|  |  |
| --- | --- |
|  | **UNIVERSIDADE CATÓLICA DE BRASÍLIA**  **ESCOLA DE EDUCAÇÃO, TECNOLOGIA E COMUNICAÇÃO**  **CURSOS DE TECNOLOGIA DA INFORMAÇÃO** |

|  |  |
| --- | --- |
| **CURSO:** Bacharelado em Ciências da Computação / Engenharia de Software | |
| **DISCIPLINA:** Programação Estruturada | **TURMA:** GPE02M0118 |
| **PROFESSORA:** Graziela Guarda | **SEMESTRE/ANO:** 2º/2019 |
| **ESTUDANTE:** Paulo César de Lima Rocha | **RA:** UC19105993 |

**Atividade Supervisionada (AS11)**

**Ponteiros**

1. Observe as seguintes afirmações, julgue V (verdadeiro) ou F (falso), no caso de afirmação F, justifique:

( **F** ) NULL é um outro nome para o delimitador de strings '\0'.

( **V** ) O operador **&** permite-nos obter o endereço de uma variável e permite também obter o endereço de um ponteiro.

( **F** ) Se x é um inteiro e ptr um ponteiro para inteiros e ambos contem no seu interior o número 100, então x+1 e ptr+1 apresentarão o número 101.

( **F** ) O operador asterisco (apontado por) permite saber qual o valor de um ponteiro.

Justificativas:

**a)NULL é essencialmente um #define para o valor “(void \*)0”, para que aponte para a memória zero. \0, por outro lado, é o terminador de *string*. Isto é importante já que C não guarda o tamanho da *string* de forma padronizada.**

**c) Toda manipulação de um ponteiro deve ser feita colocando um asterisco antes do nome do mesmo, caso contrário, o programa cria um erro fatal.**

**d) O operador asterisco só significa “apontado por” quando o ponteiro é declarado. Quando o objetivo é mostrar o conteúdo armazenado no ponteiro, é chamado de “conteúdo de”.**

1. Observe as seguintes sentenças e informe os valores:

**ptr**  **a** **b**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1002 |  | 5 |  |  | 7 |  |  |
| 1000 | 1001 | 1002 | ... | 3000 | 3001 | 3002 | ... |

\*ptr

1. Qual a saída da seguinte instrução? printf("%d %d %d", a, b, \*ptr);

**R:** **5 7 5**

1. Qual a saída depois de fazermos ptr = &b;?

**R:** **5 7 7**

1. Preencha a tabela abaixo com os valores correspondentes:

|  |  |
| --- | --- |
| **Expressão:** | **Valor:** |
| a | **5** |
| &a | **1002** |
| b | **7** |
| &b | **3001** |
| ptr | **5** |
| &ptr | **1000** |

1. Suponha a seguinte declaração: int x=2, \*px, \*py, y=3; que corresponde ao seguinte esquema de memória:

|  |  |  |
| --- | --- | --- |
|  |  | 106 |
|  |  | 105 |
|  |  | 104 |
| Y | 3 | 103 |
| py |  | 102 |
| px |  | 101 |
| X | 2 | 100 |

Suponha que a escrita de inteiros e ponteiros pode ser feita a través da função printf usando o formato %d.

1. Implemente um programa em C para colocar px apontando para x e py apontando para y.

**R: #include <stdio.h>**

**#include <stdlib.h>**

**main () {**

**int x, \*px = NULL, \*py = NULL, y;**

**x = 2;**

**y = 3;**

**px = &x;**

**py = &y;**

**return 0;**

**}**

1. Depois de executado o item a, qual a saída das seguintes instruções?

printf("%d %d\n", x, y);

printf("%d %d\n", \*px, \*py);

printf("%d %d\n", &px, &py);

**R: 2 3**

**2 3**

**101 102**

1. Caso seja feito px = py, qual a saída de:

printf("%d %d %d %d %d %d %d %d", x, &x, px, \*px, y, &y, py, \*py);

**R: 2 100 103 3 3 103 103 3**

1. Escreva um programa em C que utilize como parâmetro os seguintes dados: int x=2, y=3. Implemente os itens abaixo:
2. Mostrar na tela os endereços de memória de x, y.
3. Atribuir para os ponteiros px e py os endereços de x e y respectivamente.
4. Mostrar na tela os endereços apontados por x e y.
5. Mostrar na tela os valores de px e py.

**R: #include <stdio.h>**

**#include <stdlib.h>**

**main () {**

**int x = 2, y = 3;**

**int \*px = NULL, \*py = NULL;**

**px = &x;**

**py = &y;**

**printf("Enderecos de memoria das variaveis X e Y: %d %d\n", &x, &y);**

**printf("Enderecos de memoria apontados pelas variaveis X e Y: %d %d\n", px, py);**

**printf("Valores dos ponteiros pX e pY: %d %d\n", \*px, \*py);**

**return 0;**

**}**

1. Implemente um programa em C que utilize como parâmetro os seguintes dados: int x=7, y=9 e efetue as seguintes operações:
2. O resultado do incremento de x, y e px, py;
3. Mostre os endereços de memória de x, y, px e py.
4. O resultado do decremento entre py e px.

**R: #include <stdio.h>**

**#include <stdlib.h>**

**#include <locale.h>**

**main() {**

**setlocale(LC\_ALL, "Portuguese");**

**int x = 7, y = 9;**

**int \*px = NULL, \*py = NULL;**

**px = &x;**

**py = &y;**

**printf("%d %d %d %d\n", x, y, \*px, \*py);**

**printf("%d %d %d %d\n\n", &x, &y, px, py);**

**x++;**

**y++;**

**\*px++;**

**\*py++;**

**printf("%d %d %d %d\n", x, y, \*px, \*py);**

**printf("%d %d %d %d\n\n", &x, &y, px, py);**

**\*px--;**

**\*py--;**

**printf("%d %d %d %d\n", x, y, \*px, \*py);**

**printf("%d %d %d %d", &x, &y, px, py);**

**return 0;**

**}**

1. Escreva a função que retorna o endereço e a posição que ocorre a primeira vogal na string s. Caso não exista, retorna NULL. Protótipo: char \* firstVogal(char \* s).

**R: #include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**char\* firstVogal(char \*s);**

**int main() {**

**char\* s = NULL;**

**printf("Digite uma frase: ");**

**s=malloc(sizeof(char)\*100);**

**if(s == NULL){**

**fprintf(stderr, "%s\n","Error in malloc" );**

**exit(1);**

**}**

**fgets(s, 100, stdin);**

**char\* temp = firstVogal(s);**

**if (temp == NULL) {**

**printf("\nNao ha uma vogal na frase informada.");**

**} else {**

**printf("\nA posicao da primeira vogal na string e: %d", firstVogal(s) - &s[0] + 1);**

**printf("\nO endereco de memoria da primeira vogal e: %d\n", firstVogal(s));**

**}**

**return 0;**

**}**

**char \*firstVogal(char \*s) {**

**int i;**

**for(i = 0; s[i] != '\0'; i++) {**

**if(s[i] == 'a' || s[i] == 'A' || s[i] == 'e' || s[i] == 'E' || s[i] == 'i' || s[i] == 'I' || s[i] == 'o' || s[i] == 'O' || s[i] == 'u' || s[i] == 'U') {**

**return &s[i];**

**}**

**}**

**return NULL;**

**}**