Polsko-Japońska Wyższa Szkoła Technik Komputerowych

Instrukcja do laboratorium Systemów Operacyjnych

Ćwiczenie 5

Temat: Wprowadzenie do programowania w C

Opracował:
mgr. inż Arkadiusz Adolph

Wprowadzenie

I. Historia C

Poprzednikiem C był język B. W latach 1969-1973 język rozwijał się, aż w 1973 roku udało się zaimplementować w tym języku jądro systemu Unix. W 1978 r. B. Kernighan i D. Ritchie opublikowali książkę "C Programming Language". W Polsce można tą książkę kupić pod tytułem ANSI C, Wydawnictwa Naukowo - Technicznego. Bardziej popularny język C stał się po 1980 roku i stał się jednym z najbardziej popularnych języków do pisania systemów operacyjnych. To na jego bazie powstał w latach osiemdziesiątych język C++, wprowadzając programowanie obiektowe. Standard języka C został opisany w normie ISO 9899 w 1990 roku, i był modyfikacją standardu ANSI X3. Język zgodny z tą wersją często jest określany jako C89. Najnowsza wersja powstała w 1999 roku i jest określana jako C99. Jest jednak niekompatybilna z C++.

II. Struktura programu w C

#include "nazwa_pliku"
//wstawianie plików - w/w wiersz jest zastępowany plikiem o nazwie nazwa_pliku
#include <nazwa_pliku>
//efekt zastosowania tej instrukcji jw. z tym, że dodatkowo zleca się kompilatorowi poszukiwanie pliku w
//pewnym wyróżnionym katalogu (w Unix jest to skorowidz /usr/include)
defme YES 1
//zastąpienie nazwy przez ciąg znaków
main ()
{
.....
}

III. Podstawowe typy

słowo kluczowe	**Typ/Liczba**	**Rozmiar**	**Zakres**
char	znak	8 bitów (1 bajt)	0-255
int	l. całkowita	16 bitów (2 bajty)	-32768 ÷ 32767
int	l. całkowita	32 bity (4 bajty)	-2147483648 ÷ 2147483647
short	l. całkowita	16 bitów (2 bajty)	-32768 ÷ 32767
float	l. zmiennoprzecinkowa	4 bajty	1.2e-38 ÷ 3.4e+38
double	l. zm. o podwójnej precyzji	8 bajtów	2.3e-308 ÷ 1.7e+308
long double		10 bajtów	3.4e-4932 ÷ 1.1e+4932
void	brak wartości		

IV. Pierwszy program

```
#include <stdio.h>

int main()
{
	system("PAUSE");
	return 0;
}
```

V. Wczytywanie, zmienne i wypisywanie

```
#include <stdio.h>
/*Wczytywanie zmiennych*/
int main()
{
	int a;
	scanf("%d",&a);
	printf("Wartosc zmiennej a:");
	printf("%d",a);
	system("PAUSE");
	return 0;
}
```

```
#include <stdio.h>
/*Wczytywanie zmiennych*/
int main()
{
	int a;
	scanf("%d",&a);
	printf("Wartosc zmiennej a: %d",a);
	system("PAUSE");
	return 0;
}
```

VI. Sekwencje specjalne języka C

Lista ciągów znakow zastępujących niektóre sekwencje:

- \a znak alarmu
- \b znak cofania (ang. Backspace)
- \f znak nowej strony
- \n znak nowego wiersza
- \r znak powrotu karetki
- \t znak tabulacji poziomej
- \v znak tabulacji pionowej
- \\ znak \
- \? Znak zapytania
- \' znak apostrofu
- \" znak cudzysłowu
- \ooo liczba ósemkowa
- \xhh liczba szesnastkowa

Lista znakow przekształceń:

- %d - argument będzie przekształcony do postaci dziesiętnej
- %o - argument beędzie przekształcony do postaci ósemkowej
- %x - argument będzie przekształcony do postaci szesnastkowej
- %c - argument będzie traktowany jako jeden znak
- %s - argument jest tekstem znakowym
- %e - argument będzie traktowany jako liczba typu float lub double ([-]m.nnnnnE[+-]xx)
- %f - argument będzie traktowany jako liczba typu float lub double ([-]mmm.nnn)

