego. Pierwszy polega na umieszczeniu komentarza między następującymi parami znaków: `/*` i `*/`. Taki komentarz może obejmować wiele wierszy kodu. Drugi sposób polega na umieszczeniu pary znaków `//` na początku komentarza. Ten komentarz kończy się wraz z końcem wiersza, w którym został umieszczony. Trzecie rozwiązanie polega na umieszczeniu treści komentarza między dwoma wierszami zawierającymi następujące pary instrukcji preprocesora: `#if 0` i `#endif`.

Komentarze

Przykłady

Listing 3: Komentarze w C

```
1  /* To jest komentarz pierwszego rodzaju. Nie można w nim umieszczać
2  innych komentarzy tego samego typu. */
3
4  /*
5  * To też jest komentarz pierwszego rodzaju, ale czytelniej
6  * zapisany.
7  */
8
9  // To jest komentarz drugiego rodzaju.
10 // Taki komentarz może być umieszczony w komentarzach innego
11 // typu.
12
13 #if 0
14 Tak wygląda trzeci rodzaj komentarzy w języku C.
15 Jak widać, taki komentarz może obejmować wiele wierszy kodu.
16 /* A nawet można w nim zagnieżdżać komentarze pierwszego
17 rodzaju. */
18 #endif
```

Wady i zalety komentarzy

+ ułatwiają zrozumienie kodu,

Wady i zalety komentarzy

+ ułatwiają zrozumienie kodu,
- mogą być oznaką tego, że dany fragment kodu, którego dotyczą nie jest dostatecznie dobrze zapisany,

Wady i zalety komentarzy

+ ułatwiają zrozumienie kodu,
- mogą być oznaką tego, że dany fragment kodu, którego dotyczą nie jest dostatecznie dobrze zapisany,
- zmiany w kodzie źródłowym, których dotyczy komentarz powodują, że komentarz staje się przestarzały,

Wady i zalety komentarzy

+ ułatwiają zrozumienie kodu,
  - mogą być oznaką tego, że dany fragment kodu, którego dotyczą nie jest dostatecznie dobrze zapisany,
  - zmiany w kodzie źródłowym, których dotyczy komentarz powodują, że komentarz staje się przestarzały,
+ pozwalają tymczasowo wyłączyć fragmenty kodu, które mogą być niepoprawnie napisane.

Stałe

Stałe podobnie jak w matematyce, czy fizyce są nazwami nadawanymi pewnym konkretnym, niezmiennym wartościom. W przypadku języka C stałe mogą być definiowane na kilka różnych sposobów. Pierwszym jaki poznamy jest wykorzystanie tak zwanych makr preprocesora.

Wzorzec

```
#define NAZWA WARTOŚĆ
```

Przykład

```
#define GRAVITY 9.81
```

Nazwy stałych najczęściej w całości pisane są wielkimi literami. Wszędzie tam, gdzie kompilator (preprocesor) znajdzie w kodzie źródłowym identyfikator GRAVITY zastąpi go wartością 9.81.

Zmienne

Zmienna, to miejsce, w którym program komputerowy przechowuje dane, które przetwarza. Zmienna z punktu widzenia komputera jest wydzielonym fragmentem pamięci operacyjnej. Posiada ona trzy własności (atrybuty): nazwę, zasięg i typ. Nazwa jest *identyfikatorem* zmiennej i została dokładniej opisana na kolejnych slajdach. Zasięg określa, które fragmenty kodu mogą korzystać ze zmiennej i jest on zależny od miejsca, gdzie zmienna została zadeklarowana. Najpierw zapoznamy się ze zmiennymi globalnymi, które są dostępne w każdym miejscu programu. Typ definiuje rozmiar zmiennej (ile pamięci zajmuje dana zmienna) i jakiego rodzaju informacje ona zawiera.

