Podstawy programowania

I rok Automatyka i Robotyka Eka PWr

Ćwiczenia – Zestaw 4

Zakres materiału

Analiza poprawności konstrukcji, wyliczanie wyrażeń z wskaźnikami i tablicami, ręczna symulacja, opracowywanie funkcji programu.

Zadania

1. Mając zdefiniowane w programie

   ```
   int i=7, *p=&i, *t[10], **q1;
   ```

   odpowiedz na poniższe pytania.

   - Które z podstawień są poprawne:
     a) `q1=&i;`
     b) `q1=&(&i);`
     c) `q1=p;`
     d) `q1=&p;`
     e) `q1=*t[5];`
     f) `q1=&t[5];`
     g) `q1=&t;`
     h) `q1=t;`
   - Czy po wykonaniu `q1=t; q1++;` wartości `t[1]` i `*q1` są równe? Czy wynoszą one 7?

2. Załóżmy, że mamy następujące deklaracje:

   ```
   #define ROZMIAR 5
   int liczby[ROZMIAR]={3, 8, 10, 3, 9};
   int *wsk1, *wsk2;
   wsk1=liczby;
   wsk2=&liczby[3];
   ```

   (3: $a1$, 8: $a2$, 10: $a3$, 3: $a4$, 9: $a5-adresy$)

   określ wartość każdego z poniższych wyrażeń:

   a) `liczby+2==&liczby[2]`
   b) `*(liczby+2)==liczby[2]`
   c) `*(liczby+1)`
   d) `*liczby+2`
   e) `liczby`
   f) `*wsk1`
   g) `printf("%d,%p",*wsk1++,wsk1)`
   h) `printf("%d,%p",*++wsk1,wsk1)`

i) `(*wsk2)++`

j) `wsk1+4==&liczby[4]`

k) `wsk2-2!=&liczby[1]`

l) `*--wsk2`

m) `wsk2-wsk1`

n) `wsk2==wsk1`

o) `*wsk2==*wsk1`

3. Jaka jest wartość `*wsk` i `*(wsk+2)` w każdym przypadku?

a)
```
int *wsk;
int kulki[2][2]={12,14,16}
wsk=kulki[0]
```

b)
```
int *wsk;
int pilki[2][2]={{12},{14,16}}
wsk=pilki[0]
```

4. Ile wynosi `**wsk` i `**(wsk+1)` w każdym przypadku?

a)
```
int (*wsk)[2];
int kulki[2][2]={12,14,16}
wsk=kulki
```

b)
```
int (*wsk)[2];
int pilki[2][2]={{12},{14,16}}
wsk=pilki
```

5. Zadeklarowano zmienne:

```
char a[10]="0123456789";
char *pa;
char x, y, z;
```

Które z poniższych instrukcji są poprawne? Jeśli są poprawne, jaki będzie efekt ich działania.

```
pa = a;
pa++;
a++;
a=pa;
*pa++;
(*pa)++;

x = *pa;
y = *pa++;
z = *++pa;

printf("%c%c%c\n",x,y,z);
printf("%s\n",pa-2);
```

6. Podaj różnice pomiędzy poniższymi definicjami zmiennej `napis`

```
char napis[80] = "Ala ma kota";
char napis[]   = "Ala ma kota";
char *napis    = "Ala ma kota";
```

7. Przy definicjach:
```
int *zPtr;
int *aPtr;
void *sPtr = 0;
int liczba, i;
int z[5] = {1,2,3,4,5};
zPtr = z;
```
wykaż błędy w poniższych fragmentach programu i opisz w jaki sposób można je poprawić.

   a) `++aPtr;`

   b) Aby pobrać pierwszy element tablicy, należy skorzystać z konstrukcji
   `liczba = zPtr;`

   c) Przypisanie trzeciego elementu tablicy (o wartości 3) zmiennej `liczba` uzyskuje się przez
   `liczba=*zPtr[2];`

   d) Na wydrukowanie całej zawartości tablicy `z` pozwalają instrukcje
```
for(i=0; i<=5; i++)
  printf("%d\n", zPtr[i]);
```

   e) Przypisanie wartości wskazywanej przez `sPtr` zmiennej `liczba`
   `liczba = *sPtr;`

   f) `++z;`

8. Załóżmy, że liczby zmiennoprzecinkowe pojedynczej precyzji przechowywane są w 4 bajtach, natomiast pierwszy element tablicy znajduje się pod adresem 1002500. Wykonaj następujące operacje:

   a) Załóżmy, że stałej symbolicznej `ROZMIAR` została nadana wartość 10. Zadeklaruj tablicę liczb typu `float` o nazwie `liczby`, zawierającą 10 elementów i zainicjuj ją wartościami: 0.0, 1.1, 2.2, ... 9.9.

   b) Zadeklaruj wskaźnik `nPtr` wskazujący na obiektu typy `float`.

   c) Wydrukuj na standardowym wyjściu zawartość tablicy `liczby` posługując się indeksem. Posłuż się instrukcją `for`, zakładając, że została zdefiniowana kontrolująca ją zmienna całkowita `i`. Każdy z elementów tablicy powinien być wydrukowany z jednym miejscem po przecinku.

   d) Podaj przykład dwóch różnych wyrażeń, przypisujących zmiennej `nPtr` adres tablicy `liczby`.

   e) Wydrukuj zawartość tablicy `liczby` korzystając z notacji wskaźnik/przesunięcie oraz ze wskaźnika `nPtr`.