VII. Instrukcje strujące

Instrukcja **if** (ang. jeśli) to podstawowa instrukcja warunkowa w C – gdy warunek1 jest spełniony (zwraca wartość niezerową), wykonany zostanie kod zawarty w bloku ograniczonym klamrami. Instrukcje else if i else są opcjonalne, sprawdzane są wyłącznie, gdy podstawowy warunek nie jest spełniony.

```
if (warunek1) {
   instrukcje;
} else if(warunek2){
   instrukcje;
} else {
   instrukcje;
}
```

Pętla **while** (ang. podczas gdy) – instrukcja wykonuje kod zawarty w bloku ograniczonym klamrami tak długo, dopóki jej warunek jest spełniony (ma wartość różną od zera). Instrukcja sprawdza warunek przed wykonaniem ciała pętli. Pętla while może wykonywać się nieskończoną ilość razy, gdy wyrażenie nigdy nie przyjmie wartości 0, może także nie wykonać się nigdy, gdy wartość przed pierwszym przebiegiem będzie zerowa.

```
while (wyrażenie) {
 instrukcje;
}
```

Pętla **do...while** (ang. wykonuj...dopóki) jest podobna do pętli while z tą różnicą, że warunek sprawdzany jest po każdym wykonaniu pętli, a więc instrukcje w pętli zawsze wykonają się co najmniej raz.

```
do {
instrukcje;
} while(warunek);
```

Pętla **for** (ang. dla) jest rozwinięciem pętli while o instrukcję wykonywaną przed pierwszym obiegiem oraz dodatkową instrukcję wykonywaną po każdym przebiegu – najczęściej służącą jako licznik obiegów. Często zmienną liczącą kolejne wykonania ciała pętli nazywa się iteratorem.

```
for (wyrażenie1; wyrażenie2; wyrażenie3) {
  instrukcje;
}
```

Instrukcją decyzyjną **switch** (ang. przełącznik) zastąpić można wielokrotne wywoływanie instrukcji warunkowej if np. dla różnych wartości tej samej zmiennej – przykładowo, gdy zmienna może przyjąć 10 różnych wartości, a dla każdej z nich należy podjąć inne działanie.

```
switch (wyrażenie) {
    case wartość1 :
      instrukcje;
      [break;]
    case wartość2 :
      instrukcje;
      [break;]
```

```
    default :
      instrukcje;
      [break;]
}
```

VIII. Funkcje

Funkcje w C tworzy się za pomocą następującej składni:

```
[klasa_pamieci] [typ] nazwa([lista_parametrów])
{
 instrukcje;
 [return wartość;]
}
```

Klasa pamięci, określenie zwracanego typu oraz lista parametrów są opcjonalne. Jeżeli nie podano typu, domyślnie jest to typ liczbowy int, a instrukcję return kończącą funkcję i zwracającą wartość do funkcji nadrzędnej można pominąć. Listę argumentów tworzą wszystkie zmienne (zarówno przekazywane przez wartość jak i wskaźniki) wraz z określeniem ich typu. Dozwolona jest rekurencja, nie ma natomiast możliwości przeciążania funkcji (wprowadzonego m.in. w C++).

Przykład

```
int kwadrat(int x)
{
 return x*x;
}
```

IX. Tablice

W języku C tablica jest listą (ciągiem) zmiennych, które są tego samego typu i do których można się odwołać za pomocą wspólnej nazwy. Pojedyncza zmienna w tablicy jest nazywana elementem tablicy. Tablice są prostym sposobem obsługiwania grup powiązanych danych.
Aby zdeklarować tablicę, należy użyć następującego schematu:

```
typ nazwa-tab[wielkość];
```

Typ musi być poprawnym typem języka C, nazwa-tab definiuje nazwę tablicy, a wielkość - liczbę jej elementów (rozmiar).

Element tablicy jest dostępny za pomocą indeksowania tablicy numerami elementów. W języku C tablica zaczyna się od indeksu zerowego. Oznacza to, że aby osiągnąć pierwszy element tablicy, należy użyć zera jako indeksu. Indeksowanie tablicy polega na podaniu numeru szukanego elementu w nawiasach kwadratowych.

Poniższy zapis odwołuje się do drugiego elementu tablicy:

```
mojatablica[1]
```

Pamiętaj o tym, że tablice rozpoczynają się od numeru 0, zatem indeks 1oznacza drugi element tablicy!

Aby nadać elementowi tablicy wartość, należy ją zapisać w instrukcji przypisania:

mojatablica[0] = 100

Powyższa instrukcja nadaje pierwszemu elementowi tablicy (o nazwie: mojatablica) wartość 100.