Wzorzec deklaracji zmiennej

Zanim będzie można użyć zmiennej w programie musi ona być najpierw zadeklarowana.

```
typ_zmiennej nazwa_zmiennej;
```

Przykład

Listing 4: Przykład deklaracji zmiennej

```
1  #include<stdio.h>
2
3  unsigned int number_of_students;
4
5  int main(void)
6  {
7          return 0;
8  }
```

Nazwa zmiennej

Reguły

Nazwa zmiennej jest identyfikatorem, który pozwala ją jednoznacznie określić w programie. Podobnie jak wszystkie identyfikatory w programie jest ona tworzona zgodnie z następującymi regułami:

- nazwy zmiennych nie mogą się powtarzać (są pewne wyjątki od tej reguły),
- nazwa nie może się zaczynać cyfrą,
- w nazwie nie mogą występować znaki specjalne (niebędące cyframi lub literami), poza podkreśleniem (_),
- język C rozróżnia wielkość liter (ang. *case sensitive*),
- nazwy zmiennych nie mogą zawierać polskich znaków,
- nazwa nie może być słowem kluczowym języka (elementem składni języka, takim jak np. nazwa typu).

Nazwa zmiennej

Porady

Reguły podane na poprzednim slajdzie są przestrzegane przez kompilator. Wypisane poniżej porady nie dotyczą wymagań kompilatora, ale wprowadzają pewną konwencję, która zwiększa czytelność programu. **Należy o nią dbać, bo kody źródłowe programów są znacznie częściej czytane niż pisane.**

- Nazwy zmiennych powinny być czytelne.
- Nazwy zmiennych powinny zawierać rzeczownik.
- Jeśli zmienne składają się z kilku wyrazów, to powinny być one połączone znakiem podkreślenia i każdy z nich powinien się zaczynać małą literą.
- Nazwy jednoliterowe należy używać tylko w przypadku niektórych konstrukcji, które będą omawiane na kolejnych wykładach, lub kiedy są one jednoznaczne, np. używane w dobrze znanym wzorze matematycznym.

Typy zmiennych

Najczęściej używane typy

Nazwa	Rozmiar (w bajtach)	Wartości
`int`	4	liczby całkowite
`short int` lub `short`	2	liczby całkowite
`long int` lub `long`	8	liczby całkowite
`long long int` lub `long long`	8	liczby całkowite
`char`	1	znaki lub liczby całkowite
`float`	4	liczby zmiennoprzecinkowe
`double`	8	liczby zmiennoprzecinkowe
`long double`	12	liczby zmiennoprzecinkowe

Jeden bajt to 8 bitów. Bit to najmniejsza porcja informacji, jaką może przetwarzać komputer (0 lub 1). Podane w tabeli rozmiary dotyczą 64-bitowego komputera klasy PC. Standard języka nie definiuje wprost rozmiarów zmiennych, a jedynie zależności między nimi.

System dwójkowy i pokrewne (podstawy)

System dziesiętny

$$128_{(\mathrm{DEC})} = 1 \cdot 10^2 + 2 \cdot 10^1 + 8 \cdot 10^0$$

System dwójkowy i pokrewne (podstawy)

System dziesiętny

$$128_{(\mathrm{DEC})} = 1 \cdot 10^2 + 2 \cdot 10^1 + 8 \cdot 10^0$$

System dwójkowy

$$1001_{(\mathrm{NKB})} = 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 9_{(\mathrm{DEC})}$$

System dwójkowy i pokrewne (podstawy)

System dziesiętny

$$128_{(\mathrm{DEC})} = 1 \cdot 10^2 + 2 \cdot 10^1 + 8 \cdot 10^0$$

System dwójkowy

$$1001_{(\mathrm{NKB})} = 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 9_{(\mathrm{DEC})}$$

System dwójkowy uzupełnienie do 2

$$(-5)_{(\mathrm{DEC})} \Rightarrow \overline{0101}_{(\mathrm{NKB})} \Rightarrow 1010_{(\mathrm{U1})} + 1 \Rightarrow 1011_{(\mathrm{U2})}$$

Typy danych

Zakresy typów całkowitych dla komputerów 64-bitowych

Nazwa	Wartość minimalna	Wartość maksymalna
`int`	-2 147 483 648	2 147 483 647
`short int` lub `short`	-32 768	32 767
`long int` lub `long`	−9 223 372 036 854 775 808	9 223 372 036 854 775 807
`long long int` lub `long long`	−9 223 372 036 854 775 808	9 223 372 036 854 775 807
`char`	-128	127

Typy danych

Znaki

Pojedyncze znaki, tj. litery lub cyfry i znaki niealfanumeryczne są przechowywane w zmiennej typu `char`. Każda wartość w tej zmiennej jest traktowana jako kod znaku nazywany kodem ASCII (American Standard Code for Information Interchange).