   f) Wydrukuj zawartość tablicy `liczby` korzystając z notacji wskaźnik/przesunięcie, gdy funkcję wskaźnika pełni nazwa tablicy.

   g) Wydrukuj zawartość tablicy `liczby` indeksując wskaźnik `nPtr`.

   h) Pokaż czwarty element tablicy `liczby` posługując się metodą indeksowania tablicy oraz notacją wskaźnik/przesunięcie, gdy funkcję wskaźnika pełni nazwa tablicy, a także metodą indeksowania wskaźnika oraz notacją wskaźnik/przesunięcie, gdy funkcję wskaźnika pełni `nPtr`.

   i) Zakładając, że `nPtr` wskazuje początek tablicy `liczby`, jaki adres jest określany przez `nPtr+8`? Jaka znajduje się tam wartość?

   j) Zakładając, że `nPtr` wskazuje na `liczby[5]`, jaki adres będzie wskazywany przez `nPtr` po wykonaniu operacji `nPtr-=4`? W jaki sposób jest przechowywana wartość pod tym adresem?

9. Czy następujący fragment kodu jest poprawny?

```
int main() {
        int *st;
        st = &st;
        return 0;
}
```

10. Co jest niepoprawnego w następującym fragmencie kodu i jak można to poprawić?

```
int main() {
        int *st;
        *st = 100;
        printf("%d\n", *st);
        return 0;
}
```

11. Pracujesz pewnie już długo. Odpocznij chwilę :)

12. Czy następujący fragment kodu poprawnie skompiluje się i wykona? Jeżeli tak, to co zostanie wyświetlone na ekranie?

```
int main() {
        int a = 5;
        a = *&*&*&*&a;
        printf("a wynosi %d\n", a);
        return 0;
}
```

13. Czy następujący fragment kodu poprawnie skompiluje się i wykona?

```
int main() {
        int a = 5;
        a = **&&a;
        printf("a wynosi %d\n", a);
        return 0;
}
```

14. Jaki wynik da następujący program?

```
int main() {
        char *p;
        for(p = "WNT"; *p; p++) printf("%c", *p - 1);
        printf("\n");
        return 0;
}
```

15. Napisz funkcje int `strlen(char *str)` i `int strlen(char str[])`, które zwracają długość napisu.

16. Napisz program, który w danej tablicy liczb całkowitych wyszuka element największy. Wszędzie, gdzie to jest możliwe wykorzystać wskaźniki.

17. Jeszcze jedna chwila relaksu nie zaszkodzi. Tym bardziej, że poniżej może być ciężko :)

18. Napisz funkcję `void dopisz(char *p, char *q)` która dopisuje ciąg znaków wskazywany przez `q` do ciągu znaków wskazywanego przez `p`. Przyjąć, że długość struktur danych służących do przechowywania ciągów wskazywanych przez `p` i `q` jest określona stałą `LENGTH`. Zaproponować przebieg testów funkcji.

19. Przeanalizuj poniższy program, a następnie odpowiedz na pytania. Odpowiedzi uzasadnij.

```
#include<stdio.h>

int main() {
  int i, t_i[15], *t_p1[15],*t_p2[15],*t_p3[15],*t_p4[15];

  for (i=0;i<=14;i++) {
    t_i[i]=i;
    t_p1[i]=i;
    t_p2[i]=&i;
    t_p3[i]=&t_i[i];
    t_p4[14-i]=t_i+i;
  }

  printf("%d\n",t_p1[10]);
  printf("%p\n",t_p1[10]);
  printf("%d\n",*(t_p1[10]));

  /* TEST */
  return 1;
}
```

a) Które linie zawierają polecenia powodujące błąd kompilacji? Które linie zawierają polecenia, które mogą powodować błąd w czasie wykonywania programu?

b) Jaką wartość mają `tp_2[5]`, `*tp_2[5]` w linii 18?

c) Jaką wartość mają `tp_3[5]`, `*tp_3[5]` w linii 18?

d) Co zostanie wypisane w linii 14? A co w linii 15?

e) Czy polecenie

```
printf("%d\n",*(t_i+*t_p4[*(t_i+3)]));
```

jest poprawne? Jeśli tak, to co zostałoby wypisane po umieszczeniu go w linii 18?

20. Programista miał napisać grę w przesuwankę na planszy 5x5 (na planszy znajdują się 24 elementy ponumerowane od 1 do 24, jedno pole jest puste, gra polega na ułożeniu elementów w kolejności). Jako sposób reprezentacji danych przyjął tablicę jednowymiarową zdefiniowaną jako `int plansza[25];` Przyjął także następujące oznaczenia kierunków przesuwania się wolnego pola na planszy: 1–góra, 2–prawo, 4–dół, 8–lewo. Pomóż mu zdefinować funkcje:

a) `z1d_na2d`, której pierwszym argumentem będzie indeks w tablicy, a będzie zwracać parę liczb będących współrzędnymi (x,y) elementu;

b) `z2d_na1d`, która na podstawie współrzędnych (x,y) wyliczy indeks tablicy;

c) `ruch`, która na podstawie aktualnego indeksu wolnego miejsca oraz kierunku wyliczy nowe położenie wolnego miejsca.

21. Nie ma to jak lista zawierająca dwudzieścia jeden zadań. Na szczęście to zadanie też pozwala na chwilę relaksu. Zwłaszcza, że to już koniec (takie info na wszelki wypadek, dla tych co nie zauważyli:).