X. Operacje na plikach:

Aby otworzyć plik i związać z nim strumień, należy użyć funkcji fopen(). Wygląda ona następująco:

FILE *fopen(char *nazwa,char *tryb);

Funkcja fopen() korzysta z pliku nagłówkowego stdio.h. Nazwa otwieranego pliku jest wskazywana przez wskaźnik nazwa. Musi to być nazwa poprawna w używanym systemie plików. Możliwe wartości trybu znajdują się w poniższym zestawieniu:

Tryb	**Opis**
r	Otwieranie pliki tekstowego do odczytu
w	Tworzenie pliku tekstowego do zapisu
a	Dołączanie do pliku tekstowego
rb	Otwieranie pliku binarnego do odczytu
wb	Tworzenie pliku binarnego do zapisu
ab	Dołączanie do pliku binarnego
r+	Otwieranie pliku tekstowego do odczytu-zapisu
w+	Tworzenie pliku tekstowego do odczytu-zapisu
a+	Dołączanie do pliku tekstowego do odczytu-zapisu lub jego tworzenie
r+b	Otwieranie pliku binarnego do odczytu-zapisu; można także użyćrb+
w+b	Tworzenie pliku binarnego do odczytu-zapisu; można także użyćwb+
a+b	Dołączanie pliku binarnego do odczytu-zapisu lub jego tworzenie; możnat także użyćab+

Poprawnie wykonana funkcja fopen() zwraca wskaźnik null.

Aby zamknąć plik należy skorzystać z funkcji fclose(). Wygląda ona następująco:

int fclose(FILE *fp);

Funkcja fclose() zamyka plik związany ze wskaźnikiem fp, który musi być poprawnym wskaźnikiem otrzymanym w wyniku użycia funkcji fopen(), oraz odłącza strumień od pliku.

Nie wolno wywoływać funkcji fclose() z niepoprawnym argumentem. Może to spowodować uszkodzenie systemu plików i niedwracalną utratę danych. Po prawidłowym wykonaniu funkcji fclose() zwraca ona wartość zero. Jeśli wystąpi błąd zwraca ona EOF.

Zadania do wykonania:

1. Przeanalizaować program:

```
#include <stdio.h>
int main()
{
	float a,b,c;
	int wybor;
	do
	{
	printf("Co chcesz zrobic?\n1 – dodawanie\n2 – odejmowanie\n3 – mnozenie\n4
– dzielenie\ninne – konczy\n");
	scanf("%d",&wybor);
	switch (wybor)
	{
		case 1: /*Dodawanie*/
			{
				printf("Podaj 1 liczbe: ");
				scanf("%f",&a);
				printf("Podaj 2 liczbe: ");
				scanf("%f",&b);
				printf("Wartosc zmiennej a:\t%.2f\n",a);
				printf("Wartosc zmiennej b:\t%.2f\n",b);
				printf("Wartosc sumy:\t%.2f\n",a+b);
				break;
			}
		case 2: /*Odejmowanie*/
			{
				printf("Podaj 1 liczbe: ");
				scanf("%f",&a);
				printf("Podaj 2 liczbe: ");
				scanf("%f",&b);
				printf("Wartosc zmiennej a:\t%.2f\n",a);
				printf("Wartosc zmiennej b:\t%.2f\n",b);
				printf("Wartosc roznicy:\t%.2f\n",a–b);
				break;
			}
		case 3: /*Mnozenie*/
			{
				printf("Podaj 1 liczbe: ");
				scanf("%f",&a);
				printf("Podaj 2 liczbe: ");
				scanf("%f",&b);
				printf("Wartosc zmiennej a:\t%.2f\n",a);
				printf("Wartosc zmiennej b:\t%.2f\n",b);
				printf("Wartosc iloczynu:\t%.2f\n",a*b);
				break;
			}
		case 4: /*Dzielenie*/
			{
				printf("Podaj 1 liczbe: ");
				scanf("%f",&a);
				printf("Podaj 2 liczbe: ");
				scanf("%f",&b);
				printf("Wartosc zmiennej a:\t%.2f\n",a);
				printf("Wartosc zmiennej b:\t%.2f\n",b);
				printf("Wartosc ilorazu:\t%.2f\n",a/b);
				break;
			}
		default:	 break;
	}
```

```
      if (wybor!=1 && wybor!=2 && wybor!=3 && wybor!=4)
            break;
      printf("Czy chcesz powtorzyc?\n0 – konczy\n");
      scanf("%d",&wybor);
      }while(wybor!=0);
      system("PAUSE");
      return 0;
}
```