Typy danych

Specyfikatory

Specyfikatory są słowami kluczowymi języka C, które pozwalają zmodyfikować znaczenie niektórych typów zmiennych. Specyfikator `unsigned` stosowany jest razem z typami `int` i `char` i powoduje, że zmienna tego typu przechowuje tylko liczby naturalne. Innymi słowy: zamienia typ całkowity na naturalny. Specyfikator `signed` jest uzupełnieniem dla `unsigned` i powoduje, że zmienna przechowuje wartości całkowite. W większości przypadków jest on pomijany w deklaracji zmiennej, choć zaleca się stosowanie go dla zmiennych typu `char`, jeśli ma w nich być przechowywana liczba całkowita, a nie znak. Specyfikator `long` powoduje zwiększenie pojemności typów `int` i `double`. Specyfikator `short` powoduje zmniejszenie pojemności typu `int`.

Typy danych

Zakresy typów naturalnych dla komputerów 64-bitowych

Nazwa	Wartość minimalna	Wartość maksymalna
`unsigned int` lub `unsigned`	0	4 294 967 295
`unsigned short int` lub `unsigned short`	0	65 535
`unsigned long int` lub `unsigned long`	0	18 446 744 073 709 551 615
`unsigned long long int` lub `unsigned long long`	0	18 446 744 073 709 551 615
`unsigned char`	0	255

W pliku nagłówkowym `limits.h` zdefiniowano odpowiednie stałe dla każdej granicznej wartości typów zarówno całkowitych, jak i naturalnych.

Typy danych

Typ `void`

Słowo kluczowe `void` używane jest jako określenie typu danych, ale nie jest typem zmiennej. Nie można zadeklarować zmiennej tego typu. Jest ono jednak przydatne w innych sytuacjach, które będą omawiane na przyszłych wykładach.

Typy danych

Typy zmiennoprzecinkowe

W pamięci komputera nie można odwzorować dokładnie liczby rzeczywistych. Dlatego stworzono ich analog w postaci liczb zmiennoprzecinkowych. Podstawą dla tego zapisu jest notacja wykładnicza, gdzie podstawa potęgi jest zawsze równa dwa. W pamięci komputera te liczby mogą być przechowywane np. następująco: pierwszy bit oznacza znak liczby, następna część część ułamkową, a ostatnia wykładnik. Część ułamkowa jest zapisana w postaci binarnej, gdzie każda kolejna pozycja, począwszy od lewej stron ma ujemny wykładnik wagi. Wyjątkiem jest pierwsza pozycja, która ma wykładnik równy zero. W przypadku języka C, te liczby przechowywane są w zmiennych typu `float`, `double` i `long double`. Różnica między nimi polega nie tylko na tym ile miejsca w pamięci zajmują w całości, ale też na tym ile z niego przeznaczają na zapamiętanie części ułamkowej i wykładnika.

Typy danych

Typy logiczny

W języku C przyjmuje się, że wszystko co ma wartość różną od zera jest prawdą, a to, co ma wartość zero jest fałszem. We wcześniejszych standardach tego języka nie przewidziano miejsca dla typu logicznego. Dopiero standard ISO C99 wprowadził typ `bool` i stałe `true` i `false`, ale nie jako integralną część języka, tylko jako zewnętrzny dodatek dostępny po dodaniu instrukcji `#include<stdbool.h>` na początku programu.

Pytania

?

KONIEC

Dziękuję Państwu za uwagę!